GRAPE: Generalizable Replaceable Atomic Position Engine Reference Manual

Generated by Doxygen 1.2.5

Fri Aug 24 17:08:33 2001

Contents

Chapter 1

GRAPE: Generalizable Replaceable Atomic Position Engine Hierarchical Index

1.1 GRAPE: Generalizable Replaceable Atomic Position Engine Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

GRAPE: Generalizable Replaceable Atomic Position Engine Compound Index

2.1 GRAPE: Generalizable Replaceable Atomic Position Engine Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

GRAPE: Generalizable Replaceable Atomic Position Engine File Index

3.1 GRAPE: Generalizable Replaceable Atomic Position Engine File List

Here is a list of all files with brief descriptions:

Chapter 4

GRAPE: Generalizable Replaceable Atomic Position Engine Class Documentation

4.1 ABitSet Class Reference

`#include <abitset.h>`

Public Methods

- **ABitSet** ()
- **ABitSet** (**uint**)
- **ABitSet** (const ABitSet &)
- ~**ABitSet** ()
- const ABitSet& **operator=** (const ABitSet &)
- bool **Resize** (**uint** n)
- bool **IsAllSet** () const
- bool **IsAllClear** () const
- bool **IsEmpty** () const
- bool **operator[]** (**uint**) const
- bool **IsSet** (**uint** i) const
- void **Set** (**uint**)
- void **Clear** (**uint**)
- void **Invert** (**uint**)
- bool **In** (**uint** i) const
- void **Add** (**uint** i)
- void **Remove** (**uint** i)
- void **Set** (**uint**, **uint**)
- void **Clear** (**uint**, **uint**)
- void **Invert** (**uint**, **uint**)
- void **SetAll** ()
- void **ClearAll** ()
- void **InvertAll** ()

- const ABitSet& **operator|=** (const ABitSet &)
- const ABitSet& **operator &=** (const ABitSet &)
- const ABitSet& **operator^=** (const ABitSet &)
- const ABitSet& **operator-=** (const ABitSet &)
- ABitSet **operator|** (const ABitSet &) const
- ABitSet **operator &** (const ABitSet &) const
- ABitSet **operator^** (const ABitSet &) const
- ABitSet **operator-** (const ABitSet &) const
- bool **Increment** ()
- bool **Decrement** ()
- int **Compare** (const ABitSet &) const
- int **Number** () const
- int **Count** () const
- **MAKECOMPARISONS_WITH_COMPARE** (ABitSet)

Static Public Methods

- **uint DefaultSize** (**uint** s)
- **uint DefaultSize** ()

Static Protected Attributes

- **uint default_size** = 256

Private Attributes

- **uint num**
- char* **data**

Friends

- ostream& **operator<<** (ostream &os, const ABitSet &)
- ostream& **OutputBinary** (ostream &, const ABitSet &)
- istream& **InputBinary** (istream &, ABitSet &)
- ostream& **operator<=** (ostream &os, const ABitSet &bs)
- istream& **operator>=** (istream &is, ABitSet &bs)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 ABitSet::ABitSet ()

Definition at line 28 of file abitset.C.

```
: num(default_size)
{
        data = new char[b2c1(num)];
        ClearAll();

```

4.1.1.2 ABitSet::ABitSet (uint *n*)

Definition at line 35 of file abitset.C.

```
: num(n)
{
        if (num<1)
                num = 1;
        data = new char[b2c1(num)];
        ClearAll();

```

4.1.1.3 ABitSet::ABitSet (const ABitSet & *abs*)

Definition at line 44 of file abitset.C.

```
: num(abs.num)
{
        const uint x=b2c1(num);
        data = new char[x];
        memcpy(data, abs.data, x);

```

4.1.1.4 ABitSet::∼ABitSet ()

Definition at line 52 of file abitset.C.

```
{
        delete[] data;
        data = 0;
        num = 0;

```

4.1.2 Member Function Documentation

4.1.2.1 void ABitSet::Add (uint *i*) `[inline]`

Definition at line 75 of file abitset.h.

```
                                { Set(i); }

```

4.1.2.2 void ABitSet::Clear (uint *i*, uint *j*)

Definition at line 163 of file abitset.C.

```
{
        if (i>j)
                util_swap(i,j);
        if ( j>=num )
                return;

        for (uint k=i; k<=j; ++k)
                Clear(k);

```

4.1.2.3 void ABitSet::Clear (uint *i*)

Definition at line 133 of file abitset.C.

Referenced by Clear(), KAdjMat::Disconnect(), and Remove().

```
{
        if ( i>=num )
                return;

        data[b2c0(i)] &= ~b2i(i);

```

4.1.2.4 void ABitSet::ClearAll ()

Definition at line 215 of file abitset.C.

Referenced by ABitSet(), bf_iterator::reset(), and df_iterator::reset().

```
{
        memset(data, 0, b2c1(num));

```

4.1.2.5 int ABitSet::Compare (const ABitSet & *abs*) const

Definition at line 350 of file abitset.C.

```
{
        if (num != abs.num)
                return abs.num - num;

        uint    n = b2c1(num);

        for (uint i=0; i<n; ++i)
        {
                if (data[i] != abs.data[i])
                        return abs.data[i] - data[i];
        }

```

4.1.2.6 int ABitSet::Count () const

Definition at line 229 of file abitset.C.

```
{
        uint    n = b2c1(num);
        int     tot = 0;
        for (uint i=0; i<n; ++i)
                tot += bit_count(data[i]);
        return tot;

```

4.1.2.7 bool ABitSet::Decrement ()

Definition at line 330 of file abitset.C.

```
{
        uint    n = b2c1(num);
        bool            carry_flag = 1;
        uint    i = 0;

        while ( carry_flag && (i<n) )
        {
                carry_flag = 0;
                int     z = data[i];
                ++z;
                if (z & 256)
                        carry_flag = 1;
                data[i] = z & 255;
                ++i;
        }
```

4.1.2.8 uint ABitSet::DefaultSize () [static]

Definition at line 109 of file abitset.C.

```
{
        return default_size;
```

4.1.2.9 uint ABitSet::DefaultSize (uint *s*) [static]

Definition at line 99 of file abitset.C.

```
{
        uint    o = default_size;
        if (s<1)
                s = 1;
        default_size = s;
        return o;
```

4.1.2.10 bool ABitSet::In (uint *i*) const [inline]

Definition at line 73 of file abitset.h.

```
                                { return ($\ast$this)[i]; }
```

4.1.2.11 bool ABitSet::Increment ()

Definition at line 310 of file abitset.C.

```
{
        uint    n = b2c1(num);
        bool            carry_flag = 1;
        uint    i = 0;

        while ( carry_flag && (i<n) )
        {
```

```
                carry_flag = 0;
                int     z = data[i];
                ++z;
                if (z & 256)
                        carry_flag = 1;
                data[i] = z & 255;
                ++i;
        }

```

4.1.2.12 void ABitSet::Invert (uint *i*, uint *j*)

Definition at line 175 of file abitset.C.

```
{
        if (i>j)
                util_swap(i,j);
        if ( j>=num )
                return;

        for (uint k=i; k<=j; ++k)
                Invert(k);

```

4.1.2.13 void ABitSet::Invert (uint *i*)

Definition at line 142 of file abitset.C.

Referenced by Invert().

```
{
        if ( i>=num )
                return;

        data[b2c0(i)] ^= b2i(i);

```

4.1.2.14 void ABitSet::InvertAll ()

Definition at line 221 of file abitset.C.

```
{
        uint    n = b2c1(num);
        for (uint i=0; i<n; ++i)
                data[i] ^= ~0;

```

4.1.2.15 bool ABitSet::IsAllClear () const

Definition at line 198 of file abitset.C.

Referenced by IsEmpty().

```
{
        uint    n = b2c1(num);
```

```
00201         if (n>1)
00202                 for (uint i=0; i<n-1; ++i)
00203                         if (data[i] != 0)
00204                                 return 0;
00205         return (data[n-1] & all_set[num%8]) == '\x00';
00206
```

4.1.2.16 bool ABitSet::IsAllSet () const

Definition at line 187 of file abitset.C.

Referenced by bf_iterator::operator++(), and df_iterator::operator++().

```
00188 {
00189         uint    n = b2c1(num);
00190         if (n>1)
00191                 for (uint i=0; i<n-1; ++i)
00192                         if (data[i] != '\xFF')
00193                                 return 0;
00194         return (data[n-1] & all_set[num%8]) == all_set[num%8];
00195
```

4.1.2.17 bool ABitSet::IsEmpty () const [inline]

Definition at line 62 of file abitset.h.

```
00063                         { return IsAllClear(); }
00064
```

4.1.2.18 bool ABitSet::IsSet (uint *i*) const [inline]

Definition at line 66 of file abitset.h.

Referenced by sdGraph::TravLists(), bf_iterator::operator++(), and df_iterator::operator++().

```
00067                                 { return ($\ast$this)[i]; }
00068
```

4.1.2.19 ABitSet::MAKECOMPARISONS_WITH_COMPARE (ABitSet)

4.1.2.20 int ABitSet::Number () const [inline]

Definition at line 102 of file abitset.h.

```
00103                         { return num; }
00104
```

4.1.2.21 void ABitSet::Remove (uint *i*) [inline]

Definition at line 77 of file abitset.h.

```
00078                                 { Clear(i); }
00079
```

4.1.2.22 bool ABitSet::Resize (uint *n*)

Definition at line 80 of file abitset.C.

```
00081 {
00082         if (n<1)
00083                 n = 1;
00084         if (n==num)
00085                 return 1;
00086 
00087         int     m = util_min(n,num);
00088         char*   newd = new char[n];
00089         if (!newd)
00090                 return 0;
00091 
00092         memcpy (newd, data, m);
00093         delete[] data;
00094         data = newd;
00095         return 1;
00096
```

4.1.2.23 void ABitSet::Set (uint *i*, uint *j*)

Definition at line 151 of file abitset.C.

```
00152 {
00153         if (i>j)
00154                 util_swap(i,j);
00155         if ( j>=num )
00156                 j=num-1;
00157 
00158         for (uint k=i; k<=j; ++k)
00159                 Set(k);
00160
```

4.1.2.24 void ABitSet::Set (uint *i*)

Definition at line 124 of file abitset.C.

Referenced by Add(), KAdjMat::Connect(), Set(), bf_iterator::operator++(), df_iterator::operator++(), bf_iterator::reset(), and df_iterator::reset().

```
00125 {
00126         if ( i>=num )
00127                 return;
00128 
00129         data[b2c0(i)] |= b2i(i);
00130
```

4.1.2.25 void ABitSet::SetAll ()

Definition at line 209 of file abitset.C.

```
00210 {
00211         memset(data, ~0, b2c1(num));
00212
```

4.1.2.26 ABitSet ABitSet::operator & (const ABitSet & *abs*) const

Definition at line 289 of file abitset.C.

```
{
        ABitSet res(*this);
        return res &= abs;

```

4.1.2.27 const ABitSet & ABitSet::operator &= (const ABitSet & *abs*)

Definition at line 250 of file abitset.C.

```
{
        uint    n = b2c1(util_min(num, abs.num));

        for (uint i=0; i<n; ++i)
                data[i] &= abs.data[i];

        return *this;

```

4.1.2.28 ABitSet ABitSet::operator- (const ABitSet & *abs*) const

Definition at line 296 of file abitset.C.

```
{
        ABitSet res(*this);
        return res -= abs;

```

4.1.2.29 const ABitSet & ABitSet::operator-= (const ABitSet & *abs*)

Definition at line 261 of file abitset.C.

```
{
        uint    n = b2c1(util_min(num, abs.num));
        for (uint i=0; i<n; ++i)
                data[i] &= ~(abs.data[i]);

        return *this;

```

4.1.2.30 const ABitSet & ABitSet::operator= (const ABitSet & *abs*)

Definition at line 60 of file abitset.C.

```
{
        if (this == &abs)
                return *this;

        const uint x=b2c1(abs.num);

        if (b2c1(num) < x)
        {
```

```
                delete[] data;
                data = new char[x];
        }

```

4.1.2.31 bool ABitSet::operator[] (uint *i*) const

Definition at line 115 of file abitset.C.

```
{
        if ( i>=num )
                return 0;

        return data[b2c0(i)] & b2i(i);

```

4.1.2.32 ABitSet ABitSet::operator^ (const ABitSet & *abs*) const

Definition at line 303 of file abitset.C.

```
{
        ABitSet res(*this);
        return res ^= abs;

```

4.1.2.33 const ABitSet & ABitSet::operator^= (const ABitSet & *abs*)

Definition at line 271 of file abitset.C.

```
{
        uint    n = b2c1(util_min(num, abs.num));

        for (uint i=0; i<n; ++i)
                data[i] ^= abs.data[i];

        return *this;

```

4.1.2.34 ABitSet ABitSet::operator| (const ABitSet & *abs*) const

Definition at line 282 of file abitset.C.

```
{
        ABitSet res(*this);
        return res |= abs;

```

4.1.2.35 const ABitSet & ABitSet::operator|= (const ABitSet & *abs*)

Definition at line 239 of file abitset.C.

```
{
        uint    n = b2c1(util_min(num, abs.num));

        for (uint i=0; i<n; ++i)
                data[i] |= abs.data[i];

        return *this;

```

4.1.3 Friends And Related Function Documentation

4.1.3.1 istream& InputBinary (istream & *is*, ABitSet & *bs*) `[friend]`

Definition at line 448 of file abitset.C.

Referenced by InputBinary(), and operator>=().

```
{
        if (!is)
                return is;

        uint    n;
        InputBinary(is,n);

        delete[] bs.data;
        bs.num = n;
        bs.data = new char[b2c1(n)];
        if (!bs.data)
                raise(SIGQUIT);

        is.read(bs.data, b2c1(bs.num));
        return is;

```

4.1.3.2 ostream& OutputBinary (ostream & *os*, const ABitSet & *bs*) `[friend]`

Definition at line 436 of file abitset.C.

Referenced by OutputBinary(), and operator<=().

```
{
        if (!os)
                return os;
        OutputBinary(os,bs.num);
        if (!os)
                return os;
        os.write(bs.data, b2c1(bs.num));
        return os;

```

4.1.3.3 ostream& operator<< (ostream & *os*, const ABitSet & *abs*) `[friend]`

Definition at line 367 of file abitset.C.

```
{
        os << '<';
        for (uint i=0; i<abs.num; ++i)
                os << (abs[i] ? '1' : '0');
```

```
    os << '>';
    return os;
```

4.1.3.4 ostream & operator<= (ostream & *os*, const ABitSet & *bs*) [friend]

Definition at line 113 of file abitset.h.

```
            { return OutputBinary(os,bs); }
```

4.1.3.5 istream & operator>= (istream & *is*, ABitSet & *bs*) [friend]

Definition at line 115 of file abitset.h.

```
            { return InputBinary(is,bs); }
```

4.1.4 Member Data Documentation

4.1.4.1 char * ABitSet::data [private]

Definition at line 22 of file abitset.h.

4.1.4.2 uint ABitSet::default_size = 256 [static, protected]

Definition at line 9 of file abitset.C.

4.1.4.3 uint ABitSet::num [private]

Definition at line 21 of file abitset.h.

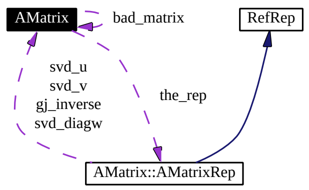
The documentation for this class was generated from the following files:

- **abitset.h**
- **abitset.C**

4.2 AMatrix Class Reference

`#include <amatrix.h>`

Collaboration diagram for AMatrix:



Public Types

- enum **init_type** { **garbage** = -1, **zero** = 0, **ident**, **rand** }
- enum **coord_type** { **rect_c** = 0, **spher_c**, **polar_c** = spher_c, **spher2_c**, **polar2_c** = spher2_c, **cyl_c**, **cyl2_c** }

Public Methods

- **AMatrix** ()
- **AMatrix** (int r, int c, double d)
- **AMatrix** (int r, int c, **init_type** which=garbage)
- **AMatrix** (const AMatrix &m)
- **~AMatrix** ()
- const AMatrix& **operator=** (const AMatrix &)
- AMatrix **UniqueCopy** () const
- AMatrix **Inverse** () const
- AMatrix **SVD_U** () const
- AMatrix **SVD_W** () const
- AMatrix **SVD_DiagW** () const
- AMatrix **SVD_V** () const
- AMatrix **SVDInverse** () const
- AMatrix **SVD_Fit** (int num_funcs, void(∗funcer)(double x, double ∗dest, int nf), double &chisq) const
- AMatrix **GJSolve** (const AMatrix &b) const
- AMatrix **GJInverse** () const
- AMatrix **LinearSolve** (const AMatrix &b) const
- AMatrix **Transpose** () const
- double& **operator()** (int r, int c)
- double **operator()** (int r, int c) const
- double& **operator()** (int r)
- double **operator()** (int r) const
- double& **operator**[] (int c)
- double **operator**[] (int c) const
- int **Rows** () const
- int **Cols** () const
- int **Compare** (const AMatrix &m) const
- bool **operator==** (const AMatrix &m) const

- bool **operator!=** (const AMatrix &m) const
- bool **operator<** (const AMatrix &m) const
- bool **operator>** (const AMatrix &m) const
- bool **operator<=** (const AMatrix &m) const
- bool **operator>=** (const AMatrix &m) const
- bool **IsBadMatrix** () const
- double **Lp_Distance** (AMatrix m, double p=0.0) const
- double **Lp_Metric** (double p) const
- double **Lp_Distance** (AMatrix m, double p, int minx, int maxx, int miny, int maxy) const
- double **Lp_Metric** (double p, int minx, int maxx, int miny, int maxy) const
- double **ModLp_Distance** (AMatrix m, AMatrix modulus, double p=0.0) const
- double **ModLp_Metric** (AMatrix modulus, double p) const
- double **ModLp_Distance** (AMatrix m, AMatrix modulus, double p, int minx, int maxx, int miny, int maxy) const
- double **ModLp_Metric** (AMatrix modulus, double p, int minx, int maxx, int miny, int maxy) const
- const AMatrix& **operator+=** (double d)
- const AMatrix& **operator-=** (double d)
- const AMatrix& **operator *=** (double d)
- const AMatrix& **operator/=** (double d)
- AMatrix **operator+** (double d) const
- AMatrix **operator-** (double d) const
- AMatrix **operator *** (double d) const
- AMatrix **operator/** (double d) const
- const AMatrix& **operator+=** (const AMatrix &m)
- const AMatrix& **operator-=** (const AMatrix &m)
- const AMatrix& **operator *=** (const AMatrix &m)
- const AMatrix& **operator/=** (const AMatrix &m)
- AMatrix **operator+** (const AMatrix &m) const
- AMatrix **operator-** (const AMatrix &m) const
- AMatrix **operator *** (const AMatrix &m) const
- AMatrix **operator/** (const AMatrix &m) const
- AMatrix **Filter** (double d=**epsilon**, double repl=0.0) const
- AMatrix **ModNormalize** (AMatrix modulus) const
- void **Reset** () const
- AMatrix **Embed** (const AMatrix &m, int r=1, int c=1)
- AMatrix **Submatrix** (int r, int c, int rlen, int clen) const
- AMatrix **ConvertFromRect** (**coord_type** to) const
- AMatrix **ConvertToRect** (**coord_type** from) const
- AMatrix **Convert** (**coord_type** from, **coord_type** to) const

Static Public Methods

- AMatrix **RotationMatrix** (int n, int a, int b, double rad)
- void **Set_Begin_Row** (const char *)
- void **Set_End_Row** (const char *)

Static Public Attributes

- double **epsilon**
- char* **os_begin_row**
- char* **os_end_row**

Protected Methods

- void **calculate_gj_inverse** () const
- void **calculate_svd** () const
- void **mod** ()

Static Protected Methods

- AMatrix **SmallProduct** (const AMatrix &m1, const AMatrix &m2)
- AMatrix **BigProduct** (const AMatrix &m1, const AMatrix &m2)
- void **converttorect** (const AMatrix &, AMatrix &, int, **coord_type**)
- void **convertfromrect** (const AMatrix &, AMatrix &, int, **coord_type**)

Static Protected Attributes

- const int **product_threshold**
- const AMatrix **bad_matrix**
- double **bad_val**

Private Attributes

- **AMatrixRep* the_rep**

Friends

- AMatrix **operator+** (double d, const AMatrix &m)
- AMatrix **operator-** (double d, const AMatrix &m)
- AMatrix **operator *** (double d, const AMatrix &m)
- AMatrix **operator/** (double d, const AMatrix &m)
- ostream& **operator<<** (ostream &os, const AMatrix &m)
- istream& **operator>>** (istream &is, AMatrix &m)
- ostream& **operator<=** (ostream &, const AMatrix &)
- istream& **operator>=** (istream &, AMatrix &)
- ostream& **OutputBinary** (ostream &, const AMatrix &)
- istream& **InputBinary** (istream &, AMatrix &)

4.2.1 Member Enumeration Documentation

4.2.1.1 enum AMatrix::coord_type

Enumeration values:

rect_c

spher_c

polar_c

spher2_c

polar2_c

cyl_c

cyl2_c

Definition at line 68 of file amatrix.h.

```
00068                             {
00069                                  rect_c=0, spher_c, polar_c=spher_c,
00070
```

4.2.1.2 enum AMatrix::init_type

Enumeration values:

garbage

zero

ident

rand

Definition at line 67 of file amatrix.h.

```
00067                           { garbage=-1, zero=0, ident, rand }
00068
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AMatrix::AMatrix () [inline]

Definition at line 56 of file amatrix.inc.

```
00057 {
00058         the_rep = new AMatrixRep;
00059
```

4.2.2.2 AMatrix::AMatrix (int *r*, int *c*, double *d*)

4.2.2.3 AMatrix::AMatrix (int *r*, int *c*, init_type *which* = garbage)

4.2.2.4 AMatrix::AMatrix (const AMatrix & *m*) [inline]

Definition at line 60 of file amatrix.inc.

```
00061 {
00062         the_rep = m.the_rep;
00063         the_rep->Reference();
00064 
```

4.2.2.5 AMatrix::~AMatrix () [inline]

Definition at line 66 of file amatrix.inc.

```
00067 {
00068         the_rep->Dereference();
00069         the_rep = 0;
00070 
```

4.2.3 Member Function Documentation

4.2.3.1 AMatrix AMatrix::BigProduct (const AMatrix & *m1*, const AMatrix & *m2*) [static, protected]

Referenced by operator ∗().

4.2.3.2 int AMatrix::Cols () const [inline]

Definition at line 303 of file amatrix.h.

Referenced by Compare(), GJInverse(), operator ∗(), operator ∗=(), operator()(), operator/(), operator/=(), and operator[ ]().

```
00303                           { return the_rep->cols; }
00304 
```

4.2.3.3 int AMatrix::Compare (const AMatrix & *m*) const [inline]

Definition at line 190 of file amatrix.inc.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

```
00191 {
00192         if (the_rep == m.the_rep)
00193                 return 0;
00194 
00195         if (Rows() > m.Rows())
00196                 return 1;
00197         else if (Rows() < m.Rows())
00198                 return -1;
00199 
00200         if (Cols() > m.Cols())
00201                 return 1;
00202         else if (Cols() < m.Cols())
00203                 return -1;
00204 
00205         for (int i=1; i<=the_rep->rows; ++i)
00206                 for (int j=1; j<=the_rep->cols; ++j)
00207                 {
00208                         if ( the_rep->operator()(i,j) > m.the_rep->operator()(i,j) )
```

```
                                return 1;
                        else if ( the_rep->operator()(i,j) < m.the_rep->operator()(i,j) )
                                return -1;
                }

```

4.2.3.4 AMatrix AMatrix::Convert (coord_type *from*, coord_type *to*) const

4.2.3.5 AMatrix AMatrix::ConvertFromRect (coord_type *to*) const

4.2.3.6 AMatrix AMatrix::ConvertToRect (coord_type *from*) const

4.2.3.7 AMatrix AMatrix::Embed (const AMatrix & *m*, int *r* = 1, int *c* = 1)

4.2.3.8 AMatrix AMatrix::Filter (double *d* = epsilon, double *repl* = 0.0) const

4.2.3.9 AMatrix AMatrix::GJInverse () const `[inline]`

Definition at line 287 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if (Rows() != Cols())
    {
#       ifdef   DEBUG_MATRIX
        log << "Error: " << Rows() << 'x' << Cols() << " GJInverse" << endl;
#       endif
        return bad_matrix;
    }

```

4.2.3.10 AMatrix AMatrix::GJSolve (const AMatrix & *b*) const

4.2.3.11 AMatrix AMatrix::Inverse () const

Referenced by operator/(), and operator/=().

4.2.3.12 bool AMatrix::IsBadMatrix () const `[inline]`

Definition at line 184 of file amatrix.inc.

```
{
        return (this->the_rep) == (bad_matrix.the_rep);

```

4.2.3.13 AMatrix AMatrix::LinearSolve (const AMatrix & *b*) const

4.2.3.14 double AMatrix::Lp_Distance (AMatrix *m*, double *p*, int *minx*, int *maxx*, int *miny*, int *maxy*) const

4.2.3.15 double AMatrix::Lp_Distance (AMatrix *m*, double *p* = 0.0) const

4.2.3.16 double AMatrix::Lp_Metric (double *p*, int *minx*, int *maxx*, int *miny*, int *maxy*) const

4.2.3.17 double AMatrix::Lp_Metric (double *p*) const

4.2.3.18 double AMatrix::ModLp_Distance (AMatrix *m*, AMatrix *modulus*, double *p*, int *minx*, int *maxx*, int *miny*, int *maxy*) const

4.2.3.19 double AMatrix::ModLp_Distance (AMatrix *m*, AMatrix *modulus*, double *p* = 0.0) const

4.2.3.20 double AMatrix::ModLp_Metric (AMatrix *modulus*, double *p*, int *minx*, int *maxx*, int *miny*, int *maxy*) const

4.2.3.21 double AMatrix::ModLp_Metric (AMatrix *modulus*, double *p*) const

4.2.3.22 AMatrix AMatrix::ModNormalize (AMatrix *modulus*) const

4.2.3.23 void AMatrix::Reset () const `[inline]`

Definition at line 361 of file amatrix.inc.

```
00362 {
00363         if (the_rep->have_svd_flag)
00364         {
00365                 the_rep->have_svd_flag = 0;
00366                 delete the_rep->svd_u;
00367                 the_rep->svd_u = 0;
00368                 delete the_rep->svd_diagw;
00369                 the_rep->svd_diagw = 0;
00370                 delete the_rep->svd_v;
00371                 the_rep->svd_v = 0;
00372         }
00373         if (the_rep->have_inverse_flag);
00374         {
00375                 the_rep->have_inverse_flag = 0;
00376                 delete the_rep->gj_inverse;
00377                 the_rep->gj_inverse = 0;
00378         }
00379
```

4.2.3.24 AMatrix AMatrix::RotationMatrix (int *n*, int *a*, int *b*, double *rad*) `[static]`

4.2.3.25 int AMatrix::Rows () const `[inline]`

Definition at line 300 of file amatrix.h.

Referenced by Compare(), GJInverse(), operator *(), operator *=(), operator()(), operator/(), operator/=(), and operator[]().

```
                            { return the_rep->rows; }
```

4.2.3.26 AMatrix AMatrix::SVDInverse () const

4.2.3.27 AMatrix AMatrix::SVD_DiagW () const [inline]

Definition at line 304 of file amatrix.inc.

```
{
        if (!the_rep->have_svd_flag)
                calculate_svd();
        return *(the_rep->svd_diagw);
```

4.2.3.28 AMatrix AMatrix::SVD_Fit (int *num_funcs*, void(∗ *funcer*)(double x, double ∗dest, int nf), double & *chisq*) const

4.2.3.29 AMatrix AMatrix::SVD_U () const [inline]

Definition at line 311 of file amatrix.inc.

```
{
        if (!the_rep->have_svd_flag)
                calculate_svd();
        return *(the_rep->svd_u);
```

4.2.3.30 AMatrix AMatrix::SVD_V () const [inline]

Definition at line 318 of file amatrix.inc.

```
{
        if (!the_rep->have_svd_flag)
                calculate_svd();
        return *(the_rep->svd_v);
```

4.2.3.31 AMatrix AMatrix::SVD_W () const

4.2.3.32 void AMatrix::Set_Begin_Row (const char ∗ *s*) [inline, static]

Definition at line 382 of file amatrix.inc.

```
{
        if (strlen(s) > strlen(os_begin_row))
        {
                delete[] os_begin_row;
                os_begin_row = new char[strlen(s)+1];
        }
```

4.2.3.33 void AMatrix::Set_End_Row (const char * *s*) [inline, static]

Definition at line 393 of file amatrix.inc.

```
{
        if (strlen(s) > strlen(os_end_row))
        {
                delete[] os_end_row;
                os_end_row = new char[strlen(s)+1];
        }
```

4.2.3.34 AMatrix AMatrix::SmallProduct (const AMatrix & *m1*, const AMatrix & *m2*) [static, protected]

Referenced by operator *().

4.2.3.35 AMatrix AMatrix::Submatrix (int *r*, int *c*, int *rlen*, int *clen*) const

4.2.3.36 AMatrix AMatrix::Transpose () const

4.2.3.37 AMatrix AMatrix::UniqueCopy () const [inline]

Definition at line 82 of file amatrix.inc.

```
{
        AMatrix m;
        m.the_rep->Dereference();
        m.the_rep = new AMatrixRep(*the_rep);
        return m;
```

4.2.3.38 void AMatrix::calculate_gj_inverse () const [protected]

Referenced by GJInverse().

4.2.3.39 void AMatrix::calculate_svd () const [protected]

Referenced by SVD_DiagW(), SVD_U(), and SVD_V().

4.2.3.40 void AMatrix::convertfromrect (const AMatrix &, AMatrix &, int, coord_type) [static, protected]

4.2.3.41 void AMatrix::converttorect (const AMatrix &, AMatrix &, int, coord_type) [static, protected]

4.2.3.42 void AMatrix::mod () [protected]

Referenced by operator *=(), operator()(), operator/=(), and operator[]().

4.2.3.43 AMatrix AMatrix::operator * (const AMatrix & *m*) const [inline]

Definition at line 248 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if (Cols() != m.Rows())
    {
#       ifdef   DEBUG_MATRIX
        log << "Error: " << Rows() << 'x' << Cols()
            << " * " << m.Rows() << 'x' << m.Cols() << endl;
#       endif
        return bad_matrix;
    }
```

4.2.3.44 AMatrix AMatrix::operator * (double *d*) const

4.2.3.45 const AMatrix & AMatrix::operator *= (const AMatrix & *m*) [inline]

Definition at line 268 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if (Cols() != m.Rows())
    {
#       ifdef   DEBUG_MATRIX
        log << "Error: " << Rows() << 'x' << Cols()
            << " *= " << m.Rows() << 'x' << m.Cols() << endl;
#       endif
        return bad_matrix;
    }
```

4.2.3.46 const AMatrix& AMatrix::operator *= (double *d*)

4.2.3.47 bool AMatrix::operator!= (const AMatrix & *m*) const [inline]

Definition at line 222 of file amatrix.inc.

```
{
        return Compare(m) != 0;
```

4.2.3.48 double AMatrix::operator() (int *r*) const [inline]

Definition at line 90 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (r < 1) || (r > Rows()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " ( " << r << " ) const" << endl;
#       endif
```

```
                return bad_val;
        }

```

4.2.3.49 double & AMatrix::operator() (int *r*) [inline]

Definition at line 105 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (r < 1) || (r > Rows()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " ( " << r << " ) " << endl;
#       endif
                return bad_val;
        }

```

4.2.3.50 double AMatrix::operator() (int *r*, int *c*) const [inline]

Definition at line 152 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (r < 1) || (c < 1) || (r > Rows()) || (c > Cols()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " ( " << r << ',' << c << " ) const" << endl;
#       endif
                return bad_val;
        }

```

4.2.3.51 double & AMatrix::operator() (int *r*, int *c*) [inline]

Definition at line 167 of file amatrix.inc.

Referenced by Compare().

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (r < 1) || (c < 1) || (r > Rows()) || (c > Cols()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " ( " << r << ',' << c << " ) " << endl;
#       endif
                return bad_val;
        }

```

4.2.3.52 AMatrix AMatrix::operator+ (const AMatrix & *m*) const

4.2.3.53 AMatrix AMatrix::operator+ (double *d*) const

4.2.3.54 const AMatrix& AMatrix::operator+= (const AMatrix & *m*)

4.2.3.55 const AMatrix& AMatrix::operator+= (double *d*)

4.2.3.56 AMatrix AMatrix::operator- (const AMatrix & *m*) const

4.2.3.57 AMatrix AMatrix::operator- (double *d*) const

4.2.3.58 const AMatrix& AMatrix::operator-= (const AMatrix & *m*)

4.2.3.59 const AMatrix& AMatrix::operator-= (double *d*)

4.2.3.60 AMatrix AMatrix::operator/ (const AMatrix & *m*) const `[inline]`

Definition at line 327 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if (Cols() != m.Rows())
    {
#       ifdef   DEBUG_MATRIX
        log << "Error: " << Rows() << 'x' << Cols()
            << " / " << m.Rows() << 'x' << m.Cols() << endl;
#       endif
        return bad_matrix;
    }

```

4.2.3.61 AMatrix AMatrix::operator/ (double *d*) const

4.2.3.62 const AMatrix & AMatrix::operator/= (const AMatrix & *m*) `[inline]`

Definition at line 343 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if (Cols() != m.Rows())
    {
#       ifdef   DEBUG_MATRIX
        log << "Error: " << Rows() << 'x' << Cols()
            << " /= " << m.Rows() << 'x' << m.Cols() << endl;
#       endif
        return bad_matrix;
    }

```

4.2.3.63 const AMatrix& AMatrix::operator/= (double *d*)

4.2.3.64 bool AMatrix::operator< (const AMatrix & *m*) const `[inline]`

Definition at line 232 of file amatrix.inc.

```
{
        return Compare(m) < 0;

```

4.2.3.65 bool AMatrix::operator<= (const AMatrix & *m*) const [inline]

Definition at line 227 of file amatrix.inc.

```
{
        return Compare(m) <= 0;

```

4.2.3.66 const AMatrix & AMatrix::operator= (const AMatrix & *m*) [inline]

Definition at line 72 of file amatrix.inc.

```
{
        if (the_rep == m.the_rep)
                return *this;
        the_rep->Dereference();
        the_rep = m.the_rep;
        the_rep->Reference();
        return *this;

```

4.2.3.67 bool AMatrix::operator== (const AMatrix & *m*) const [inline]

Definition at line 217 of file amatrix.inc.

```
{
        return Compare(m) == 0;

```

4.2.3.68 bool AMatrix::operator> (const AMatrix & *m*) const [inline]

Definition at line 242 of file amatrix.inc.

```
{
        return Compare(m) > 0;

```

4.2.3.69 bool AMatrix::operator>= (const AMatrix & *m*) const [inline]

Definition at line 237 of file amatrix.inc.

```
{
        return Compare(m) >= 0;

```

4.2.3.70 double AMatrix::operator[] (int *c*) const [inline]

Definition at line 121 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (c < 1) || (c > Cols()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " [ " << c << " ] const" << endl;
#       endif
                return bad_val;
        }

```

4.2.3.71 double & AMatrix::operator[] (int *c*) [inline]

Definition at line 136 of file amatrix.inc.

```
{
#ifndef NOERRORCHECKING_MATRIX
        if ( (c < 1) || (c > Cols()) )
        {
#       ifdef   DEBUG_MATRIX
                log << "Error: " << Rows() << 'x' << Cols()
                        << " [ " << c << " ] " << endl;
#       endif
                return bad_val;
        }

```

4.2.4 Friends And Related Function Documentation

4.2.4.1 istream& InputBinary (istream &, AMatrix &) [friend]

4.2.4.2 ostream& OutputBinary (ostream &, const AMatrix &) [friend]

4.2.4.3 AMatrix operator * (double *d*, const AMatrix & *m*) [friend]

4.2.4.4 AMatrix operator+ (double *d*, const AMatrix & *m*) [friend]

4.2.4.5 AMatrix operator- (double *d*, const AMatrix & *m*) [friend]

4.2.4.6 AMatrix operator/ (double *d*, const AMatrix & *m*) [friend]

4.2.4.7 ostream& operator<< (ostream & *os*, const AMatrix & *m*) [friend]

4.2.4.8 ostream& operator<= (ostream & *os*, const AMatrix & *m*) [friend]

Definition at line 404 of file amatrix.inc.

```
{ return OutputBinary(os,m); }

```

4.2.4.9 istream& operator>= (istream & *is*, AMatrix & *m*) [friend]

Definition at line 408 of file amatrix.inc.

```
00409 { return InputBinary(is,m); }
00410
```

4.2.4.10 istream& operator>> (istream & *is*, AMatrix & *m*) [friend]

4.2.5 Member Data Documentation

4.2.5.1 const AMatrix AMatrix::bad_matrix [static, protected]

Definition at line 110 of file amatrix.h.

4.2.5.2 double AMatrix::bad_val [static, protected]

Definition at line 111 of file amatrix.h.

4.2.5.3 double AMatrix::epsilon [static]

Definition at line 128 of file amatrix.h.

4.2.5.4 char * AMatrix::os_begin_row [static]

Definition at line 494 of file amatrix.h.

4.2.5.5 char * AMatrix::os_end_row [static]

Definition at line 495 of file amatrix.h.

4.2.5.6 const int AMatrix::product_threshold [static, protected]

Definition at line 108 of file amatrix.h.

4.2.5.7 AMatrixRep * AMatrix::the_rep [private]

Definition at line 102 of file amatrix.h.

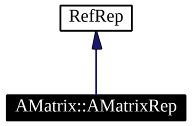
The documentation for this class was generated from the following files:

- **amatrix.h**
- **amatrix.inc**

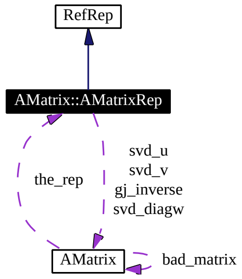
4.3 AMatrix::AMatrixRep Class Reference

`#include <amatrix.h>`

Inheritance diagram for AMatrix::AMatrixRep



Collaboration diagram for AMatrix::AMatrixRep:



Public Methods

- **AMatrixRep** ()
- **AMatrixRep** (const AMatrixRep &mr)
- **AMatrixRep** (int r, int c)
- double& **operator()** (int r, int c)

Protected Methods

- ∼**AMatrixRep** ()

Static Protected Methods

- double∗∗ **do_new** (int r, int c)
- void **do_delete** (double ∗∗**d**, int r, int c)

Private Attributes

- int **rows**
- int **cols**
- double∗∗ **d**
- bool **have_svd_flag**
- **AMatrix**∗ **svd_u**
- **AMatrix**∗ **svd_diagw**
- **AMatrix**∗ **svd_v**
- bool **have_inverse_flag**
- **AMatrix**∗ **gj_inverse**

Friends

- class **AMatrix**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 AMatrix::AMatrixRep::~AMatrixRep () `[protected]`

4.3.1.2 AMatrix::AMatrixRep::AMatrixRep ()

4.3.1.3 AMatrix::AMatrixRep::AMatrixRep (const AMatrixRep & *mr*)

4.3.1.4 AMatrix::AMatrixRep::AMatrixRep (int *r*, int *c*)

4.3.2 Member Function Documentation

4.3.2.1 void AMatrix::AMatrixRep::do_delete (double ** *d*, int *r*, int *c*) `[inline, static, protected]`

Definition at line 35 of file amatrix.inc.

```
00036 {
00037         for (int i=1; i<=r; ++i)
00038                 delete[] d[i];
00039         delete[] d;
00040
```

4.3.2.2 double ** AMatrix::AMatrixRep::do_new (int *r*, int *c*) `[inline, static, protected]`

Definition at line 19 of file amatrix.inc.

```
00020 {
00021         double**        d;
00022         d = new double*[r+1];
00023         if (!d)
00024                 return 0;
00025         for (int i=1; i<=r; ++i)
00026         {
00027                 d[i] = new double[c+1];
00028                 if (!d[i])
00029                         return 0;
00030         }
00031
```

4.3.2.3 double & AMatrix::AMatrixRep::operator() (int *r*, int *c*) `[inline]`

Definition at line 43 of file amatrix.inc.

```
00044 {
00045         return d[r][c];
00046
```

4.3.3 Friends And Related Function Documentation

4.3.3.1 class AMatrix [friend]

Definition at line 98 of file amatrix.h.

4.3.4 Member Data Documentation

4.3.4.1 int AMatrix::AMatrixRep::cols [private]

Definition at line 76 of file amatrix.h.

4.3.4.2 double ** AMatrix::AMatrixRep::d [private]

Definition at line 77 of file amatrix.h.

4.3.4.3 AMatrix * AMatrix::AMatrixRep::gj_inverse [private]

Definition at line 85 of file amatrix.h.

4.3.4.4 bool AMatrix::AMatrixRep::have_inverse_flag [private]

Definition at line 84 of file amatrix.h.

4.3.4.5 bool AMatrix::AMatrixRep::have_svd_flag [private]

Definition at line 79 of file amatrix.h.

4.3.4.6 int AMatrix::AMatrixRep::rows [private]

Definition at line 76 of file amatrix.h.

4.3.4.7 AMatrix * AMatrix::AMatrixRep::svd_diagw [private]

Definition at line 81 of file amatrix.h.

4.3.4.8 AMatrix * AMatrix::AMatrixRep::svd_u [private]

Definition at line 80 of file amatrix.h.

4.3.4.9 AMatrix * AMatrix::AMatrixRep::svd_v [private]

Definition at line 82 of file amatrix.h.

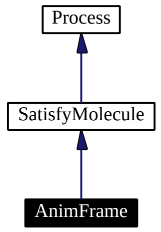
The documentation for this class was generated from the following files:

- **amatrix.h**
- **amatrix.inc**

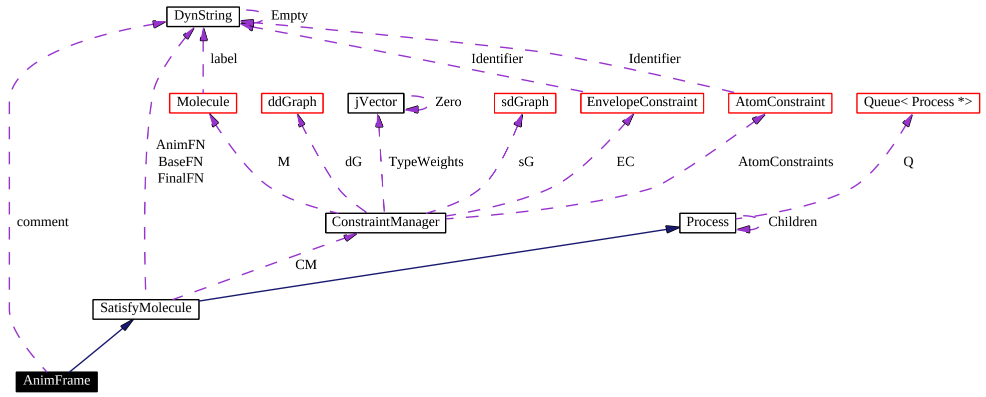
4.4 AnimFrame Class Reference

`#include <jd_anim_frame.h>`

Inheritance diagram for AnimFrame



Collaboration diagram for AnimFrame:



Public Methods

- **AnimFrame** (const AnimFrame &ca)
- **AnimFrame** (**ConstraintManager** &**CM**, **DynString** val="")
- ∼**AnimFrame** ()
- int **run** ()
- int **run** (ostream ∗os)
- void **Set_Output_Stream** (**DynString** val)

Public Attributes

- **DynString comment**

Static Public Attributes

- bool **Enabled** = false

Private Attributes

- bool **deleteStream**
- char* **name**
- int* **colors**
- ostream* **Output_Stream**

Static Private Attributes

- int **FrameNum** = 0

4.4.1 Detailed Description

///

File: **jd_anim_frame.h**

Created by: Jon Dugan

Writes a single animation frmae.

///

Definition at line 25 of file jd_anim_frame.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 AnimFrame::AnimFrame (const AnimFrame & *ca*)

Definition at line 5 of file jd_anim_frame.C.

```
  : SatisfyMolecule(*(ca.CM))
{
  // FIX fix copy constructor
  DIE("anim frame copy constructor called");
```

4.4.2.2 AnimFrame::AnimFrame (ConstraintManager & *cm_in*, DynString *val* = "")

Definition at line 14 of file jd_anim_frame.C.

```
  : SatisfyMolecule(cm_in)
{
  //  ostream *Output_Stream = &cout; // default
  Set_Output_Stream(val);
  int atoms =  cm_in.numAtoms();
  colors = new int[atoms];
  comment = "--";
```

4.4.2.3 AnimFrame::~AnimFrame ()

Definition at line 26 of file jd_anim_frame.C.

```
00026                     {
00027 
00028   delete [] colors;
00029   Output_Stream->flush();
00030 
00031   //  if (Output_Stream != &cerr && Output_Stream != &cout)
00032   if (deleteStream)
00033     delete Output_Stream;
00034 
00035 
```

4.4.3 Member Function Documentation

4.4.3.1 void AnimFrame::Set_Output_Stream (DynString *val*)

Definition at line 37 of file jd_anim_frame.C.

Referenced by AnimFrame().

```
00037                                                 {
00038   deleteStream = false;
00039   if (val == DynString("cerr") ||(val.Length() == 0))
00040     Output_Stream = &cerr;
00041   else if (val == DynString("cout"))
00042     Output_Stream = &cout;
00043   else {
00044     if (val[0] == '-'){
00045       cerr << "Confused by filename " << val << " . Try -help option." << endl;
00046       DIE("");
00047     }
00048     deleteStream = true;
00049     char * s = &(val[0]);  // ACK  not good, but it doesn't leak. -jd
00050     Output_Stream = new ofstream(s);
00051   }
00052 
```

4.4.3.2 int AnimFrame::run (ostream ∗ *os*)

4.4.3.3 int AnimFrame::run () [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 68 of file jd_anim_frame.C.

Referenced by jdRand_Eval::run(), jdIterator::run(), jdErr_Eval::run(), and AtomBuildup::run().

```
00068                   {
00069 #ifdef JOB_VERBOSE
00070   cerr << "Start: \tAnimFrame::run()" << endl;
00071 #endif
00072 
00073   if (AnimFrame::Enabled){
00074     jVector StressVals = CM->getStressList();
00075 
00076     for (int i =0; i < StressVals.Length(); i++){
00077       int color =  (int) StressVals[i];
00078       if (color > 9) color = 9;
00079       if (color < 0) color = 0;
00080       colors[i] = color;
```

```
    }
    AnimFrame::FrameNum++;
    CM->getMolecule().writeXmolData(*Output_Stream, colors, comment);
  }

```

4.4.4 Member Data Documentation

4.4.4.1 bool AnimFrame::Enabled = false [static]

Definition at line 37 of file globals.C.

4.4.4.2 int AnimFrame::FrameNum = 0 [static, private]

Definition at line 3 of file jd_anim_frame.C.

4.4.4.3 ostream * AnimFrame::Output_Stream [private]

Definition at line 59 of file jd_anim_frame.h.

4.4.4.4 int * AnimFrame::colors [private]

Definition at line 58 of file jd_anim_frame.h.

4.4.4.5 DynString AnimFrame::comment

Definition at line 46 of file jd_anim_frame.h.

4.4.4.6 bool AnimFrame::deleteStream [private]

Definition at line 55 of file jd_anim_frame.h.

4.4.4.7 char * AnimFrame::name [private]

Definition at line 56 of file jd_anim_frame.h.

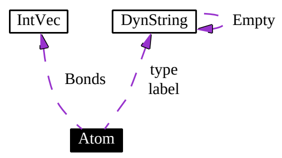
The documentation for this class was generated from the following files:

- **jd_anim_frame.h**
- **globals.C**
- **jd_anim_frame.C**

4.5 Atom Class Reference

`#include <atom.h>`

Collaboration diagram for Atom:



Public Methods

- void **AddBond** (int other_id)
- bool **Bonded** (int other_id) const
- const int& **ID** () const
- int& **ID** ()
- const **DynString**& **Type** () const
- **DynString**& **Type** ()
- const **DynString**& **Label** () const
- **DynString**& **Label** ()
- const double& **VDW** () const
- double& **VDW** ()
- const int& **ResNum** () const
- int& **ResNum** ()
- const double& **cov_dist** () const
- double& **cov_dist** ()
- const bool& **Moved** () const
- bool& **Moved** ()
- const **jVector**& **operator** ∗ () const
- **jVector**& **operator** ∗ ()
- int **Compare** (const Atom &a) const
- const Atom& **operator=** (const Atom &a)
- int **Output** (ostream &os) const
- int **Input** (istream &is)
- int **OutputBinary** (ostream &os) const
- int **InputBinary** (istream &is)
- **MAKECOMPARISONS_WITH_COMPARE** (Atom)
- **MAKE_IO_STREAM_OPERATORS** (Atom)
- **MAKE_BIN_STREAM_OPERATORS** (Atom)

Private Methods

- **Atom** (int identifier=UNINITIALIZED_ATOM_ID)
- **Atom** (const Atom &a)

Private Attributes

- **DynString label**
- **DynString type**
- int **id**
- double **VDW_Radius**
- double **cov_dist_Radius**
- bool **moved**
- int **residue_number**
- **IntVec Bonds**

Friends

- class **Molecule**
- class **Residue**

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Atom::Atom (int *identifier* = UNINITIALIZED_ATOM_ID) `[inline, private]`

Definition at line 100 of file atom.h.

```
00101 {
00102   id = identifier;
00103   type = "";
00104   label = "uninitialized atom";
00105   moved =  false;
00106   VDW_Radius = 0;
00107   cov_dist_Radius = 0;
00108   residue_number = -1;
00109   Bonds.Resize(0);
00110
00111
```

4.5.1.2 Atom::Atom (const Atom & *a*) `[inline, private]`

Definition at line 114 of file atom.h.

```
00115 {
00116   id = a.id;
00117   type = a.type;
00118   label = a.label;
00119   VDW_Radius = a.VDW_Radius;
00120   cov_dist_Radius = a.cov_dist_Radius;
00121   moved = a.moved;
00122   residue_number = a.residue_number;
00123   Bonds = a.Bonds;
00124
00125
```

4.5.2 Member Function Documentation

4.5.2.1 void Atom::AddBond (int *other_id*) [inline]

Definition at line 141 of file atom.h.

```
00141                                    {
00142   Bonds.AppendUnique(other_id);
00143
```

4.5.2.2 bool Atom::Bonded (int *other_id*) const [inline]

Definition at line 145 of file atom.h.

```
00145                                         {
00146   return Bonds.Contains(other_id);
00147
```

4.5.2.3 int Atom::Compare (const Atom & *a*) const [inline]

Definition at line 194 of file atom.h.

```
00195 { return (id-a.id); }
00196
```

4.5.2.4 int & Atom::ID () [inline]

Definition at line 151 of file atom.h.

```
00152 { return id; }
00153
```

4.5.2.5 const int & Atom::ID () const [inline]

Definition at line 149 of file atom.h.

Referenced by Molecule::Allocate(), Molecule::Molecule(), and AtomRegistry::SetType().

```
00150 { return id; }
00151
```

4.5.2.6 int Atom::Input (istream & *is*) [inline]

Definition at line 201 of file atom.h.

```
00202 { is >> id >> VDW_Radius >> cov_dist_Radius  >> moved ; return id; }
00203
```

4.5.2.7 int Atom::InputBinary (istream & *is*) `[inline]`

Definition at line 208 of file atom.h.

```
    { is >= id; is >= pos; return id; }

```

4.5.2.8 DynString & Atom::Label () `[inline]`

Definition at line 156 of file atom.h.

```
{ return label; }

```

4.5.2.9 const DynString & Atom::Label () const `[inline]`

Definition at line 154 of file atom.h.

Referenced by AtomRegistry::SetType().

```
{ return label; }

```

4.5.2.10 Atom::MAKECOMPARISONS_WITH_COMPARE (Atom)

4.5.2.11 Atom::MAKE_BIN_STREAM_OPERATORS (Atom)

4.5.2.12 Atom::MAKE_IO_STREAM_OPERATORS (Atom)

4.5.2.13 bool & Atom::Moved () `[inline]`

Definition at line 181 of file atom.h.

```
{ return moved; }

```

4.5.2.14 const bool & Atom::Moved () const `[inline]`

Definition at line 179 of file atom.h.

```
{ return moved; }

```

4.5.2.15 int Atom::Output (ostream & *os*) const `[inline]`

Definition at line 197 of file atom.h.

```
    { os << id << ' ' << pos; return id; }

```

4.5.2.16 int Atom::OutputBinary (ostream & *os*) const [inline]

Definition at line 204 of file atom.h.

```
00205     { os <= id; os <= pos; return id; }
00206
```

4.5.2.17 int & Atom::ResNum () [inline]

Definition at line 171 of file atom.h.

```
00172 {  return residue_number; }
00173
```

4.5.2.18 const int & Atom::ResNum () const [inline]

Definition at line 169 of file atom.h.

Referenced by AtomRegistry::SetType().

```
00170 {  return residue_number; }
00171
```

4.5.2.19 DynString & Atom::Type () [inline]

Definition at line 161 of file atom.h.

```
00162 { return type; }
00163
```

4.5.2.20 const DynString & Atom::Type () const [inline]

Definition at line 159 of file atom.h.

Referenced by AtomRegistry::SetType().

```
00160 { return type; }
00161
```

4.5.2.21 double & Atom::VDW () [inline]

Definition at line 166 of file atom.h.

```
00167 {  return VDW_Radius; }
00168
```

4.5.2.22 const double & Atom::VDW () const [inline]

Definition at line 164 of file atom.h.

Referenced by Envelope::Build(), AtomRegistry::SetType(), and Molecule::VDW().

```
00165 {  return VDW_Radius; }
00166
```

4.5.2.23 double & Atom::cov_dist () [inline]

Definition at line 176 of file atom.h.

```
00177 { return cov_dist_Radius; }
00178
```

4.5.2.24 const double & Atom::cov_dist () const [inline]

Definition at line 174 of file atom.h.

Referenced by AtomRegistry::SetType().

```
00175 { return cov_dist_Radius; }
00176
```

4.5.2.25 jVector& Atom::operator * ()

4.5.2.26 const jVector& Atom::operator * () const

4.5.2.27 const Atom & Atom::operator= (const Atom & *a*) [inline]

Definition at line 127 of file atom.h.

```
00127                                                   {
00128
00129   id = a.id;
00130   type = a.type;
00131   label = a.label;
00132   VDW_Radius = a.VDW_Radius;
00133   cov_dist_Radius = a.cov_dist_Radius;
00134   moved = a.moved;
00135   residue_number = a.residue_number;
00136   Bonds = a.Bonds;
00137
00138   return *this;
00139
```

4.5.3 Friends And Related Function Documentation

4.5.3.1 class Molecule [friend]

Definition at line 35 of file atom.h.

4.5.3.2 class Residue [friend]

Definition at line 36 of file atom.h.

4.5.4 Member Data Documentation

4.5.4.1 IntVec Atom::Bonds [private]

Definition at line 96 of file atom.h.

4.5.4.2 double Atom::VDW_Radius `[private]`

Definition at line 92 of file atom.h.

4.5.4.3 double Atom::cov_dist_Radius `[private]`

Definition at line 93 of file atom.h.

4.5.4.4 int Atom::id `[private]`

Definition at line 91 of file atom.h.

4.5.4.5 DynString Atom::label `[private]`

Definition at line 89 of file atom.h.

4.5.4.6 bool Atom::moved `[private]`

Definition at line 94 of file atom.h.

4.5.4.7 int Atom::residue_number `[private]`

Definition at line 95 of file atom.h.

4.5.4.8 DynString Atom::type `[private]`

Definition at line 90 of file atom.h.

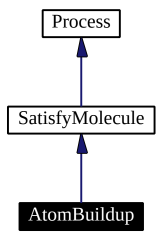
The documentation for this class was generated from the following file:

- **atom.h**

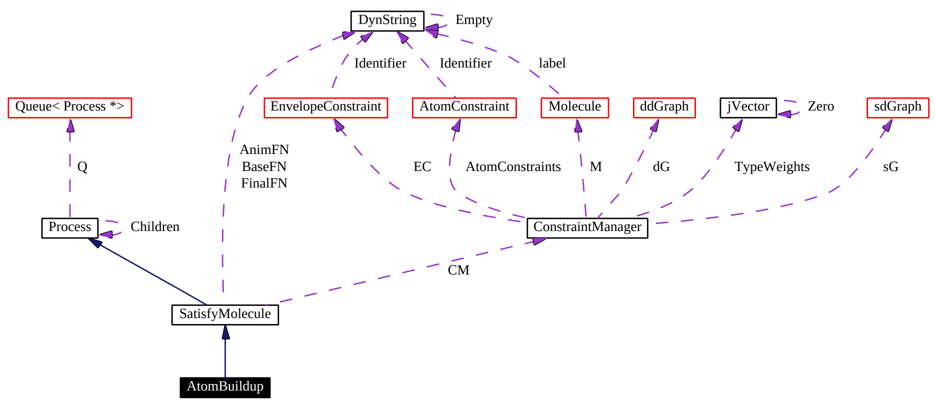
4.6 AtomBuildup Class Reference

`#include <atom_buildup.h>`

Inheritance diagram for AtomBuildup



Collaboration diagram for AtomBuildup:



Public Methods

- **AtomBuildup** (const AtomBuildup &ca)
- **AtomBuildup** (**ConstraintManager** &**CM**)
- ~**AtomBuildup** ()
- int **run** ()

Static Public Attributes

- int **MaxIterations** = 50
- int **MaxFinalIterations** = 1000

Private Attributes

- int **IterationNumber**

4.6.1 Detailed Description

//

File: **atom_buildup.h**

Created by: Jon Dugan

AtomBuildup job. iterates over all atoms based on error

//

Definition at line 28 of file atom_buildup.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 AtomBuildup::AtomBuildup (const AtomBuildup & *ca*)

Definition at line 3 of file atom_buildup.C.

```
  : SatisfyMolecule(*(ca.CM))
{
  DIE("AtomBuildup copy constructor called. ");

```

4.6.2.2 AtomBuildup::AtomBuildup (ConstraintManager & *cm*)

Definition at line 10 of file atom_buildup.C.

```
  : SatisfyMolecule(cm)
{


```

4.6.2.3 AtomBuildup::~AtomBuildup ()

Definition at line 16 of file atom_buildup.C.

```
  {
    // Don't delete my children (since Process takes care of it).
    // i.e. dont do for(int i=0; i<numChildren; ++i) delete
    // Children[i].  or delete[] Children.

    // NB in THIS class I am not adding my children to the queue for
    // the parent class, Procss'  As a result, the children need to be
    // explicitly deleted.

    delete Children[0];
    delete Children[1];
    //    delete Children[2];
    delete Children[3];
    // Normally a job would NOT do this --
    // ~Process would do it, but I am not adding these to the queue

```

4.6.3 Member Function Documentation

4.6.3.1 int AtomBuildup::run () [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 34 of file atom_buildup.C.

```
00034                      {
00035
00036 #ifdef JOB_VERBOSE
00037   cerr << "Start: \tAtom Buildup::run()" << endl;
00038 #endif
00039
00040   //CM->PrintGraph(cout);
00041
00042   int StopFlag = 0;
00043   int iteration = 0;
00044   int curr_atom = 0;
00045
00046   Children = new Process*[4];
00047
00048   Children[0] = new CycleAll(*CM);
00049   Children[1] = new AnimFrame(*CM, AnimFN);
00050   //  Children[2] = new StopEval(*CM, BaseFN);
00051   Children[3] = new RelaxSubset(*CM, .4);
00052
00053   StopEval * SE = new StopEval(*CM);
00054
00055   jVector JVdelta;
00056
00057   JVdelta[0] = 0.5;
00058   JVdelta[1] = 1.0;
00059   JVdelta[2] = 1.5;
00060
00061   jVector JVzero3(3,0);
00062
00063   jVector away(3,200);
00064   for ( int i = 0;i<CM->numAtoms();i++){
00065     CM->setPosition(i, away); // move them out of the way
00066   }
00067   CM->TurnAtomsOff();
00068   CM->TurnEnvelopeOn();
00069
00070   CM->TurnAtomOn(0);
00071   CM->setPosition(0, JVzero3); // put 0 at origin
00072
00073   curr_atom = 1;
00074   CM->TurnAtomOn(curr_atom);
00075   CM->setPosition(curr_atom, CM->getPosition(curr_atom-1)+JVdelta);
00076
00077   while (curr_atom<(CM->numAtoms()-1)) {
00078
00079 #ifdef JOB_VERBOSE
00080     cerr << "Info: \t\t\t\tAtom Buildup added atom " << curr_atom+1 << endl;
00081 #endif
00082
00083     iteration = StopFlag = 0;
00084     while (!StopFlag && iteration < MaxIterations){
00085
00086 #ifdef JOB_VERBOSE
00087     cerr << "Value: \tAtom Buildup iteration =" << iteration << endl;
00088 #endif
00089
00090     // CYCLE and RELAX
00091     StopFlag = Children[0]->run();
00092
00093
00094       //Children[3]->run();
```

```
      //Children[3]->run();


      // EVALUATE STOP CONDITION
//     if ((StopFlag = (StopFlag && SE->check_errors(1e-7, 1e-6)))){  // FIX TO BE PARAMS
//
// #ifdef JOB_VERBOSE2
//      cerr << "Value: \tAtom Buildup stopping on  SE->check_errors " << iteration << endl;
// #endif
//
//       }
      iteration++;

    }

    curr_atom++;
    CM->TurnAtomOn(curr_atom);
    CM->setPosition(curr_atom, CM->getPosition(curr_atom-1)+JVdelta); // put 0 at origin

    // WRITE ANIMATION FRAME
    Children[1]->run();


  }

  //  do final cycle
  //  CM->TurnAtomOn(curr_atom);
  //  CM->setPosition(curr_atom,  CM->getPosition(curr_atom-1)+JVdelta); // put 0 at origin

  iteration = StopFlag = 0;
  while (!StopFlag && iteration < MaxFinalIterations){
    Children[0]->run();
       //    StopFlag = (SE->check_stress_max_tot(1e-6, 1e-4)) || (Children[2]->run());
    StopFlag = SE->check_stress_max_tot(1e-6, 1e-4);
    SE->check_low_error(gwNumerics::MaxAbsDistRes, gwNumerics::AveAbsDistRes);

    Children[1]->run();
    iteration++;
    //       cerr << iteration << " ";
  }

```

4.6.4 Member Data Documentation

4.6.4.1 int AtomBuildup::IterationNumber [private]

Definition at line 46 of file atom_buildup.h.

4.6.4.2 int AtomBuildup::MaxFinalIterations = 1000 [static]

Definition at line 57 of file globals.C.

4.6.4.3 int AtomBuildup::MaxIterations = 50 [static]

Definition at line 56 of file globals.C.

The documentation for this class was generated from the following files:

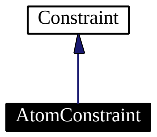
- **atom_buildup.h**

- **atom_buildup.C**
- **globals.C**

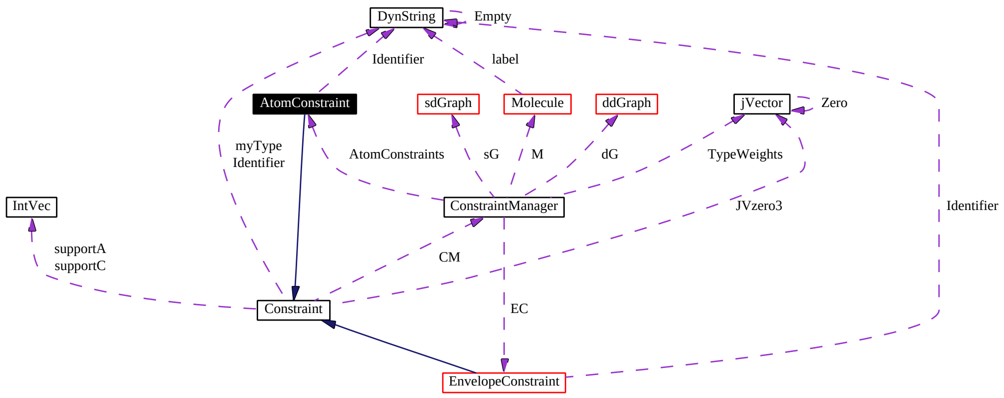
4.7 AtomConstraint Class Reference

`#include <atom_constraint.h>`

Inheritance diagram for AtomConstraint



Collaboration diagram for AtomConstraint:



Public Methods

- **AtomConstraint** (**ConstraintManager** &cm)
- **AtomConstraint** (**ConstraintManager** &cm, int atom_i)
- **~AtomConstraint** ()
- double **computeStress** () const
- double **computeStress** (int k, const **jVector** &pos) const
- **jVector dStress** (int i, const **jVector** &pos_k) const
- int **Compare** (const AtomConstraint &dc) const
- **MAKECOMPARISONS_WITH_COMPARE** (AtomConstraint)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const

Static Public Attributes

- **DynString Identifier** = "atm:"

Protected Methods

- int **Read** (istream &is)
- int **ReadBinary** (istream &is)

    Same scenerio as **Read**() (p. 57) *only binary version.*

- void **notifyValid** (**Constraint** ∗child, bool validity=true)
- void **notifyRemove** (void)

    I'm about to be removed from the constraint graph.

4.7.1 Detailed Description

//

atom_constraint.h

This constraint class is really a wrapper class for the atom class so that it can be massaged into a **ConstraintManager** (p. 97)'s constraint graphs.

//

Definition at line 21 of file atom_constraint.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 AtomConstraint::AtomConstraint (ConstraintManager & *cm*)

Definition at line 3 of file atom_constraint.C.

```
00004   : Constraint(cm,IntVec(0))
00005   {
00006
00007
```

4.7.2.2 AtomConstraint::AtomConstraint (ConstraintManager & *cm*, int *i*)

Definition at line 9 of file atom_constraint.C.

```
00010   : Constraint(cm, IntVec(0))
00011   {
00012     myID = i;
00013
```

4.7.2.3 AtomConstraint::∼AtomConstraint () `[inline]`

Definition at line 29 of file atom_constraint.h.

```
00029                   {}
00030
```

4.7.3 Member Function Documentation

4.7.3.1 int AtomConstraint::Compare (const AtomConstraint & *ac*) const `[inline]`

Definition at line 55 of file atom_constraint.h.

```
00056 {
00057
00058   if (myID == ac.ID()) return 0;
00059   if (myID < ac.ID()) return -1;
00060   return 1;
00061
00062   //
00063   //  if(atomIndex[0] == dc.atomIndex[0] && atomIndex[1] == dc.atomIndex[1])
00064   //    return 0;
00065   //  if(atomIndex[0]<dc.atomIndex[0])
00066   //    return -1;
00067   //  if(atomIndex[0]>dc.atomIndex[0])
00068   //    return 1;
00069   //  if(atomIndex[1]<dc.atomIndex[1])
00070   //    return -1;
00071   //  return 1;
00072
```

4.7.3.2 AtomConstraint::MAKECOMPARISONS_WITH_COMPARE (AtomConstraint)

4.7.3.3 int AtomConstraint::Output (ostream & *os*) const `[virtual]`

Reimplemented from **Constraint** (p. 84).

Definition at line 75 of file atom_constraint.C.

```
00076 {
00077   os << myID;
00078   return 1;
00079
```

4.7.3.4 int AtomConstraint::OutputBinary (ostream & *os*) const `[virtual]`

Reimplemented from **Constraint** (p. 84).

Definition at line 87 of file atom_constraint.C.

```
00088 {
00089   os <= myID;
00090   return 1;
00091
```

4.7.3.5 int AtomConstraint::Read (istream & *is*) `[protected, virtual]`

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 57) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 57) function, the derived class should *not* log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented from **Constraint** (p. 85).

Definition at line 81 of file atom_constraint.C.

```
00082 {
```

```
00083   is >> myID;
00084   return 1;
00085
```

4.7.3.6 int AtomConstraint::ReadBinary (istream & *is*) [protected, virtual]

Same scenerio as **Read**() (p. 57) only binary version.

Reimplemented from **Constraint** (p. 85).

Definition at line 106 of file atom_constraint.C.

```
00107 {
00108   is >= myID;
00109   return 1;
00110
```

4.7.3.7 double AtomConstraint::computeStress (int *k*, const jVector & *pos_k*) const [virtual]

Returns the stress on this constraint after "pretending" the atom with index a' moved to position pos_a'. This also must be overwitten by all constraints that inherit from this class

Reimplemented from **Constraint** (p. 89).

Definition at line 42 of file atom_constraint.C.

```
00043 {
00044   if (myID != k)
00045     DIE("Tried to computeStress' on atom constraint for an atom other than the one pointed to by the constraint");
00046   double returnVal = 0;
00047   int numCyc = supportC.Length();
00048   for (int i =0; i < numCyc; i++)
00049     //min variance goes here
00050     returnVal += CM->getConstraint(supportC[i])->stress(myID,pos_k);
00051
00052   return returnVal;
00053
```

4.7.3.8 double AtomConstraint::computeStress () const [virtual]

One of the only functions that *must* be overwritten by subclasses. Returns a value reflecting how dissatisfied this constraint is with the current molecule residing in CM'.

Reimplemented from **Constraint** (p. 89).

Definition at line 20 of file atom_constraint.C.

```
00021 {
00022   // Return the sum of stress of constraints that are "on" me.
00023   // We're not sure we want to do it like this yet.  It violates
00024   // the DAG semantics of constraint use in the constraint graph.
00025   // I.e. Constraints are parents of the constraints whose values they
00026   // use to obtain their stress() value.
00027   //
00028   double returnVal = 0;
00029   int numCyc = supportC.Length();
00030
```

```
  for (int i =0; i < numCyc; i++){
    returnVal += CM->getConstraint(supportC[i])->stress();
//      cerr << "I'm atom_constraint " << myID << ", Added stress "
//        << CM->getConstraint(supportC[i])->stress() << " from "
//        << supportC[i] <<"("<< CM->getConstraint(supportC[i])->Label()
//        << "), total = " << returnVal << endl;
  }

```

4.7.3.9 jVector AtomConstraint::dStress (int *k*, const jVector & *pos_k*) const [virtual]

asking constraint number k for its 12 element dStress vector we pass in pos_a vector – corresponding to the atom position if the constraint is an atom, and corresponding to the position of the atom k in question if the call is to a non-atom constraint. EG for a distance constraint atom 12 needs the dStress from the d-constraint, so it will call dStress on the d-constraint(12, atom_pos) so the d_constr can answer the question if atom 12 were in atom_pos, I would say my dStress is blah.

Reimplemented from **Constraint** (p. 89).

Definition at line 55 of file atom_constraint.C.

```
{
  if (myID != k)
    DIE("Tried to call dStress on atom constraint for an atom other than the one pointed to by its constraint");
  //  jVector ReturnVal = jVector(12,0);
  jVector ReturnVal = jVector(3,0);


  int numCyc = supportC.Length();

  for (int i =0; i < numCyc; i++)
    //min variance goes here
    if (CM->constraintON(supportC[i])){
      //      cerr << "adding to dstress from constraint " << supportC[i] << endl;
      ReturnVal += CM->getConstraint(supportC[i])->dStress(myID,pos_k);
    }

```

4.7.3.10 void AtomConstraint::notifyRemove (void) [protected, virtual]

I'm about to be removed from the constraint graph.

Reimplemented from **Constraint** (p. 91).

Definition at line 113 of file atom_constraint.C.

```
  {
    DIE(" Atoms cannot be removed.");

```

4.7.3.11 void AtomConstraint::notifyValid (Constraint ∗ *child*, bool *validity* = true) [protected, virtual]

Informs me that one of my supports now contains valid/invalid information depending on the validity' flag.

Reimplemented from **Constraint** (p. 91).

Definition at line 93 of file atom_constraint.C.

```
00094 {
00095
00096   // this is used to tell the atom that a constraint on which it
00097   // depends (supportC -- up in the graph) has become invalid.
00098
00099   if (validity){
00100     DIE(" Called notifyValid(true) on an Atom constraint.");
00101   }
00102
```

4.7.4 Member Data Documentation

4.7.4.1 DynString AtomConstraint::Identifier = "atm:" [static]

Reimplemented from **Constraint** (p. 95).

Definition at line 94 of file globals.C.

The documentation for this class was generated from the following files:

- **atom_constraint.h**
- **atom_constraint.C**
- **globals.C**

4.8 AtomRegistry Class Reference

`#include <atom_registry.h>`

Static Public Methods

- int **SetType** (**Atom** &a, const **DynString** &in_type, **DynString** &in_label, int resum=-2)

Static Public Attributes

- const char* **names** [num_known_atom_species]

  names of the atom types.

- const double **covalent_distance** [num_known_atom_species]

  min separation distance to covalently bonded atom.

- const double **radius** [num_known_atom_species]

  VDW radius.

4.8.1 Member Function Documentation

4.8.1.1 int AtomRegistry::SetType (Atom & *a*, const DynString & *in_type*, DynString & *in_label*, int *resnum* = -2) `[inline, static]`

Definition at line 54 of file atom_registry.h.

Referenced by Residue::AddAtom(), and Molecule::readXmol().

```
                                                                                                {
  for (int i = 0; i < num_known_atom_species; i++){
    if (in_type == names[i]){
      if (a.Type() == in_type) {return a.ID();}
      if (a.Type().Length() > 0) {cerr <<"|"<< a.Type() <<"|" <<in_type <<"|" << endl; WARN ("atom::SetType failed");}
      // a.type already set differently
      a.Type() = in_type;
      a.Label() = in_label;
      a.VDW() = radius[i];
      a.cov_dist() = covalent_distance[i];
      a.ResNum() = resnum;
      return a.ID();
    }
  }

```

4.8.2 Member Data Documentation

4.8.2.1 const double AtomRegistry::covalent_distance[num_known_atom_species] `[static]`

Initial value:

```
{
 0.68, 0.51, 0.47, 0.91, 0.91
}
```

min separation distance to covalently bonded atom.

Definition at line 148 of file globals.C.

4.8.2.2 const char * AtomRegistry::names[num_known_atom_species] [static]

Initial value:

```
{
 "C","N","O","P","S"
}
```

names of the atom types.

Definition at line 139 of file globals.C.

4.8.2.3 const double AtomRegistry::radius[num_known_atom_species] [static]

Initial value:

```
{
 1.18,1.1,1,1.25,1.31
}
```

VDW radius.

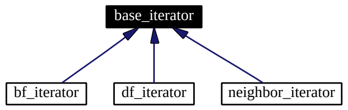
Definition at line 144 of file globals.C.

The documentation for this class was generated from the following files:

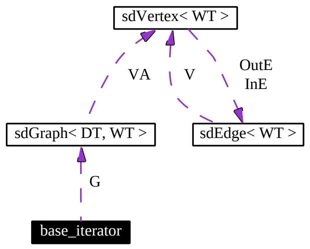
- **atom_registry.h**
- **globals.C**

4.9 base_iterator Class Reference

Inheritance diagram for base_iterator



Collaboration diagram for base_iterator:



Public Methods

- **base_iterator** (int reverse=0)
- **base_iterator** (const base_iterator &bi)
- void **Reverse** ()
- const base_iterator& **operator=** (const base_iterator &bi)
- int **bad** () const
- virtual **~base_iterator** ()
- virtual void **reset** ()
- virtual **operator int** () const
- virtual const base_iterator& **operator++** ()
- virtual int **operator** ∗ () const

Protected Methods

- int **Invalid** (int vndx) const
- **base_iterator** (**sdGraph**< DT, WT > ∗g, int vndx, int reverse=0)

Protected Attributes

- **sdGraph**<DT,WT>∗ **G**
- int **vstart**
- int **vid**
- int **rev**

Friends

- class **sdGraph**< **DT**, **WT** >

4.9.1 Constructor & Destructor Documentation

4.9.1.1 base_iterator::base_iterator (sdGraph< DT, WT > * *g*, int *vndx*, int *reverse* = 0) `[inline, protected]`

Definition at line 23 of file sdgiter.inc.

```
00024                 {
00025                   G=g;
00026                   vstart = Invalid(vndx) ? -1 : vndx;
00027                   rev=reverse;
00028                   reset();
00029
```

4.9.1.2 base_iterator::base_iterator (int *reverse* = 0) `[inline]`

Definition at line 31 of file sdgiter.inc.

```
00032                 { G=0; vstart=-1; vid=-1; rev=reverse; }
00033
```

4.9.1.3 base_iterator::base_iterator (const base_iterator & *bi*) `[inline]`

Definition at line 33 of file sdgiter.inc.

```
00034                 { G=bi.G; vstart=bi.vstart; vid=bi.vid; rev=bi.rev; }
00035
```

4.9.1.4 base_iterator::~base_iterator () `[inline, virtual]`

Definition at line 46 of file sdgiter.inc.

```
00046                                         { /*empty*/ }
00047
```

4.9.2 Member Function Documentation

4.9.2.1 int base_iterator::Invalid (int *vndx*) const `[inline, protected]`

Definition at line 20 of file sdgiter.inc.

Referenced by base_iterator().

```
00021                 { return (!G || (vndx<0) || (vndx>=G->NumV)) ? 1 : 0; }
00022
```

4.9.2.2 void base_iterator::Reverse () `[inline]`

Definition at line 35 of file sdgiter.inc.

```
00035                           { rev = rev ? 0 : 1; }
00036
```

4.9.2.3 int base_iterator::bad () const [inline]

Definition at line 43 of file sdgiter.inc.

Referenced by path_iterator::operator++(), bf_iterator::reset(), df_iterator::reset(), and neighbor_-iterator::reset().

```
00044                  { return (!G || vstart<0); }
00045
```

4.9.2.4 int base_iterator::operator * () const [inline, virtual]

Reimplemented in **neighbor_iterator** (p. 375), **df_iterator** (p. 151), and **bf_iterator** (p. 68).

Definition at line 50 of file sdgiter.inc.

```
00050                                             { return vid; }
00051
```

4.9.2.5 base_iterator::operator int () const [inline, virtual]

Reimplemented in **neighbor_iterator** (p. 375), **df_iterator** (p. 152), and **bf_iterator** (p. 69).

Definition at line 48 of file sdgiter.inc.

```
00048                                            { return (vid!=-1); }
00049
```

4.9.2.6 const base_iterator & base_iterator::operator++ () [inline, virtual]

Reimplemented in **neighbor_iterator** (p. 375), **df_iterator** (p. 152), and **bf_iterator** (p. 69).

Definition at line 49 of file sdgiter.inc.

```
00049                                                       { return *this; }
00050
```

4.9.2.7 const base_iterator & base_iterator::operator= (const base_iterator & *bi*) [inline]

Definition at line 36 of file sdgiter.inc.

```
00037                  {
00038                    if(this!=&bi)
00039                      { G=bi.G; vstart=bi.vstart; vid=bi.vid; rev=bi.rev; }
00040
```

4.9.2.8 void base_iterator::reset () [inline, virtual]

Reimplemented in **neighbor_iterator** (p. 376), **df_iterator** (p. 153), and **bf_iterator** (p. 70).

Definition at line 47 of file sdgiter.inc.

Referenced by base_iterator(), bf_iterator::bf_iterator(), df_iterator::df_iterator(), and neighbor_-iterator::neighbor_iterator().

```
                                        { vid = vstart; }

```

4.9.3 Friends And Related Function Documentation

4.9.3.1 class sdGraph [friend]

Reimplemented in **neighbor_iterator** (p. 376), **df_iterator** (p. 153), and **bf_iterator** (p. 70).

Definition at line 13 of file sdgiter.inc.

4.9.4 Member Data Documentation

4.9.4.1 sdGraph< DT, WT > * base_iterator::G [protected]

Definition at line 16 of file sdgiter.inc.

4.9.4.2 int base_iterator::rev [protected]

Definition at line 19 of file sdgiter.inc.

4.9.4.3 int base_iterator::vid [protected]

Definition at line 18 of file sdgiter.inc.

4.9.4.4 int base_iterator::vstart [protected]

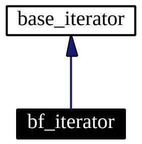
Definition at line 17 of file sdgiter.inc.

The documentation for this class was generated from the following file:

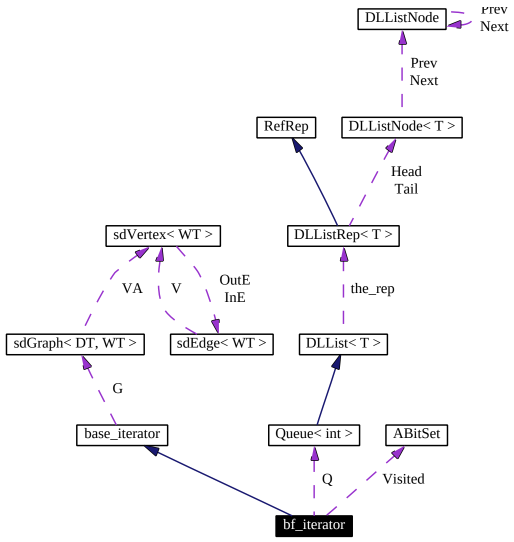
- **sdgiter.inc**

4.10 bf_iterator Class Reference

Inheritance diagram for bf_iterator



Collaboration diagram for bf_iterator:



Public Methods

- **bf_iterator** (int reverse=0)
- **bf_iterator** (const bf_iterator &bfi)
- **~bf_iterator** ()
- const bf_iterator& **operator=** (const bf_iterator &bfi)
- void **reset** ()
- **operator int** () const
- const **base_iterator**& **operator++** ()
- const **base_iterator**& **operator–** ()
- int **operator** ∗ () const

Protected Methods

- **bf_iterator** (**sdGraph**< DT, WT > ∗g, int vndx, int reverse=0)

Private Attributes

- **ABS∗ Visited**
- **Queue**<int> **Q**

Friends

- class **sdGraph< DT, WT >**

4.10.1 Constructor & Destructor Documentation

4.10.1.1 bf_iterator::bf_iterator (sdGraph< DT, WT > * *g*, int *vndx*, int *reverse* = 0) `[inline, protected]`

Definition at line 220 of file sdgiter.inc.

```
00221                    : base_iterator(g,vndx,reverse)
00222
```

4.10.1.2 bf_iterator::bf_iterator (int *reverse* = 0) `[inline]`

Definition at line 225 of file sdgiter.inc.

```
00225                          : base_iterator(reverse)
00226
```

4.10.1.3 bf_iterator::bf_iterator (const bf_iterator & *bfi*) `[inline]`

Definition at line 229 of file sdgiter.inc.

```
00229                               : base_iterator(bfi)
00230     {
00231       Visited = new ABS(*(bfi.Visited));
00232       Q = bfi.Q;
00233
```

4.10.1.4 bf_iterator::~bf_iterator () `[inline]`

Definition at line 235 of file sdgiter.inc.

```
00236     { delete Visited; }
00237
```

4.10.2 Member Function Documentation

4.10.2.1 int bf_iterator::operator * () const `[inline, virtual]`

Reimplemented from **base_iterator** (p. 65).

Definition at line 311 of file sdgiter.inc.

```
00311                     { return vid; }
00312
```

4.10.2.2 bf_iterator::operator int () const [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 262 of file sdgiter.inc.

```
                    { return (vid!=-1); }

```

4.10.2.3 const base_iterator & bf_iterator::operator++ () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 265 of file sdgiter.inc.

```
                    {
                      int visited, nbrid;

                      // We can stop if we've visited all the vertices:
                      if(Visited->IsAllSet()) vid = -1;

                      if(vid==-1) return *this;

                      const sdEdge<WT>* edges = rev ? G->VA[vid]->InE :
                                                      G->VA[vid]->OutE;
                      int n = rev ? G->VA[vid]->NumI : G->VA[vid]->NumO;

                      // Tack all the unvisited neighbors onto the queue:
                      for(int i=0; i<n; ++i)
                        {
                          nbrid = edges[i].V->ID;
                          if(!Visited->IsSet(nbrid))
                              Q.Add(nbrid);
                        }

                      // Skip all the visited vertices in the queue and
                      // assign 'vid' to the first unvisited vertex:
                      vid = -1;
                      visited = 1;
                      while(!Q.IsEmpty() && visited)
                        {
                          vid = Q.Remove();
                          if((visited = Visited->IsSet((uint) vid)))
                              vid = -1;
                        }

```

4.10.2.4 const base_iterator & bf_iterator::operator– () [inline]

Definition at line 303 of file sdgiter.inc.

```
                    {
                      rev = rev ? 0 : 1;
                      ++($\ast$this);
                      rev = rev ? 0 : 1;
                      return *this;

```

4.10.2.5 const bf_iterator & bf_iterator::operator= (const bf_iterator & *bfi*) [inline]

Definition at line 238 of file sdgiter.inc.

```
{
  if(this!=&bfi)
    {
      G=bfi.G; vstart=bfi.vstart; vid=bfi.vid; rev=bfi.rev;
      Q = bfi.Q;
      delete Visited;
      Visited = new ABS(*(bfi.Visited));
    }
```

4.10.2.6 void bf_iterator::reset () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 250 of file sdgiter.inc.

```
{
  if(bad()) vid=-1;
  else
    {
      Q.Clear();
      vid = vstart;
      Visited->ClearAll();
      Visited->Set((uint) vid);
    }
```

4.10.3 Friends And Related Function Documentation

4.10.3.1 class sdGraph [friend]

Reimplemented from **base_iterator** (p. 66).

Definition at line 217 of file sdgiter.inc.

4.10.4 Member Data Documentation

4.10.4.1 Queue< int > bf_iterator::Q [private]

Definition at line 317 of file sdgiter.inc.

4.10.4.2 ABS * bf_iterator::Visited [private]

Definition at line 314 of file sdgiter.inc.

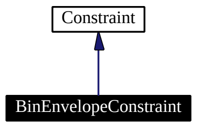
The documentation for this class was generated from the following file:

- **sdgiter.inc**

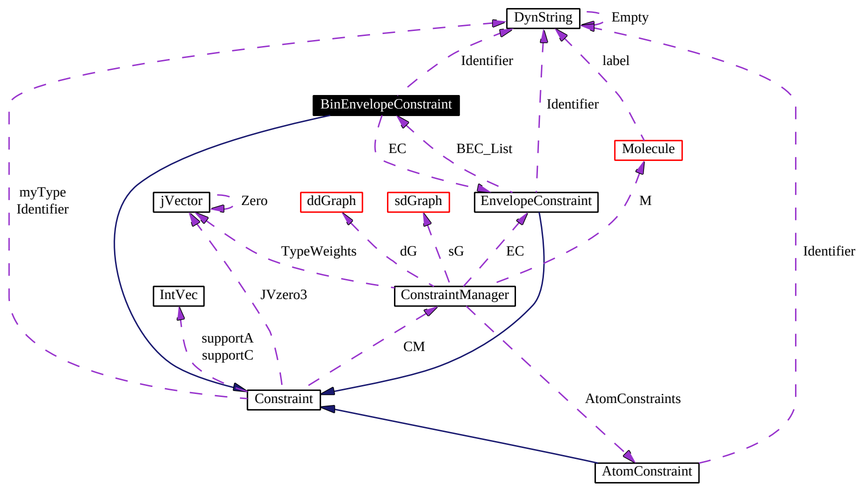
4.11 BinEnvelopeConstraint Class Reference

`#include <bin_envelope_constraint.h>`

Inheritance diagram for BinEnvelopeConstraint



Collaboration diagram for BinEnvelopeConstraint:



Public Methods

- **BinEnvelopeConstraint** (**ConstraintManager** &cm, int atom, **EnvelopeConstraint** *envelope_in)

  Constructor: atom is constraint number for atom, same for envelope.

- **~BinEnvelopeConstraint** ()
- **jVector dStress** (int k, const **jVector** &pos) const

  Returns the (3x1)– actually a 12x1 vector of derivatives of residual with respect to each dimension atom k at position pos.

- int **Compare** (const BinEnvelopeConstraint &ec) const
- **MAKECOMPARISONS_WITH_COMPARE** (BinEnvelopeConstraint)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const

Static Public Attributes

- **DynString Identifier** = "b_env:"

Protected Methods

- double **computeStress** () const

    returns stress on constraint.

- double **computeStress** (int k, const **jVector** &pos) const

    returns stress on constraint if atom k' were in position pos'.

- int **Read** (istream &is)
- int **ReadBinary** (istream &is)

    Same scenerio as **Read***()* (p. 74) *only binary version.*

- void **PreSetup** ()
- void **PostSetup** ()

Protected Attributes

- **EnvelopeConstraint**∗ **EC**

    passed in – pointing to the parent envelope constraint.

- int **EC_id**
- int **Atom_id**

Friends

- class **EnvelopeConstraint**

4.11.1 Detailed Description

//

bin_envelope_constraint.h

This is a constraint type for GRAPE. It constrains the envelope around a molecule. This is the sub-constraint type which is spawned by **EnvelopeConstraint** (p. 228).

//

Definition at line 23 of file bin_envelope_constraint.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 BinEnvelopeConstraint::BinEnvelopeConstraint (ConstraintManager & *cm*, int *atom_in*, EnvelopeConstraint ∗ *envelope_in*)

Constructor: atom is constraint number for atom, same for envelope.

Definition at line 5 of file bin_envelope_constraint.C.

```
  : Constraint(cm, IntVec(1,atom_in))  // supported by 1 atom
{

  EC = envelope_in;
```

```
00011   EC_id = EC->ID();
00012   Atom_id = atom_in;
00013
00014
```

4.11.2.2 BinEnvelopeConstraint::~BinEnvelopeConstraint ()

Definition at line 16 of file bin_envelope_constraint.C.

```
00016                                              {
00017
00018
```

4.11.3 Member Function Documentation

4.11.3.1 int BinEnvelopeConstraint::Compare (const BinEnvelopeConstraint & *ec*) const [inline]

Definition at line 70 of file bin_envelope_constraint.h.

```
00071 {
00072   DIE("b_env Compare not Implemented");
00073
00074   return 0;
00075
```

4.11.3.2 BinEnvelopeConstraint::MAKECOMPARISONS_WITH_COMPARE (BinEnvelopeConstraint)

4.11.3.3 int BinEnvelopeConstraint::Output (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 124 of file bin_envelope_constraint.C.

```
00125 {
00126   return 1;
00127
00128
```

4.11.3.4 int BinEnvelopeConstraint::OutputBinary (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 136 of file bin_envelope_constraint.C.

```
00137 {
00138   //  os <= supportA <= measured <= variance;
00139   return 1;
00140
```

4.11.3.5 void BinEnvelopeConstraint::PostSetup (void) `[protected, virtual]`

constraints get their ID when added to the constraint graph. this addition is done in the Input function between the pre- and post setup functions. constraint setup that needs to occur with an ID and an added constraint go in post setup. (like adding parent constraints, etc.)

Reimplemented from **Constraint** (p. 84).

Definition at line 32 of file bin_envelope_constraint.C.

```
00032                                    {
00033
00034
00035
00036
```

4.11.3.6 void BinEnvelopeConstraint::PreSetup (void) `[protected, virtual]`

The setup function is called if the uninialized constraint method is used to create the constraint to allow different data representations within the constraint. Without this, only data read in from the input file will be available to the member functions. For simple constraints, this is really not very useful. However for multi-atom constraints that need to be a little 'smarter' about data representations, the Setup function will be helpful.

Reimplemented from **Constraint** (p. 85).

Definition at line 21 of file bin_envelope_constraint.C.

```
00021                                   {
00022   myType = BinEnvelopeConstraint::Identifier; // needs to be done here -- not read in by registry
00023   myIntType = ConstraintRegistry::getIntType(myType);
00024
00025   addConstraintSupport(EC_id);
00026   // Valid Envelope constraint is a support
00027   // here valid means atoms are rotated correctly
00028
00029
00030
```

4.11.3.7 int BinEnvelopeConstraint::Read (istream & *is*) `[protected, virtual]`

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 74) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 74) function, the derived class should *not* log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented from **Constraint** (p. 85).

Definition at line 130 of file bin_envelope_constraint.C.

```
00131 {
00132   // This called on the input stream from envelope_constraint
00133   return 1;
00134
```

4.11.3.8 int BinEnvelopeConstraint::ReadBinary (istream & *is*) [protected, virtual]

Same scenerio as **Read**() (p. 74) only binary version.

Reimplemented from **Constraint** (p. 85).

Definition at line 142 of file bin_envelope_constraint.C.

```
00143 {
00144   // This called on the input stream from envelope_constraint
00145   return 1;
00146
```

4.11.3.9 double BinEnvelopeConstraint::computeStress (int *k*, const jVector & *pos_k*) const [protected, virtual]

returns stress on constraint if atom k' were in position pos'.

Reimplemented from **Constraint** (p. 89).

Definition at line 76 of file bin_envelope_constraint.C.

```
00077 {
00078
00079   return util_max(0.,(MAX_ENV_DEN-(EC->getDensity(pos_k)))/ENV_DEN_RANGE);
00080   DIE("");
00081
```

4.11.3.10 double BinEnvelopeConstraint::computeStress () const [protected, virtual]

returns stress on constraint.

Reimplemented from **Constraint** (p. 89).

Definition at line 40 of file bin_envelope_constraint.C.

```
00040                                            {
00041
00042   double rv = util_max(0.,(MAX_ENV_DEN-(EC->getDensity(getMolecule()(supportA[0]))))/ENV_DEN_RANGE);
00043 //   cerr << "BEC " << myID << ", atom pos = " << getMolecule()(supportA[0]) << endl;
00044 //   cerr << "Calculated BEC stress = " << rv << " atom= " << Atom_id << endl;
00045
00046
00047 //   int atom = supportA[0];
00048 //   jVector pos = getMolecule()(atom);
00049 //   double den = EC->getDensity(pos);
00050 //   double diff = MAX_ENV_DEN-den;
00051 //   double scale = diff/ENV_DEN_RANGE;
00052
00053
00054
00055
00056
00057   return rv;
00058
00059
00063
```

4.11.3.11 jVector BinEnvelopeConstraint::dStress (int *k*, const jVector & *pos_k*) const [virtual]

Returns the (3x1)– actually a 12x1 vector of derivatives of residual with respect to each dimension atom k at position pos.

Reimplemented from **Constraint** (p. 89).

Definition at line 84 of file bin_envelope_constraint.C.

```
{


 if(k != supportA[0])
   // trying to act on atom other than the one this constraint applies to
   {
     cerr << "Fatal error: bin_envelope_constraint.C, dStress("<< k \
          <<", pos) not legal " << endl;
     exit(1);
   }

```

4.11.4 Friends And Related Function Documentation

4.11.4.1 class EnvelopeConstraint [friend]

Definition at line 25 of file bin_envelope_constraint.h.

4.11.5 Member Data Documentation

4.11.5.1 int BinEnvelopeConstraint::Atom_id [protected]

Definition at line 50 of file bin_envelope_constraint.h.

4.11.5.2 EnvelopeConstraint ∗ BinEnvelopeConstraint::EC [protected]

passed in – pointing to the parent envelope constraint.

Definition at line 48 of file bin_envelope_constraint.h.

4.11.5.3 int BinEnvelopeConstraint::EC_id [protected]

Definition at line 49 of file bin_envelope_constraint.h.

4.11.5.4 DynString BinEnvelopeConstraint::Identifier = "b_env:" [static]

Reimplemented from **Constraint** (p. 95).

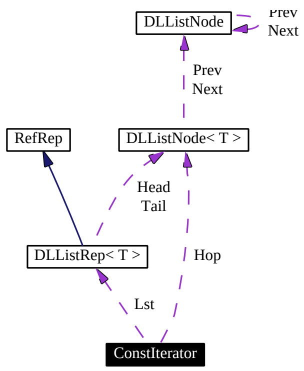
Definition at line 112 of file globals.C.

The documentation for this class was generated from the following files:

- **bin_envelope_constraint.h**
- **bin_envelope_constraint.C**
- **globals.C**

4.12 ConstIterator Class Reference

Collaboration diagram for ConstIterator:



Public Methods

- void **Reset** ()
- const T& **operator** ∗ ()
- **operator int** () const
- const ConstIterator& **operator++** ()
- void **Copy** (const ConstIterator &it)
- const ConstIterator& **operator=** (const ConstIterator &it)
- **ConstIterator** ()
- **ConstIterator** (const ConstIterator &it)
- **∼ConstIterator** ()

Private Methods

- **ConstIterator** (const **DLList**< T > &dll)

Private Attributes

- **DLListRep**<T>∗ **Lst**
- const **DLListNode**<T>∗ **Hop**

Friends

- class **DLList**< **T** >

4.12.1 Constructor & Destructor Documentation

4.12.1.1 ConstIterator::ConstIterator () `[inline]`

Definition at line 69 of file dlliter.inc.

```
                              { Lst = 0; Hop = 0; }

```

4.12.1.2 ConstIterator::ConstIterator (const ConstIterator & *it*) `[inline]`

Definition at line 70 of file dlliter.inc.

```
00071                 { Lst = 0; Hop = 0; Copy(it); }
00072
```

4.12.1.3 ConstIterator::~ConstIterator () `[inline]`

Definition at line 72 of file dlliter.inc.

```
00073                 { if(Lst) Lst->Dereference(); Lst=0; Hop = 0; }
00074
```

4.12.1.4 ConstIterator::ConstIterator (const DLList< T > & *dll*) `[inline, private]`

Definition at line 75 of file dlliter.inc.

```
00076                 {
00077                   Lst = dll.the_rep;
00078                   if(Lst) Lst->Reference();
00079                   Reset();
00080
```

4.12.2 Member Function Documentation

4.12.2.1 void ConstIterator::Copy (const ConstIterator & *it*) `[inline]`

Definition at line 55 of file dlliter.inc.

Referenced by ConstIterator(), and operator=().

```
00056                 {
00057                   if(Lst != it.Lst)
00058                     {
00059                       if(Lst) Lst->Dereference();
00060                       Lst = it.Lst;
00061                       if(Lst) Lst->Reference();
00062                     }
00063
```

4.12.2.2 void ConstIterator::Reset () `[inline]`

Definition at line 50 of file dlliter.inc.

Referenced by ConstIterator().

```
00050                            { Hop = Lst->Head; }
00051
```

4.12.2.3 const T & ConstIterator::operator * () [inline]

Definition at line 51 of file dlliter.inc.

```
00051                                         { return Hop->Data; }
00052
```

4.12.2.4 ConstIterator::operator int () const [inline]

Definition at line 52 of file dlliter.inc.

```
00052                                        { return Hop!=0; }
00053
```

4.12.2.5 const ConstIterator & ConstIterator::operator++ () [inline]

Definition at line 53 of file dlliter.inc.

```
00054                    { if(Hop) Hop = Hop->Next; return *this; }
00055
```

4.12.2.6 const ConstIterator & ConstIterator::operator= (const ConstIterator & *it*) [inline]

Definition at line 66 of file dlliter.inc.

```
00067                    { Copy(it); return *this; }
00068
```

4.12.3 Friends And Related Function Documentation

4.12.3.1 class DLList [friend]

Definition at line 48 of file dlliter.inc.

4.12.4 Member Data Documentation

4.12.4.1 const DLListNode< T > * ConstIterator::Hop [private]

Definition at line 82 of file dlliter.inc.

4.12.4.2 DLListRep< T > * ConstIterator::Lst [private]

Definition at line 81 of file dlliter.inc.

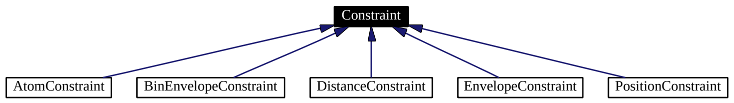
The documentation for this class was generated from the following file:

- **dlliter.inc**

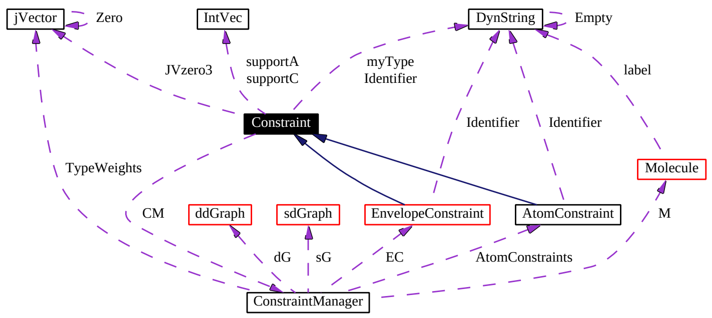
4.13 Constraint Class Reference

`#include <constraint.h>`

Inheritance diagram for Constraint



Collaboration diagram for Constraint:



Public Methods

- virtual ~**Constraint** ()
- int **ID** () const

  Returns the id assigned by the **ConstraintManager** (p. 97).

- **DynString Type** () const

  Returns the string type.

- int **IntType** () const

  Returns the string type.

- double **stress** ()

  Returns how dissatisfied this constraint is with the current molecule (stored in the **Constraint-Manager** (p. 97)*).*

- double **stress** (int a, const **jVector** &pos_a) const
- virtual **jVector dStress** (int k, const **jVector** &pos_a) const=0
- const **IntVec**& **atomSupport** () const

  Returns the id's of the atoms (wrt the constraint manager) this constraint uses to compute its **stress**() (p. 92).

- const **IntVec**& **constraintSupport** () const

  Returns the id's of the constraints this constraint uses to compute its **stress**() (p. 92).

- int **arity** () const

  The number of atoms and/or constraints this constraint uses to compute its **stress**() (p. 92).

- **MAKECOMPARISONS_NO_COMPARE** (Constraint)

  Give me all the comparison operators.

- virtual int **Output** (ostream &os) const
- virtual int **Input** (istream &is)
- virtual int **OutputBinary** (ostream &os) const
- virtual int **InputBinary** (istream &is)
- **MAKE_IO_STREAM_OPERATORS** (Constraint)
- **MAKE_BIN_STREAM_OPERATORS** (Constraint)
- void **setManager** (**ConstraintManager** &cm)
- const **Molecule**& **getMolecule** () const
- const **ConstraintManager**& **getManager** () const
- bool **isOff** () const

  Returns true if the constraint is off and is being ignored during molecular manipulations.

- virtual void **switchOff** (bool to_off=true)
- bool **isValid** () const
- virtual bool **assertValidity** (bool validity)
- virtual void **addAtomSupport** (int Id)
- virtual void **addConstraintSupport** (int Id)
- virtual double **computeStress** () const=0
- virtual double **computeStress** (int a, const **jVector** &pos_a) const=0
- virtual double **computeAbsRes** () const
- **DynString Label** () const

Static Public Attributes

- **DynString Identifier** = "undefined"

Protected Methods

- **Constraint** (**ConstraintManager** &cm, const **IntVec** &supportAIndices=**IntVec**(0),\const **IntVec** &supportCIndices=**IntVec**(0))
- virtual void **notifyOff** (Constraint ∗child, bool to_off=true)
- virtual void **notifyValid** (Constraint ∗child, bool validity)
- virtual void **notifyRemove** ()

  I'm about to be removed from the constraint graph.

- virtual int **Read** (istream &is)=0
- virtual int **ReadBinary** (istream &is)=0

  Same scenerio as **Read**() (p. 85) *only binary version.*

- virtual void **PreSetup** (void)
- virtual void **PostSetup** (void)

Protected Attributes

- **ConstraintManager∗ CM**
- **IntVec supportA**
- **IntVec supportC**
- int **myID**
- **DynString myType**

  String representation of this constraint type.

- int **myIntType**

  Int representation of this constraint type.

- double **cached**
- bool **valid**
- bool **off**
- const **jVector JVzero3**

Friends

- class **ConstraintManager**
- class **ConstraintRegistry**

4.13.1 Constructor & Destructor Documentation

4.13.1.1 Constraint::∼Constraint () [virtual]

Definition at line 16 of file constraint.C.

```
00017   { CM=0; }
00018
```

4.13.1.2 Constraint::Constraint (ConstraintManager & *cm*, const IntVec & *supportAIndices* = IntVec(0), \const IntVec & *supportCIndices* = IntVec(0)) [protected]

Must be constructed with a **ConstraintManager** (p. 97) (for consistency) and a list of indices specifying which atoms and/or constraints this constraint needs to compute its stress function.

4.13.2 Member Function Documentation

4.13.2.1 int Constraint::ID () const [inline]

Returns the id assigned by the **ConstraintManager** (p. 97).

Definition at line 41 of file constraint.h.

Referenced by BinEnvelopeConstraint::BinEnvelopeConstraint(), AtomConstraint::Compare(), ConstraintManager::TurnEnvelopeOn(), notifyOff(), and switchOff().

```
00041                 { return myID; }
00042
```

4.13.2.2 int Constraint::Input (istream & *is*) `[virtual]`

Definition at line 446 of file constraint.C.

Referenced by EnvelopeConstraint::PostSetup(), and ConstraintRegistry::readConstraints().

```
{
  //    cerr << "Constratint Input Called" << endl;
  // Remove myself from the manager:
  CM->remove(*this);

  // Read my own data from the stream:
  int result = Read(is);
  // FIX may want to check result here -jd 0 means error result ~= number of reads done.

  // Setup any internal stuff the constraint needs to set up before
  // being logged with the constraint manager.
  PreSetup();

  // Add myself back to the manager.  ** ASSIGN an ID **
  CM->add(*this);

  PostSetup();

  // At this point this constraint should be "live" -- ready to go.
  // We need to add the existence of this constraint to the
  // constraint support of all the atoms in its own atom support.

  CM->makeDynamic();
  Constraint * c =0;
  for (int i = 0; i < supportA.Length(); i++){
    c = CM->dG->GetData(supportA[i]);
    c->addConstraintSupport(myID);
  }

```

4.13.2.3 int Constraint::InputBinary (istream & *is*) `[virtual]`

Definition at line 481 of file constraint.C.

```
{
  // Remove myself from the manager:
  CM->remove(*this);

  // Read myself in:

  int result = ReadBinary(is);

  // Setup any internal stuff the constraint wants to
  PreSetup();

  // Add myself back to the manager:
  CM->add(*this);

  PostSetup();

  return result;

```

4.13.2.4 int Constraint::IntType () const [inline]

Returns the string type.

Definition at line 47 of file constraint.h.

Referenced by ConstraintManager::AbsDistResiduals().

```
00047                    {return myIntType;}
00048
```

4.13.2.5 DynString Constraint::Label () const

Definition at line 416 of file constraint.C.

```
00416                                 {
00417   return myType + " " + myID;
00418
```

4.13.2.6 Constraint::MAKECOMPARISONS_NO_COMPARE (Constraint)

Give me all the comparison operators.

4.13.2.7 Constraint::MAKE_BIN_STREAM_OPERATORS (Constraint)

4.13.2.8 Constraint::MAKE_IO_STREAM_OPERATORS (Constraint)

4.13.2.9 int Constraint::Output (ostream & *os*) const [inline, virtual]

Reimplemented in **AtomConstraint** (p. 57), **BinEnvelopeConstraint** (p. 73), **DistanceConstraint** (p. 156), **EnvelopeConstraint** (p. 234), and **PositionConstraint** (p. 387).

Definition at line 99 of file constraint.h.

```
00100       { os << Constraint::Identifier << supportA << ',' << supportC; return support; }
00101
```

4.13.2.10 int Constraint::OutputBinary (ostream & *os*) const [inline, virtual]

Reimplemented in **AtomConstraint** (p. 57), **BinEnvelopeConstraint** (p. 73), **DistanceConstraint** (p. 156), **EnvelopeConstraint** (p. 235), and **PositionConstraint** (p. 387).

Definition at line 112 of file constraint.h.

```
00113     { return 0; }
00114
```

4.13.2.11 void Constraint::PostSetup (void) [inline, protected, virtual]

constraints get their ID when added to the constraint graph. this addition is done in the Input function between the pre- and post setup functions. constraint setup that needs to occur with an ID and an added constraint go in post setup. (like adding parent constraints, etc.)

Reimplemented in **BinEnvelopeConstraint** (p. 74), and **EnvelopeConstraint** (p. 235).

Definition at line 284 of file constraint.h.

Referenced by Input(), and InputBinary().

```
00284                                      {}
00285
```

4.13.2.12 void Constraint::PreSetup (void) `[inline, protected, virtual]`

The setup function is called if the uninialized constraint method is used to create the constraint to allow different data representations within the constraint. Without this, only data read in from the input file will be available to the member functions. For simple constraints, this is really not very useful. However for multi-atom constraints that need to be a little 'smarter' about data representations, the Setup function will be helpful.

Reimplemented in **BinEnvelopeConstraint** (p. 74), **DistanceConstraint** (p. 156), **EnvelopeConstraint** (p. 235), and **PositionConstraint** (p. 388).

Definition at line 275 of file constraint.h.

Referenced by Input(), and InputBinary().

```
00275                                     {}
00276
```

4.13.2.13 int Constraint::Read (istream & *is*) `[protected, pure virtual]`

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 85) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 85) function, the derived class should *not* log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented in **AtomConstraint** (p. 57), **BinEnvelopeConstraint** (p. 74), **DistanceConstraint** (p. 157), **EnvelopeConstraint** (p. 236), and **PositionConstraint** (p. 388).

Referenced by Input().

4.13.2.14 int Constraint::ReadBinary (istream & *is*) `[protected, pure virtual]`

Same scenerio as **Read**() (p. 85) only binary version.

Reimplemented in **AtomConstraint** (p. 58), **BinEnvelopeConstraint** (p. 75), **DistanceConstraint** (p. 157), **EnvelopeConstraint** (p. 236), and **PositionConstraint** (p. 388).

Referenced by InputBinary().

4.13.2.15 DynString Constraint::Type () const `[inline]`

Returns the string type.

Definition at line 44 of file constraint.h.

Referenced by switchOff().

```
00044                        {return myType;}
00045
```

4.13.2.16 void Constraint::addAtomSupport (int *Id*) [virtual]

adds an Id to supportA' ; checks to make sure that the Id is not already in the list.

Definition at line 420 of file constraint.C.

```
00420                                     {
00421   if (myID != UNINITIALIZED_CONSTRAINT_ID) {
00422     // I am already in graph, need to add edge:
00423     CM->dG->AddEdge(this, CM->dG->GetData(Id));
00424
00425     printf ("Constraint::addAtomSupport-> added atom support from constraint %d to %d\n SHOULD THIS OCCUR?", myID, Id );
00426
00427   }
00428
```

4.13.2.17 void Constraint::addConstraintSupport (int *Id*) [virtual]

adds an Id to supportC' ; checks to make sure that the Id is not already in the list.

Definition at line 433 of file constraint.C.

Referenced by Input(), and BinEnvelopeConstraint::PreSetup().

```
00433                                           {
00434   if (myID != UNINITIALIZED_CONSTRAINT_ID) {
00435     // I am already in graph, need to add edge:
00436     CM->dG->AddEdge(this, CM->dG->GetData(Id));
00437
00438     //    printf ("Constraint::addConstraintSupport-> added constraint support from constraint %d to %d\n", myID, Id );
00439
00440   }
00441
```

4.13.2.18 int Constraint::arity () const [inline]

The number of atoms and/or constraints this constraint uses to compute its **stress**() (p. 92).

Definition at line 93 of file constraint.h.

Referenced by Output().

```
00093                   { return supportA.Length() + supportC.Length(); }
00094
```

4.13.2.19 bool Constraint::assertValidity (bool *validity*) [virtual]

If validity' is true, the constraint is guaranteed to have the correct stress value cached. It will first make sure its supports are valid by calling assertValidity(true) on them (if they're constraints as well). If validity' is false, it notifies its parents that it has been invalidated. The next call to **stress**() (p. 92) will force a recomputation of the cached value. Returns true if the constraint is valid after the call and false if it is invalid. (!!!) This function writes the cached value from the output of computeStress

Definition at line 305 of file constraint.C.

Referenced by ConstraintManager::InvalidateAtoms(), ConstraintManager::InvalidateEnvelope(), notifyOff(), notifyValid(), ConstraintManager::setMolecule(), ConstraintManager::setPosition(), stress(), and switchOff().

```
{
  // There is no way to become valid since I'm sleeping.
  if(off) {
#ifdef NOISY_CONSTRAINTS
    cerr  << "assertValidity:  " << myID <<"(" << myType
          <<").  I'm off -- cannot change valid flag " << endl;
#endif

    return false;
  }
    // We agree.
  if(valid == validity){

#ifdef NOISY_CONSTRAINTS
    cerr  << "assertValidity:  " << myID <<"(" << myType
          <<") -- my valid flag is already "<< validity << endl;
#endif

    return valid;
  }

    //
    // We disagree. Valid flag will change.
    //

    // I'm supposed to validate myself; i.e. recompute my stress with valid
    // supports.  I need to make sure all the supports are valid first.
    //
  Constraint * child =0;
  if(validity)
    {
      for(int i=0; i<supportC.Length(); ++i)
        {
          child = CM->getConstraint(supportC[i]);
          if (! child->isOff()){
#ifdef NOISY_CONSTRAINTS
            cerr  << "assertValidity:  " << myID <<"(" << myType
                  <<"). I'm trying to validate my supports => now I call assertValidity on " << child->myID
                  <<"(" << child->myType << ")" << endl;
#endif
            if(child->assertValidity(true) == false){
#ifdef NOISY_CONSTRAINTS
              cerr << endl << child->myID << " cannot be made valid. " ;
#endif
              return false;
            } //make child valid
          } // if child is on
        } // over all child constraints


      // Here we're sure that all our supports are valid and we can
      // gleefully compute stress().  After this call, I am valid too.
      cached = computeStress();
      valid = true;
    }

  // I'm supposed to invalidate myself (perhaps my support has become
  // invalid or something).  I need to also notify my parents...
  else
```

```
00365     {
00366       valid = false;
00367 //
00368 //      if (CM->dG){
00369 //      cout << "\n\nDYNAMIC VERSION" << endl;
00370 //      for (int i = 0; i <CM->dG->NumVertices();i++){
00371 //        cout << "i=" << i << "\tmyID=" << CM->dG->GetData(i)->ID()<<endl;
00372 //      }
00373 //      cout << endl;
00374 //      }
00375       if (CM->dG){
00376         if(!CM->makeStatic())
00377           {
00378             DIE("Unable to create static graph (constraint.C).\n");
00379           }
00380       }
00381
00382 //      if (CM->sG){
00383 //      cout << "\n\nSTATIC VERSION" << endl;
00384 //      for (int i = 0; i <CM->sG->NumVertices();i++){
00385 //        cout << "i=" << i << "\tmyID=" << CM->sG->GetData(i)->ID()<<endl;
00386 //      }
00387 //      cout << endl;
00388 //      }
00389
00390
00391       // when I am set to false, I must notify the atoms which depend on me
00392       // that I am now invalid
00393       // this is poor, and breaks the assumption about directedness in
00394       // the graph structure.  However, because atoms acually depend
00395       // on constraints up the graph -- this notification is required.
00396       // possibly fix this.
00397       for(int i=0; i<supportA.Length(); ++i)
00398         {
00399           child = CM->getConstraint(supportA[i]);
00400           child->notifyValid(this,false);
00401         }
00402
00403       Constraint * parent=0;
00404       sdGraph<Constraint*>::neighbor_iterator it;
00405       // Iterate over my parents (who I'm a support for):
00406       for(it = CM->sG->begin_neighbor(myID,1); it; ++it)
00407         {
00408           parent = CM->sG->GetData(*it);
00409           parent->notifyValid(this,false);
00410         }
00411     }
00412
```

4.13.2.20 const IntVec & Constraint::atomSupport () const [inline]

Returns the id's of the atoms (wrt the constraint manager) this constraint uses to compute its **stress**() (p. 92).

Definition at line 87 of file constraint.h.

Referenced by ConstraintManager::add().

```
00087                                   { return supportA; }
00088
```

4.13.2.21 double Constraint::computeAbsRes (void) const [inline, virtual]

returns abs residual as a Z score, if supported, constraint overwrites

Reimplemented in **DistanceConstraint** (p. 157).

Definition at line 191 of file constraint.h.

Referenced by ConstraintManager::AbsDistResiduals().

```
00191                                   {return 0;}
00192
```

4.13.2.22 double Constraint::computeStress (int *k*, const jVector & *pos_k*) const [pure virtual]

Returns the stress on this constraint after "pretending" the atom with index a' moved to position pos_a'. This also must be overwritten by all constraints that inherit from this class

Reimplemented in **AtomConstraint** (p. 58), **BinEnvelopeConstraint** (p. 75), **DistanceConstraint** (p. 158), **EnvelopeConstraint** (p. 237), and **PositionConstraint** (p. 388).

4.13.2.23 double Constraint::computeStress () const [pure virtual]

One of the only functions that *must* be overwritten by subclasses. Returns a value reflecting how dissatisfied this constraint is with the current molecule residing in CM'.

Reimplemented in **AtomConstraint** (p. 58), **BinEnvelopeConstraint** (p. 75), **DistanceConstraint** (p. 158), **EnvelopeConstraint** (p. 237), and **PositionConstraint** (p. 389).

Referenced by assertValidity(), and stress().

4.13.2.24 const IntVec & Constraint::constraintSupport () const [inline]

Returns the id's of the constraints this constraint uses to compute its **stress**() (p. 92).

Definition at line 90 of file constraint.h.

Referenced by ConstraintManager::add().

```
00090                                       { return supportC; }
00091
```

4.13.2.25 jVector Constraint::dStress (int *k*, const jVector & *pos_k*) const [pure virtual]

asking constraint number k for its 12 element dStress vector we pass in pos_a vector – corresponding to the atom position if the constraint is an atom, and corresponding to the position of the atom k in question if the call is to a non-atom constraint. EG for a distance constraint atom 12 needs the dStress from the d-constraint, so it will call dStress on the d-constraint(12, atom_pos) so the d_constr can answer the question if atom 12 were in atom_pos, I would say my dStress is blah.

Reimplemented in **AtomConstraint** (p. 59), **BinEnvelopeConstraint** (p. 76), **DistanceConstraint** (p. 159), **EnvelopeConstraint** (p. 237), and **PositionConstraint** (p. 389).

Referenced by ConstraintManager::dStress().

4.13.2.26 const ConstraintManager & Constraint::getManager () const

Definition at line 134 of file constraint.C.

```
00135 { return *CM; }
00136
```

4.13.2.27 const Molecule & Constraint::getMolecule () const

Definition at line 131 of file constraint.C.

Referenced by DistanceConstraint::computeAbsRes(), PositionConstraint::computeStress(), DistanceConstraint::computeStress(), BinEnvelopeConstraint::computeStress(), and DistanceConstraint::dStress().

```
00132 { return CM->getMolecule(); }
00133
```

4.13.2.28 bool Constraint::isOff () const

Returns true if the constraint is off and is being ignored during molecular manipulations.

Definition at line 138 of file constraint.C.

Referenced by assertValidity(), ConstraintManager::constraintON(), and notifyOff().

```
00139   { return off; }
00140
```

4.13.2.29 bool Constraint::isValid () const

isValid() (p. 90) returns true if the cached' value computed from the last **computeStress**() (p. 89) call would equal the value returned by another call to the **computeStress**() (p. 89) function.

Definition at line 302 of file constraint.C.

```
00303   { return valid; }
00304
```

4.13.2.30 void Constraint::notifyOff (Constraint * *child*, bool *to_off* = true) [protected, virtual]

Informs me that one of the atoms/constraints I depend on has been turned off/on depending on the to_off' flag.

Definition at line 253 of file constraint.C.

Referenced by switchOff().

```
00253                                                            {
00254
00255   // default behavior:
00256
00257   if (child->ID() < CM->numAtoms()){
00258     if (to_off){     // a child atom was just turned off
```

```
00259       assertValidity(false);
00260       switchOff(true);
00261       DIE ("This should not be called -- instead, atoms should turn off constraints when turned off");
00262 
00263     } else { // a child atom I need was just turned on
00264       if (isOff()) {
00265         // and I am off
00266         // if all my other children constraints (atoms) are on, I turn back on
00267       Constraint * c=0;
00268       sdGraph<Constraint*>::neighbor_iterator it;
00269       for(it = CM->sG->begin_neighbor(myID,0);it; ++it)
00270         {
00271           c = CM->sG->GetData(*it);
00272           if (c->isOff())
00273             return;
00274         }
00275       switchOff(false); // turn back on
00276       } else {
00277         // I am already on
00278         // and one of my child atoms just turned on
00279         DIE("constraint error!");
00280       }
00281     } // end child atom on
00282   }
00283 
```

4.13.2.31 void Constraint::notifyRemove (void) [protected, virtual]

I'm about to be removed from the constraint graph.

Reimplemented in **AtomConstraint** (p. 59).

Definition at line 290 of file constraint.C.

Referenced by ConstraintManager::remove().

```
00291 {
00292   Constraint * child;
00293   for(int i=0; i<supportC.Length(); ++i)
00294     {
00295       child = CM->dG->GetData(supportC[i]);
00296       CM->remove(*child);
00297       // Recursively calls notifyRemove() on child''s children.
00298     }
00299 
```

4.13.2.32 void Constraint::notifyValid (Constraint * *child*, bool *validity*) [protected, virtual]

Informs me that one of my supports now contains valid/invalid information depending on the validity' flag.

Reimplemented in **AtomConstraint** (p. 59).

Definition at line 239 of file constraint.C.

Referenced by assertValidity(), and ConstraintManager::setPosition().

```
00239                                                                    {
00240   // this is DEFAULT behavior.  Many Contraints will overwrite this
00241   // especially complex constraints with many deepndancies.
```

```
// This behavior allows doesnot permit ANY intelligent parsing of
// invalid subsets of children constraints.

// no problem if we match
if (validity == valid) return;

// otherwise fix the problem
if (valid) assertValidity(false);

```

4.13.2.33 void Constraint::setManager (ConstraintManager & *cm*)

Definition at line 113 of file constraint.C.

```
{
  if(CM == &cm) return;

  // requierd to check CM because most constraints will be
  // constructed w/ CM=0 initially and the have their manager set
  // with this fntn
  if (CM) CM->remove(*this);
  CM = &cm;
  CM->add(*this);

  cached = INF;
  valid = false;
  off = false;

  // My Id has now been set by the ConstraintManager.

```

4.13.2.34 double Constraint::stress (int *a*, const jVector & *pos_a*) const

Computes the stress after simulating atom a' has moved to the new position pos_a'. This function makes sense for values of a' that reference atoms (not other constraints) in the **Constraint-Manager** (p. 97).

Definition at line 105 of file constraint.C.

```
{
  if(a<0 || a>=CM->numAtoms())
      return INF;
  if (off) return 0;
  return computeStress(a,pos_a); // subclass

```

4.13.2.35 double Constraint::stress ()

Returns how dissatisfied this constraint is with the current molecule (stored in the **Constraint-Manager** (p. 97)).

Definition at line 21 of file constraint.C.

Referenced by ConstraintManager::stress().

```
{
  if(off)
```

```
      return 0;
      //       {
// #ifdef NOISY_CONSTRAINTS
//          cerr  << "Constraint::Stress:  " << myID <<"(" << myType
//                <<"). my off value is " << off << endl;
// #endif
//      DIE("stress() called on an inactive constraint.\n");
//       }
    //    cerr << "called stress() on constraint" << Label() << endl;

    if (assertValidity(true)) return cached;
    else {
      DIE("Called Constraint::stress() and unable to validate.");
      return 0; // removes compiler warning re:control to end of non-void fntn
    }

```

4.13.2.36 void Constraint::switchOff (bool *to_off* = true) `[virtual]`

Turn the constraint on or off. If the constraint is off, it is ignored during subsequent molecular manipulations. Subclasses may want to overwrite these to propigate on/off states to children (constraints they've constructed) when being switched on/off. By default,

Definition at line 141 of file constraint.C.

Referenced by ConstraintManager::TurnAtomOn(), ConstraintManager::TurnAtomsOff(), ConstraintManager::TurnConstraintOn(), and notifyOff().

```
00141                                       {
00142
00143   if(off == to_off) {
00144 #ifdef NOISY_CONSTRAINTS
00145     cerr  << "Turning off value for  " << myID <<"(" << myType
00146           <<").  to " << to_off << "  no change. " << endl;
00147 #endif
00148
00149     return;
00150   }
00151
00152   // I'm turning off
00153   if (to_off) assertValidity(false);
00154
00155 #ifdef NOISY_CONSTRAINTS
00156   cerr  << "I'm  " << myID <<"(" << myType
00157         << ") My off value is now " << to_off << endl;
00158 #endif
00159
00160   off = to_off;
00161
00162   // I just truned on
00163   if (! to_off) assertValidity(false);
00164
00165
00166
00167   CM->makeStatic();
00168   Constraint * c; // Some (child or parent) constraint.
00169   sdGraph<Constraint*>::neighbor_iterator it;
00170
00171
00172   if (myID < CM->numAtoms()){
00173     // I'm an atom
00174
00175     // Switch our child constraints on/off to match our own on/off status.
```

```
    for(it = CM->sG->begin_neighbor(myID,0);it; ++it)
      {
        c = CM->sG->GetData(*it);


        if (to_off){ // I'm an atom and just got turned off
#ifdef NOISY_CONSTRAINTS
          cerr  << "Atom Constraint  " << myID <<"(" << myType
                <<") now switching  " << c->ID() <<" "  << c->Type()
                << " off value to " << to_off << endl;
#endif
          c->switchOff(to_off);

        }         else { // I'm an atom and just got turned on
#ifdef NOISY_CONSTRAINTS
          cerr  << "Atom Constraint  " << myID <<"(" << myType
                <<") now notifying  " << c->ID() <<" "  << c->Type()
                << " that my off value is now " << to_off << endl;
#endif
          c->notifyOff(this, to_off);


        }
      }

  } else {
    // I'm a constraint on atoms or on other constraints



        // Switch our child constraints on/off to match our own on/off status.
    for(it = CM->sG->begin_neighbor(myID,0);it; ++it)
      {
        c = CM->sG->GetData(*it);

        if (c->ID() >= CM->numAtoms()){
#ifdef NOISY_CONSTRAINTS
          cerr  << "I'm  " << myID <<"(" << myType
                <<") turning  " << c->ID() <<" "  << c->Type()
                << " off value to " << to_off << endl;
#endif
          c->switchOff(to_off);
        } else {// else child is an atom, and don't turn off an atom (no loops)
          c->assertValidity(false);
          // atom must now be invalid b/c a parent turned on
        }
      }

    // Notify the parents that our on/off state has switched.
    for(it = CM->sG->begin_neighbor(myID,1);it; ++it)
      {
        c = CM->sG->GetData(*it);
#ifdef NOISY_CONSTRAINTS
        cerr  << "I'm  " << myID <<"(" << myType
              <<") notifying  " << c->ID() << " " << c->Type()
              << " that my off val is now " << to_off << endl;
#endif
        c->notifyOff(this, to_off);
      }
  }
```

4.13.3 Friends And Related Function Documentation

4.13.3.1 class ConstraintManager [friend]

Definition at line 33 of file constraint.h.

4.13.3.2 class ConstraintRegistry [friend]

Definition at line 34 of file constraint.h.

4.13.4 Member Data Documentation

4.13.4.1 ConstraintManager * Constraint::CM [protected]

Pointer to the manager handling me. If I want to reference a molecule, I must get to it through my constraint manager. This extra abstraction step is meant to guarantee that the manager and I agree on which molecule we're viewing.

Definition at line 204 of file constraint.h.

4.13.4.2 DynString Constraint::Identifier = "undefined" [static]

Reimplemented in **AtomConstraint** (p. 60), **BinEnvelopeConstraint** (p. 76), **DistanceConstraint** (p. 160), **EnvelopeConstraint** (p. 239), and **PositionConstraint** (p. 390).

Definition at line 88 of file globals.C.

4.13.4.3 const jVector Constraint::JVzero3 [protected]

Definition at line 317 of file constraint.h.

4.13.4.4 double Constraint::cached [protected]

Definition at line 299 of file constraint.h.

4.13.4.5 int Constraint::myID [protected]

This id is assigned by the **ConstraintManager** (p. 97) when the constraint is added (during construction).

Definition at line 290 of file constraint.h.

4.13.4.6 int Constraint::myIntType [protected]

Int representation of this constraint type.

Definition at line 296 of file constraint.h.

4.13.4.7 DynString Constraint::myType [protected]

String representation of this constraint type.

Definition at line 293 of file constraint.h.

4.13.4.8 bool Constraint::off [protected]

Definition at line 308 of file constraint.h.

4.13.4.9 IntVec Constraint::supportA [protected]

Stores the id's of the atoms that this constraint needs in order to compute its **stress**() (p. 92) function. It should retrieve these ID's from the constraint manager. The constraint manager assigns integers 0...(N-1) to atoms (where N is the number of atoms in the molecule) and the remainder to constraints.

Definition at line 213 of file constraint.h.

4.13.4.10 IntVec Constraint::supportC [protected]

Stores the id's of the other constraints that this constraint needs in order to compute its **stress**() (p. 92) function. It should retrieve these ID's from the constraint manager. The constraint manager assigns integers 0...(N-1) to atoms (where N is the number of atoms in the molecule) and the remainder to constraints.

Definition at line 222 of file constraint.h.

4.13.4.11 bool Constraint::valid [protected]

Definition at line 304 of file constraint.h.

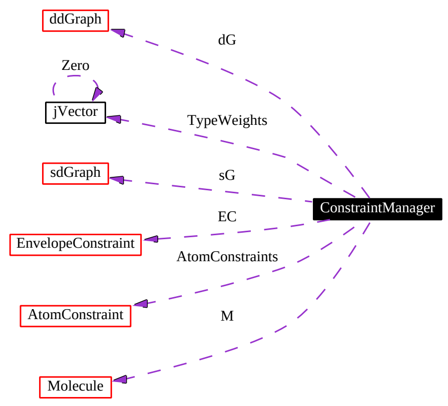
The documentation for this class was generated from the following files:

- **constraint.h**
- **constraint.C**
- **globals.C**

4.14 ConstraintManager Class Reference

#include <constraint_manager.h>

Collaboration diagram for ConstraintManager:



Public Methods

- **ConstraintManager** (const **Molecule** &mol)
- virtual **~ConstraintManager** ()
- int **Number** () const
- int **numAtoms** () const
- int **numAtomsON** () const
- int **numConstraints** () const
- const **Molecule**& **getMolecule** () const
- const **Atom**& **getAtom** (int i) const
- const **jVector**& **getPosition** (int i) const
- **jVector molecule_center** (void) const
- double **getRadius** (int i) const
- **EnvelopeConstraint**∗ **getEnvelope** () const
- void **InvalidateEnvelope** (void) const
- void **TurnEnvelopeOn** (void) const
- void **InvalidateAtoms** (void) const
- const **Atom**& **operator**[] (int i) const
- const **jVector**& **operator()** (int i) const
- **DLList**<int> **approx_neighbor_i** (const **jVector** &p, double distance) const
- **DLList**<int> **neighbor_i** (const **jVector** &p, double distance) const
- bool **constraintON** (int i) const
- **DLList**<**Constraint**∗> **getParents** (int Id) const
- **DLList**<**Constraint**∗> **getAssocAtoms** (int Id) const

  Returns the list of atoms that are involved in some constraint with the atom.

- **Constraint**∗ **getConstraint** (int id) const

  Returns the pointer to the constraint with id id'.

- ostream& **PrintGraph** (ostream &os)

output a graphical version of the constraint graph for examination.

- virtual double **getWeight** (int id) const
- void **setMolecule** (const **Molecule** &mol)
- void **readNewMoleculeXmol** (ifstream &is)
- void **setPosition** (int i, const **jVector** &v)
- void **setAtom** (int i, const **Atom** &a)
- void **registerEnvelope** (**EnvelopeConstraint** *EC_in)
- void **TurnAtomsOff** () const

  Turn all atoms constratins off.

- void **TurnAtomOn** (int i) const

  Turn single atom on.

- void **TurnConstraintOn** (int i) const
- void **TranslateModel** (**jVector** dir)

  Moves all atoms in model dir.

- void **RandomPerturb** (double dist)

  Perturbs all atoms +/- dist in each direction.

- void **resetMoved** (void)

  set all atoms in molecule to 'not moved yet'.

- virtual void **setWeight** (int id)
- virtual double **CombineStress** (const **IntVec** &IdList, const **jVector** &stressList) const
- double **stress** (int atom)

  Return stress total stress for an atom.

- double **stress** (int atom, const **jVector** &position) const

  Return total stress for an atom as if it were in a new posistion.

- double **stress** ()

  Return sum of stress over all aloms in the molecule.

- **jVector getStressList** ()

  Returns a **jVector** (p. 325) *of the stress values; one for each atom.*

- **jVector dStress** (int atom) const

  Returns the gradient and hessian of stress for an atom.

- **jVector dStress** (int atom, const **jVector** &position) const

  Returns the gradient and hessian of stress for an atom for a new position.

- void **AbsDistResiduals** (double &max, double &ave) const

  Summary Functions.

Protected Types

- typedef **ddGraph<Constraint∗> dynamicGraph**
- typedef **sdGraph<Constraint∗> staticGraph**

Protected Methods

- bool **isDynamic** () const
- bool **makeStatic** ()
- bool **makeDynamic** ()
- void **remove** (**Constraint** &c)
- int **add** (**Constraint** &c)
- void **clearConstraints** ()

  Erases all the constraints that are not atoms.

- void **clearAllConstraints** ()

  Erases all the constraints, even atoms.

Protected Attributes

- **EnvelopeConstraint∗ EC**
- **dynamicGraph∗ dG**

  The dynamic constraint graph.

- **staticGraph∗ sG**

  The static constraint graph.

- **AtomConstraint∗ AtomConstraints**

  Array of dummy constraints, each holding one atom in the graph.

- **Molecule M**

  The single molecule this mgr works on.

Private Attributes

- **jVector TypeWeights**

Friends

- class **ConstraintRegistry**
- class **Constraint**

4.14.1 Member Typedef Documentation

4.14.1.1 typedef ddGraph< Constraint *> ConstraintManager::dynamicGraph `[protected]`

Definition at line 207 of file constraint_manager.h.

Referenced by makeDynamic().

4.14.1.2 typedef sdGraph< Constraint *> ConstraintManager::staticGraph `[protected]`

Definition at line 208 of file constraint_manager.h.

Referenced by makeStatic().

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ConstraintManager::ConstraintManager (const Molecule & *in_mol*)

Definition at line 4 of file constraint_manager.C.

```
{
  M = in_mol;

  //  Weightlist.Resize(M.Number());

  dG = new dynamicGraph;
  sG = 0;
  EC = 0;

  // assign a constraint to each atom here and add it to the graph.
  // assures that constraints numbered 0 -- n-1 are all atom
  // constraints
  for(int i=0; i<M.Number(); ++i){
    add(*(new AtomConstraint(*this)));
    getConstraint(i)->myType = AtomConstraint::Identifier;
  }

```

4.14.2.2 ConstraintManager::~ConstraintManager () `[virtual]`

Definition at line 24 of file constraint_manager.C.

```
{
  //
  // We *do* remove the constraints since we dont know if we
  // were responsible for allocating this space or not (we could have
  // read in some constraints from a file, allocated the space for these
  // constraints, and then extra constraints could've been added later
  // that someone else allocated.  The client should *not* deallocate
  // constraints!
  //

  makeStatic();

  // cycle through complete graph and delete all constraints
  int num = Number();
```

```
00039   for(int i = 0; i < num; i++){
00040     delete (sG->GetData(i));
00041   }
00042
```

4.14.3 Member Function Documentation

4.14.3.1 void ConstraintManager::AbsDistResiduals (double & *max_rv*, double & *ave_rv*) const

Summary Functions.

Definition at line 499 of file constraint_manager.C.

```
00499                                                                        {
00500   int max_c = Number();
00501   int dc_type = ConstraintRegistry::getIntType(DistanceConstraint::Identifier);
00502   int n_dist = 0;
00503
00504   double temp;
00505
00506   max_rv = -1 * INF;
00507   ave_rv = 0;
00508
00509   for (int i = numAtoms();i<max_c;i++){
00510     if (constraintON(i)){
00511       if (getConstraint(i)->IntType() == dc_type){
00512         n_dist++;
00513         temp = getConstraint(i)->computeAbsRes();
00514         //      cerr << getConstraint(i)->Label();
00515         if (max_rv < temp)
00516           max_rv = temp;
00517         ave_rv += temp;
00518       }
00519     }
00520   }
00521
```

4.14.3.2 double ConstraintManager::CombineStress (const IntVec & *IdList*, const jVector & *stressList*) const [inline, virtual]

Definition at line 142 of file constraint_manager.h.

```
00143                                               {
00144     //    not defn yet
00145     return 0;
00146
```

4.14.3.3 void ConstraintManager::InvalidateAtoms (void) const

Definition at line 554 of file constraint_manager.C.

Referenced by readNewMoleculeXmol().

```
00555 {  for (int i = numAtoms()-1;i>=0;i--){
00556   //for (int i =0;i<numAtoms();i++) {
00557   getConstraint(i)->assertValidity(false);}
00558
```

4.14.3.4 void ConstraintManager::InvalidateEnvelope (void) const

Definition at line 528 of file constraint_manager.C.

```
00529 {if (EC) EC->assertValidity(false);}
00530
```

4.14.3.5 int ConstraintManager::Number () const

Definition at line 212 of file constraint_manager.C.

Referenced by AbsDistResiduals(), numConstraints(), ConstraintRegistry::readConstraints(), and ~ConstraintManager().

```
00213 {
00214   if (sG)
00215     return sG->NumVertices();
00216   else
00217     return dG->NumVertices();
00218
```

4.14.3.6 ostream & ConstraintManager::PrintGraph (ostream & *os*)

output a graphical version of the constraint graph for examination.

Definition at line 220 of file constraint_manager.C.

```
00220                                                 {
00221   makeStatic();
00222   sG->PrintDotty(os,0);
00223   return os;
00224
```

4.14.3.7 void ConstraintManager::RandomPerturb (double *dist*)

Perturbs all atoms +/- dist in each direction.

Definition at line 71 of file constraint_manager.C.

```
00072 {
00073
00074   RanDbl randomDist((-1)*dist, dist);
00075   jVector tempPos(3); // 3 element vector
00076
00077     for(int i=0; i<M.Number(); i++)
00078     {
00079       tempPos = getPosition(i);
00080       tempPos[0] += randomDist.Get();
00081       tempPos[1] += randomDist.Get();
00082       tempPos[2] += randomDist.Get();
00083       setPosition(i, tempPos);
00084     }
00085
```

4.14.3.8 void ConstraintManager::TranslateModel (jVector *dir*)

Moves all atoms in model dir.

Definition at line 51 of file constraint_manager.C.

```
00052 {
00053   if (dir.nrm() < 0.0001)
00054     return;
00055
00056   jVector tempV1;
00057   int num = numAtoms();
00058
00059   for (int a = 0; a < num; a++){
00060     if (constraintON(a)){
00061       tempV1 = getPosition(a);
00062       tempV1[0] += dir[0];
00063       tempV1[1] += dir[1];
00064       tempV1[2] += dir[2];
00065       setPosition(a, tempV1);
00066     }
00067   }
00068
```

4.14.3.9 void ConstraintManager::TurnAtomOn (int *i*) const

Turn single atom on.

Definition at line 377 of file constraint_manager.C.

```
00377                                           {
00378   if (i<M.Number()){
00379     cerr  << setw(3)  << i << " ";
00380     if (0 == i%10) {cerr << endl;}
00381
00382 #ifdef NOISY_CONSTRAINTS
00383     cerr  << endl << "CM is turning atom " << i << " on " << endl;
00384 #endif
00385     getConstraint(i)->switchOff(false);
00386   } else {
00387     DIE ("Tried to turn non-atom constraint on");
00388   }
00389
```

4.14.3.10 void ConstraintManager::TurnAtomsOff () const

Turn all atoms constratins off.

Definition at line 365 of file constraint_manager.C.

```
00365                                        {
00366
00367   for(int i=0; i<M.Number(); ++i){
00368 #ifdef NOISY_CONSTRAINTS
00369     cerr  << endl << "CM is turning constaint " << i << " off " << endl;
00370 #endif
00371
00372     getConstraint(i)->switchOff(true);
00373   }
00374
```

4.14.3.11 void ConstraintManager::TurnConstraintOn (int *i*) const

Definition at line 392 of file constraint_manager.C.

Referenced by TurnEnvelopeOn().

```
00392                                              {
00393
00394 #ifdef NOISY_CONSTRAINTS
00395     cerr  << endl << "CM is turning constraint " << i << " on " << endl;
00396 #endif
00397     getConstraint(i)->switchOff(false);
00398
```

4.14.3.12 void ConstraintManager::TurnEnvelopeOn (void) const

Definition at line 531 of file constraint_manager.C.

```
00531                                                {
00532   if (EC) {
00533     int i = EC->ID();
00534     if (!(constraintON(i))){
00535       TurnConstraintOn(i);
00536     }
00537   }
00538
```

4.14.3.13 int ConstraintManager::add (Constraint & *c*) [protected]

Only friends (e.g. Constraints) and derived classes may add or remove themselves from the manager. The **add**() (p. 104) function returns the id assigned to the newly added constraint.

Definition at line 226 of file constraint_manager.C.

Referenced by ConstraintManager(), Constraint::Input(), Constraint::InputBinary(), and Constraint::setManager().

```
00227 {
00228
00229   makeDynamic();
00230
00231   // Must add all atoms before any constraints on atoms.
00232   // This reserves the indices from 0...(M.Number()-1)
00233   // for use as atom ids.  The rest of the indices are used for
00234   // assigning unique identifiers to constraints.
00235
00236   // Generate an ID for this constraint.
00237   if((c.myID = dG->AddVertex(&c))<0)
00238     {
00239       DIE("ConstraintManager::add-> Problems adding new constraint to constraint graph.\n");
00240       return 1;
00241     }
00242
00243   // Create edges to children' of c'
00244   IntVec children = cat(c.atomSupport(),c.constraintSupport());
00245   Constraint * child;
00246
00247   for(int i=0; i<children.Length(); ++i)
00248     {
00249       // Add an edge from this constraint to the atom/constraint
```

```
00250       // it needs to compute its stress function:
00251
00252       if (children[i]>=0){
00253         // NB uninitialized constraints declare support intvecs with
00254         // appropriate length and set values = -1
00255
00256         child = dG->GetData(children[i]);
00257         dG->AddEdge(&c, child);
00258       }
00259     }
00260
```

4.14.3.14 DLList< int > ConstraintManager::approx_neighbor_i (const jVector & *p*, double *distance*) const [inline]

Definition at line 282 of file constraint_manager.h.

```
00283 {return M.approx_neighbor_i(p,distance); }
00284
```

4.14.3.15 void ConstraintManager::clearAllConstraints () [protected]

Erases all the constraints, even atoms.

Definition at line 424 of file constraint_manager.C.

```
00424                                         {
00425   makeDynamic();
00426     for (int i = (dG->NumVertices())-1; i >=0; i--)
00427       remove(*(dG->GetData(i)));
00428
```

4.14.3.16 void ConstraintManager::clearConstraints () [protected]

Erases all the constraints that are not atoms.

Definition at line 430 of file constraint_manager.C.

Referenced by ConstraintRegistry::readConstraints().

```
00431 {
00432   makeDynamic();
00433     for (int i = (dG->NumVertices())-1; i >=M.Number(); i--)
00434       remove(*(dG->GetData(i)));
00435
```

4.14.3.17 bool ConstraintManager::constraintON (int *i*) const

Definition at line 541 of file constraint_manager.C.

Referenced by AbsDistResiduals(), TranslateModel(), TurnEnvelopeOn(), molecule_center(), numAtomsON(), and setPosition().

```
00541                                              {
00542   //  cerr << "CONSTRAINT " << i << " (off) is " << getConstraint(i)->isOff() << endl;
00543
```

4.14.3.18 jVector ConstraintManager::dStress (int *atom*, const jVector & *position*) const

Returns the gradient and hessian of stress for an atom for a new position.

Definition at line 186 of file constraint_manager.C.

```
{
  return getConstraint(atom)->dStress(atom,position) ;

//  /*
//  jVector return_vect(12);
//  // FIX THIS ... THIS IS A TEMPORARY HACK
//  // 0,1,2 are gradient, 3-11 are hessian
//  // The way dStress now works, the CM *has* to also
//  // return the hessian.  Come see me w/ questions -jd
//
//   DLList<Constraint*>::ConstIterator it;
//
//   for(it = AtomConstraintList[atom].GetConstIterator(); it; ++it)
//     return_vect += (*it)->dStress(atom, position);
//   return return_vect;
//  */

```

4.14.3.19 jVector ConstraintManager::dStress (int *atom*) const

Returns the gradient and hessian of stress for an atom.

Definition at line 205 of file constraint_manager.C.

```
{

  return getConstraint(atom)->dStress(atom, M(atom));


```

4.14.3.20 DLList< Constraint ∗> ConstraintManager::getAssocAtoms (int *Id*) const

Returns the list of atoms that are involved in some constraint with the atom.

Definition at line 299 of file constraint_manager.C.

```
{
  DLList<Constraint*> ReturnList;

  if (Id >= M.Number()) return ReturnList;

  DLList<Constraint*> ParentList;
  ParentList = getParents(Id);

  IntVec atomSeen(M.Number());  // Keeps track of which atoms we've seen.
  IntVec atomList;      // List of atom (indices) in a constraint.
  int sibling;

  atomSeen.zero();      // Mark that we havent seen anything.
  atomSeen[Id] = 1; // Mark that we've seen the source atom.

  DLList<Constraint *>::ConstIterator it;
```

```
00316
00317   for(it = ParentList.GetConstIterator(); it; ++it)
00318     {
00319       atomList = ($\ast$it)->atomSupport();
00320       for(int i=0; i<atomList.Length(); i++)      {
00321         sibling = atomList[i];  // A mutually-constrained atom index.
00322
00323         // If we havent seen this atom, add it to our list.
00324         if(!atomSeen[sibling])  {
00325           ReturnList.AddToBack(sG->GetData(sibling));
00326           atomSeen[sibling] = 1;
00327         }
00328       }
00329     }
00330
```

4.14.3.21 const Atom & ConstraintManager::getAtom (int *i*) const [inline]

Definition at line 277 of file constraint_manager.h.

Referenced by operator[]().

```
00277                                                       { return M[i]; }
00278
```

4.14.3.22 Constraint ∗ ConstraintManager::getConstraint (int *id*) const

Returns the pointer to the constraint with id id'.

Definition at line 447 of file constraint_manager.C.

Referenced by AbsDistResiduals(), ConstraintManager(), InvalidateAtoms(), TurnAtomOn(), TurnAtomsOff(), TurnConstraintOn(), Constraint::assertValidity(), constraintON(), dStress(), setPosition(), and stress().

```
00448 {if(sG)
00449     return sG->GetData(id);
00450   else
00451
```

4.14.3.23 EnvelopeConstraint ∗ ConstraintManager::getEnvelope () const [inline]

Definition at line 291 of file constraint_manager.h.

```
00292 {return EC;}
00293
```

4.14.3.24 const Molecule & ConstraintManager::getMolecule () const [inline]

Definition at line 274 of file constraint_manager.h.

Referenced by SatisfyMolecule::getMolecule(), and Constraint::getMolecule().

```
00274                                                         { return M; }
00275
```

4.14.3.25 DLList< Constraint *> ConstraintManager::getParents (int *Id*) const

Returns the parents of a constraint or atom. The parents are those constraints whose support contains a' or c'.

Definition at line 277 of file constraint_manager.C.

Referenced by getAssocAtoms().

```
00278 {
00279   //  makeStatic();
00280   DLList<Constraint*> ReturnList;
00281   if (sG){
00282 
00283 
00284     staticGraph::neighbor_iterator it;
00285     // cycle through our parents and collect pointers in list
00286     for(it = sG->begin_neighbor(Id,1);it; ++it)
00287       {
00288         ReturnList.AddToBack(sG->GetData(*it));
00289         // *it is an int
00290       }
00291     return ReturnList;
00292 
00293   }
00294   else {
00295     DIE("getParents called on dynamic constraint graph");
00296     return ReturnList; // keep compiler happy
00297   }
00298 
```

4.14.3.26 const jVector & ConstraintManager::getPosition (int *i*) const `[inline]`

Definition at line 278 of file constraint_manager.h.

Referenced by RandomPerturb(), TranslateModel(), molecule_center(), and operator()().

```
00278                                                                  { return M(i); }
00279 
```

4.14.3.27 double ConstraintManager::getRadius (int *i*) const `[inline]`

Definition at line 279 of file constraint_manager.h.

```
00279                                                        { return M.VDW(i); }
00280 
```

4.14.3.28 jVector ConstraintManager::getStressList ()

Returns a **jVector** (p. 325) of the stress values; one for each atom.

Definition at line 152 of file constraint_manager.C.

```
00153 {
00154   int numCyc = M.Number();
00155   jVector StressList(numCyc);
00156   for (int i =0; i<numCyc; i++){
00157     StressList[i] = stress(i);
```

```
00158     //    cerr << "Label: " << getConstraint(i)->Label() << "STRESS: " << stress(i) << endl;
00159
00160   }
00161
```

4.14.3.29 double ConstraintManager::getWeight (int *id*) const [inline, virtual]

Definition at line 105 of file constraint_manager.h.

```
00105                                       {
00106     // not defn yet
00107     return 0;
00108
```

4.14.3.30 bool ConstraintManager::isDynamic () const [protected]

Returns true if the constraint graph is dynamic at this moment, and false if it is stored statically.

Definition at line 460 of file constraint_manager.C.

```
00461 { return (dG!=0); }
00462
```

4.14.3.31 bool ConstraintManager::makeDynamic () [protected]

Converts the graph from dynamic to static and vice verca. Returns true if successful and false otherwise.

Definition at line 479 of file constraint_manager.C.

Referenced by Constraint::Input(), add(), clearAllConstraints(), clearConstraints(), remove(), and setMolecule().

```
00480 {
00481   if(dG)
00482     return true;
00483   if(sG)
00484     {
00485       if(!(dG = new dynamicGraph(*sG)))
00486         return false;
00487       delete sG; sG = 0;
00488       return true;
00489     }
00490
```

4.14.3.32 bool ConstraintManager::makeStatic () [protected]

Converts the graph from dynamic to static and vice verca. Returns true if successful and false otherwise.

Definition at line 463 of file constraint_manager.C.

Referenced by PrintGraph(), Constraint::assertValidity(), Constraint::switchOff(), and ~ConstraintManager().

```
00464 {
00465   if(sG)
00466     return true;
00467
00468   if(!dG)
00469     return false;
00470
00471   if(!(sG = new staticGraph(*dG, true)))
00472     return false;
00473
00474   delete dG; dG = 0;
00475
00476   return true;
00477
```

4.14.3.33 jVector ConstraintManager::molecule_center (void) const

Definition at line 138 of file constraint_manager.C.

```
00139 {
00140   jVector rv(3,0);
00141   double m;
00142
00143   for(int i=numAtoms()-1; i>=0; i--){
00144     if (constraintON(i)){
00145       rv += getPosition(i);
00146       m++;
00147     }
00148   }
00149
```

4.14.3.34 DLList< int > ConstraintManager::neighbor_i (const jVector & *p*, double *distance*) const `[inline]`

Definition at line 284 of file constraint_manager.h.

```
00285 {return M.neighbor_i(p,distance); }
00286
```

4.14.3.35 int ConstraintManager::numAtoms () const `[inline]`

Definition at line 286 of file constraint_manager.h.

Referenced by AbsDistResiduals(), AnimFrame::AnimFrame(), InvalidateAtoms(), TranslateModel(), molecule_center(), Constraint::notifyOff(), numAtomsON(), numConstraints(), Constraint::stress(), and Constraint::switchOff().

```
00286                                                     { return M.Number(); }
00287
```

4.14.3.36 int ConstraintManager::numAtomsON () const

Definition at line 545 of file constraint_manager.C.

```
                                        {
int rv = 0;
for (int i = numAtoms()-1;i>=0;i--){
  if (constraintON(i))
    rv++;
}

```

4.14.3.37 int ConstraintManager::numConstraints () const [inline]

Definition at line 61 of file constraint_manager.h.

```
  {
    return Number()-numAtoms();

```

4.14.3.38 const jVector & ConstraintManager::operator() (int *i*) const [inline]

Definition at line 281 of file constraint_manager.h.

```
                                                            { return getPosition(i); }

```

4.14.3.39 const Atom & ConstraintManager::operator[] (int *i*) const [inline]

Definition at line 280 of file constraint_manager.h.

```
                                                         { return getAtom(i); }

```

4.14.3.40 void ConstraintManager::readNewMoleculeXmol (ifstream & *is*) [inline]

Definition at line 276 of file constraint_manager.h.

```
                                                           {M.readXmol(is);InvalidateAtoms();}

```

4.14.3.41 void ConstraintManager::registerEnvelope (EnvelopeConstraint ∗ *EC_in*) [inline]

Definition at line 289 of file constraint_manager.h.

```
{if (EC) DIE("only 1 envelope allowed"); EC = EC_in;}

```

4.14.3.42 void ConstraintManager::remove (Constraint & *c*) [protected]

Only friends (e.g. Constraints) and derived classes may add or remove themselves from the manager. The **add**() (p. 104) function returns the id assigned to the newly added constraint.

Definition at line 263 of file constraint_manager.C.

Referenced by Constraint::Input(), Constraint::InputBinary(), clearAllConstraints(), clearConstraints(), Constraint::notifyRemove(), and Constraint::setManager().

```
00264 {
00265   // Make sure this constraint has a valid ID:
00266   if(dG && !dG->HasVertex(&c))
00267     return;
00268   else if(sG && !sG->HasVertex(&c))
00269     return;
00270
00271   makeDynamic();                // Changing graph architecture, so need dynamic.
00272   c.notifyRemove();             // Notify the constraint it's being removed.
00273   // Recursively calls CM->remove() on all of c''s constraint support.
00274   dG->RemVertex(&c);    // Disconnect its node in the graph.
00275
```

4.14.3.43 void ConstraintManager::resetMoved (void) [inline]

set all atoms in molecule to 'not moved yet'.

Definition at line 287 of file constraint_manager.h.

```
00288 {for (int i = 0; i < M.Number(); i++) M[i].Moved()=false;}
00289
```

4.14.3.44 void ConstraintManager::setAtom (int *i*, const Atom & *a*) [inline]

Definition at line 117 of file constraint_manager.h.

```
00117                                      { M[i] = a; }
00118
```

4.14.3.45 void ConstraintManager::setMolecule (const Molecule & *mol*)

Definition at line 115 of file constraint_manager.C.

Referenced by SatisfyMolecule::setMolecule().

```
00116 {
00117   if(M.Number() != mol.Number())
00118     {
00119       DIE("Invalid new molecule!\n");
00120     }
00121
00122   M = mol;
00123   Constraint * constraint;
00124   if(makeDynamic())
00125     {
00126       int numCyc = dG->NumVertices();
00127       for(int i=M.Number(); i<=numCyc; i++)
```

```
00128         {
00129           constraint = dG->GetData(i);
00130           constraint->assertValidity(false);
00131         }
00132     }  else {
00133       DIE(" Error in setMolecule(...)");
00134     }
00135
```

4.14.3.46 void ConstraintManager::setPosition (int *i*, const jVector & *v*)

Definition at line 87 of file constraint_manager.C.

Referenced by RandomPerturb(), and TranslateModel().

```
00088 {
00089   if (i>M.Number()-1){
00090     DIE("tried to setPosition on non-Atom constraint");
00091   }
00092
00093   if (constraintON(i)){
00094
00095     // Set this constraint to be invalid when moved
00096     Constraint * C = getConstraint(i);
00097     C->assertValidity(false);
00098
00099     // Notify those constraints which depend on this atom
00100     // position that I have changed.
00101     Constraint * SuppC = 0;
00102     int numC = C->supportC.Length();
00103
00104     for (int j = 0;j<numC;j++){
00105       SuppC = getConstraint(C->supportC[j]);
00106       SuppC->notifyValid(C,false);
00107     }
00108     M.setPosition(i,v);
00109     M[i].Moved() = true; // set atom moved' to true
00110   } else {
00111     DIE ("setPosition called for atom that is off");
00112   }
00113
```

4.14.3.47 void ConstraintManager::setWeight (int *id*) `[inline, virtual]`

Definition at line 137 of file constraint_manager.h.

```
00137                                 {
00138     // not defined yet
00139     return;
00140
```

4.14.3.48 double ConstraintManager::stress ()

Return sum of stress over all aloms in the molecule.

Definition at line 175 of file constraint_manager.C.

Referenced by gwNumerics::addLPERTSegments(), and getStressList().

```
00176 {
00177
00178   // calculates the stress of te whole molecule as the sum
00179
00180   double total = 0;
00181   for (int i =0; i<M.Number(); i++)
00182     total += stress(i);
00183   return total;
00184
```

4.14.3.49 double ConstraintManager::stress (int *atom*, const jVector & *position*) const

Return total stress for an atom as if it were in a new posistion.

Definition at line 170 of file constraint_manager.C.

```
00171 {
00172   return getConstraint(atom)->stress(atom,position) ;
00173
```

4.14.3.50 double ConstraintManager::stress (int *Id*)

Return stress total stress for an atom.

Definition at line 165 of file constraint_manager.C.

```
00166 {
00167   return getConstraint(Id)->stress();
00168
```

4.14.4 Friends And Related Function Documentation

4.14.4.1 class Constraint [friend]

Definition at line 48 of file constraint_manager.h.

4.14.4.2 class ConstraintRegistry [friend]

Definition at line 47 of file constraint_manager.h.

4.14.5 Member Data Documentation

4.14.5.1 AtomConstraint ∗ ConstraintManager::AtomConstraints [protected]

Array of dummy constraints, each holding one atom in the graph.

Definition at line 224 of file constraint_manager.h.

4.14.5.2 EnvelopeConstraint ∗ ConstraintManager::EC [protected]

//

Constraint (p. 80) Graphs:

Description:

There are 2 constraint graphs: one that is dynamic and one that is static. Only one of these graphs exists at any one point in time. When a constraint manager is constructed, it is given a new dynamic graph, dG, and the static graph, sG, is undefined. Friend classes will check whether the current graph of constraints is being stored statically or dynamically and act accordingly.

//

Definition at line 205 of file constraint_manager.h.

4.14.5.3 Molecule ConstraintManager::M `[protected]`

The single molecule this mgr works on.

Definition at line 260 of file constraint_manager.h.

Referenced by dStress(), and getPosition().

4.14.5.4 jVector ConstraintManager::TypeWeights `[private]`

Definition at line 269 of file constraint_manager.h.

4.14.5.5 dynamicGraph * ConstraintManager::dG `[protected]`

The dynamic constraint graph.

Definition at line 211 of file constraint_manager.h.

4.14.5.6 staticGraph * ConstraintManager::sG `[protected]`

The static constraint graph.

Definition at line 214 of file constraint_manager.h.

The documentation for this class was generated from the following files:

- **constraint_manager.h**
- **constraint_manager.C**

4.15 ConstraintRegistry Class Reference

`#include <constraint_registry.h>`

Static Public Methods

- **DynString getIdentifier** (int i=0)
- **Constraint* getConstraint** (const **DynString** &identifier, **ConstraintManager** &cm)
- int **readConstraints** (istream &is, **ConstraintManager** &cm, bool append=true)
- int **getIntType** (const **DynString** &identifier)

4.15.1 Detailed Description

//

File: contsraint_registry.h

The constraint registry generally interfaces the GRAPE system with real constraint types. It allows the reading of multiple different types of constraints and oganizes the correct creation of those constraints.

In the future I want the registry to facilitate run-time manipulation of constraint properties for a subset of constraints. This ability will require the calling function to have the ability to specify constraint types explicitly to the constraint manager. -jd

//

Definition at line 35 of file constraint_registry.h.

4.15.2 Member Function Documentation

4.15.2.1 Constraint ∗ ConstraintRegistry::getConstraint (const DynString & *identifier*, ConstraintManager & *cm*) [static]

Definition at line 24 of file constraint_registry.C.

Referenced by readConstraints().

```
{

  // NB rigth now this is only called by the readConstraints command below.
  //
  Constraint * c = 0;
    // if none matches, then 0 & ignore.
    // if # matches, return -1, and skip rest of line.

  if(identifier == DistanceConstraint::Identifier){
    c = new DistanceConstraint(cm);
    c->myType = DistanceConstraint::Identifier;
  }
  else if(identifier == AtomConstraint::Identifier){
    c = new AtomConstraint(cm);
    c->myType = AtomConstraint::Identifier;
    DIE("Atoms are being read during constraint addition.  Not Allowed");
  }
  else if(identifier == PositionConstraint::Identifier){
    c = new PositionConstraint(cm);
    c->myType = PositionConstraint::Identifier;
```

```
00046   }
00047   else if(identifier == EnvelopeConstraint::Identifier){
00048     c = new EnvelopeConstraint(cm);
00049     c->myType = EnvelopeConstraint::Identifier;
00050
00051   }
00052
```

4.15.2.2 DynString ConstraintRegistry::getIdentifier (int *i* = 0) [static]

4.15.2.3 int ConstraintRegistry::getIntType (const DynString & *identifier*) [static]

Definition at line 6 of file constraint_registry.C.

Referenced by ConstraintManager::AbsDistResiduals(), BinEnvelopeConstraint::PreSetup(), and getConstraint().

```
00006                                                              {
00007   // this is the ONLY place these numbers should be accessed
00008   if(identifier == AtomConstraint::Identifier)
00009     return 0;
00010   else if(identifier == DistanceConstraint::Identifier)
00011     return 1;
00012   else if(identifier == PositionConstraint::Identifier)
00013     return 2;
00014   else if(identifier == EnvelopeConstraint::Identifier)
00015     return 3;
00016   else if(identifier == BinEnvelopeConstraint::Identifier)
00017     return 4;
00018   else
00019     DIE("improper identifier");
00020
00021   return -1; // compiler warning
00022
```

4.15.2.4 int ConstraintRegistry::readConstraints (istream & *is*, ConstraintManager & *cm*, bool *append* = true) [static]

Definition at line 66 of file constraint_registry.C.

Referenced by main().

```
00068   {
00069     if(!append)
00070         cm.clearConstraints();
00071
00072     int start_num = cm.Number();
00073
00074     DynString constraintIdentifier;
00075     Constraint * c=0;
00076     //    while(!is.eof() && !is.bad())
00077     while(is.good())
00078       {
00079         // Read in the constraint identifier:
00080         //
00081         is >> constraintIdentifier;
00082
00083         // Given this identifer, retrieve the appropriate constraint
00084         // instance:
```

```
      //
      c = getConstraint(constraintIdentifier, cm);

      // cout <<  " \"" << constraintIdentifier << "\"";

      // Now that we have a particular type of constraint, we can
      // read it in:
      if(c > 0) {
        //cout << ",  read " <<  c->myType << endl;
        c->Input(is);  // The constraint manager "magically" logs this.
      }
    }
  if (cm.Number() - start_num > 0)
    return cm.Number();
  else {
    DIE ("Read 0 constaints from constraint file");
    return 0;
  }

```

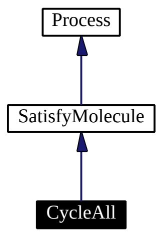
The documentation for this class was generated from the following files:

- **constraint_registry.h**
- **constraint_registry.C**

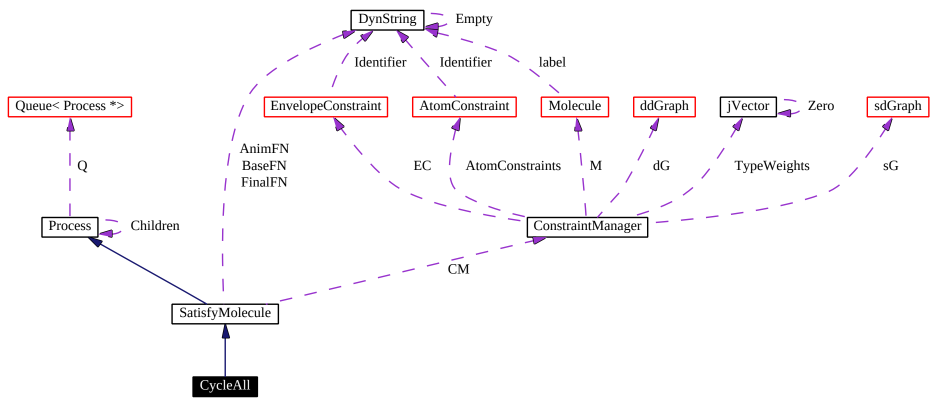
4.16 CycleAll Class Reference

#include <cycle_all.h>

Inheritance diagram for CycleAll



Collaboration diagram for CycleAll:



Public Methods

- **CycleAll** (const CycleAll &ca)
- **CycleAll** (**ConstraintManager** &cm)
- ~**CycleAll** ()
- int **run** ()

4.16.1 Detailed Description

//

File: **cycle_all.h**

Created by: Jon Dugan

This is a 'job' that works in the GRAPE system. It touches each atom once and relaxes its position.

//

Definition at line 21 of file cycle_all.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 CycleAll::CycleAll (const CycleAll & *ca*)

Definition at line 5 of file cycle_all.C.

```
00006   : SatisfyMolecule(*(ca.CM)){}
00007
```

4.16.2.2 CycleAll::CycleAll (ConstraintManager & *cm*)

Definition at line 9 of file cycle_all.C.

```
00010   : SatisfyMolecule(cm)
00011   {
00012     // Create one child process for each atom in the molecule:
00013     numC = CM->numAtoms();
00014     Children = new Process*[numC];
00015
00016     // Create a new relaxation process for every atom in the molecule:
00017     for(int i=0; i<numC; ++i)
00018       Children[i] = new gwRelaxAtom(*CM,i);
00019
00020
```

4.16.2.3 CycleAll::∼CycleAll ()

Definition at line 22 of file cycle_all.C.

```
00023   {
00024     // Don't delete my children (since Process takes care of it).
00025     // i.e. dont do for(int i=0; i<numChildren; ++i) delete Children[i].
00026     // or delete[] Children.
00027
```

4.16.3 Member Function Documentation

4.16.3.1 int CycleAll::run () `[virtual]`

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 29 of file cycle_all.C.

```
00029                   {
00030 #ifdef JOB_VERBOSE
00031   cerr << "Start: \tCycleAll::run()" << endl;
00032 #endif
00033
00034     // save the largest current distance residual to statics
00035     CM->AbsDistResiduals(gwNumerics::MaxAbsDistRes,
00036                          gwNumerics::AveAbsDistRes);
00037
00038     // Force Envelope update next update
00039     CM->InvalidateEnvelope();
00040
```

```
#ifdef JOB_VERBOSE2
        cerr << setprecision(50) << "ORDER:  \t" << "MaxAbsDistRes = "<< gwNumerics::MaxAbsDistRes << endl;
#endif


  //      if(*maxcyclerr<0.0001) break;
  if (gwNumerics::MaxAbsDistRes < 0.0001)
    return 1; // stop now

  // Compute the priorities of the children, based on their
  // residuals
  double tempPriority;
  for(int i=0; i<numC; ++i)
    {
      if (CM->constraintON(i)){
        tempPriority = CM->stress(i);

#ifdef JOB_VERBOSE2
        cerr << setprecision(50) << "Set ORDER:  \t" << "Priority for atom " <<i<<"="<< tempPriority << endl;
#endif

      Children[i]->setPriority(tempPriority);

      }
    }

  CM->resetMoved();

    // Queue all the children to run queue
  //        for(int i=0; i<numC; ++i)    {
        for(int i=numC-1; i>=0; i--)    {
          if (CM->constraintON(i))
            addChild(Children[i]);
    }
```

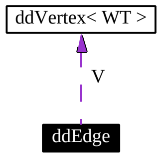
The documentation for this class was generated from the following files:

- **cycle_all.h**
- **cycle_all.C**

4.17 ddEdge Class Template Reference

`#include <ddvertex.h>`

Collaboration diagram for ddEdge:



Public Methods

- **ddEdge** ()
- ~**ddEdge** ()
- int **Compare** (const ddEdge< WT > &e) const
- int **Input** (istream &is)
- int **Output** (ostream &os) const
- int **InputBinary** (istream &is)
- int **OutputBinary** (ostream &os) const
- **MAKECOMPARISONS_WITH_COMPARE** (ddEdge< WT >)
- **MAKE_IO_STREAM_OPERATORS** (ddEdge< WT >)
- **MAKE_BIN_STREAM_OPERATORS** (ddEdge< WT >)

Public Attributes

- **ddVertex**<WT>∗ **V**
- WT **W**

template<class WT = double> class ddEdge

4.17.1 Constructor & Destructor Documentation

4.17.1.1 template<class WT = double> ddEdge<WT>::ddEdge<WT> () `[inline]`

Definition at line 29 of file ddvertex.h.

```
00029                             { V=0; }
00030
```

4.17.1.2 template<class WT = double> ddEdge<WT>::~ddEdge<WT> () `[inline]`

Definition at line 30 of file ddvertex.h.

```
00030                              { /*empty*/ }
00031
```

4.17.2 Member Function Documentation

4.17.2.1 template<class WT> int ddEdge< WT >::Compare (const ddEdge< WT > & *e*) const `[inline]`

Definition at line 2 of file ddvertex.inc.

```
{ return (V->ID - e.V->ID); }
```

4.17.2.2 template<class WT = double> int ddEdge<WT>::Input (istream & *is*) `[inline]`

Definition at line 39 of file ddvertex.h.

```
                { is >> W; return 1; }
```

4.17.2.3 template<class WT = double> int ddEdge<WT>::InputBinary (istream & *is*) `[inline]`

Definition at line 44 of file ddvertex.h.

```
                { is >= W; return 1; }
```

4.17.2.4 template<class WT = double> ddEdge<WT>::MAKECOMPARISONS_-WITH_COMPARE (ddEdge< WT >)

4.17.2.5 template<class WT = double> ddEdge<WT>::MAKE_BIN_STREAM_-OPERATORS (ddEdge< WT >)

4.17.2.6 template<class WT = double> ddEdge<WT>::MAKE_IO_STREAM_-OPERATORS (ddEdge< WT >)

4.17.2.7 template<class WT = double> int ddEdge<WT>::Output (ostream & *os*) const `[inline]`

Definition at line 41 of file ddvertex.h.

```
                { os << W; return 1; }
```

4.17.2.8 template<class WT = double> int ddEdge<WT>::OutputBinary (ostream & *os*) const `[inline]`

Definition at line 46 of file ddvertex.h.

```
                { os <= W; return 1; }
```

4.17.3 Member Data Documentation

4.17.3.1 template<class WT = double> ddVertex< WT > * ddEdge<WT>::V

Definition at line 32 of file ddvertex.h.

4.17.3.2 template<class WT = double> WT ddEdge<WT>::W

Definition at line 34 of file ddvertex.h.

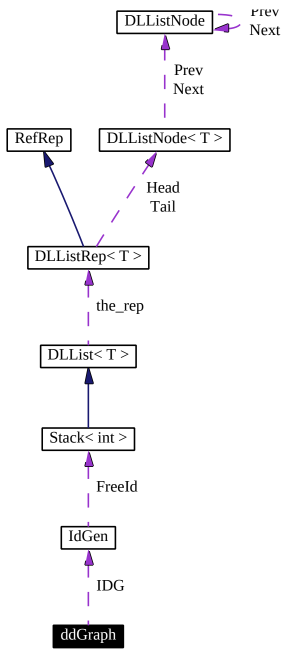
The documentation for this class was generated from the following files:

- **ddvertex.h**
- **ddvertex.inc**

4.18 ddGraph Class Template Reference

`#include <ddgraph.h>`

Collaboration diagram for ddGraph:



Public Types

- typedef ddGraph<DT,WT> **DDG**

Public Methods

- **ddGraph** ()
- **ddGraph** (const ddGraph< DT, WT > &ddg)
- **ddGraph** (const **sdGraph**< DT, WT > &sdg)
- **~ddGraph** ()
- void **Clear** ()
- void **Copy** (const ddGraph< DT, WT > &ddg)
- void **Copy** (const **sdGraph**< DT, WT > &ddg)
- int **NumVertices** () const
- int **NumEdges** () const
- int **HasVertex** (const DT &x) const
- int **HasVertex** (int n) const
- int **HasEdge** (const DT &from, const DT &to) const
- int **OutDegree** (const DT &x) const
- int **InDegree** (const DT &x) const
- int **Degree** (const DT &x) const
- int **NextFreeId** () const
- int **AddVertex** (const DT &x)

- **ddVertex**<WT>* **GetVertex** (const DT &x)
- int **GetVertexId** (const DT &x) const
- DT& **GetData** (int vid)
- const DT& **GetData** (int vid) const
- DT& **GetData** (const **ddVertex**< WT > *v)
- const DT& **GetData** (const **ddVertex**< WT > *v) const
- int **AddEdge** (const DT &from, const DT &to, const WT &fwd_weight, const WT &bwd_weight)
- int **AddEdge** (const DT &from, const DT &to, const WT &weight)
- int **AddEdge** (const DT &from, const DT &to)
- const WT& **GetWeight** (const DT &from, const DT &to, int fwd=1) const
- const WT& **operator()** (const DT &from, const DT &to) const
- void **SetWeight** (const DT &from, const DT &to, const WT &fwd_weight, const WT &bwd_weight)
- void **SetWeight** (const DT &from, const DT &to, const WT &weight)
- void **ChangeWeight** (const DT &from, const DT &to, const WT &fwd_del, const WT &bwd_del)
- void **ChangeWeight** (const DT &from, const DT &to, const WT &del)
- void **RemOutEdges** (const DT &x)
- void **RemInEdges** (const DT &x)
- void **RemEdges** (const DT &x)
- int **RemVertex** (const DT &x)
- int **RemEdge** (const DT &from, const DT &to)
- const ddGraph<DT,WT>& **operator=** (const ddGraph< DT, WT > &ddg)
- const ddGraph<DT,WT>& **operator=** (const **sdGraph**< DT, WT > &sdg)
- void **OutputSimple** (ostream &os)
- int **Output** (ostream &os) const
- int **Input** (istream &is)
- **MAKE_IO_STREAM_OPERATORS** (**DDG**)

Private Types

- typedef **ddEdge**<WT> **Edge**
- typedef **ddVertex**<WT> **Vertex**
- typedef map<DT, **Vertex***> **VertexArray**
- typedef map< int, DT > **DataArray**

Private Methods

- int **OutDegree** (const **ddVertex**< WT > *vert) const
- int **InDegree** (const **ddVertex**< WT > *vert) const
- int **Degree** (const **ddVertex**< WT > *vert) const
- int **AddEdge** (**ddVertex**< WT > *from, **ddVertex**< WT > *to, const WT &fwd_weight, const WT &bwd_weight)
- int **RemEdge** (**ddVertex**< WT > *from, **ddVertex**< WT > *to)
- void **SetWeight** (**ddVertex**< WT > *from, **ddVertex**< WT > *to, const WT &fwd_weight, const WT &bwd_weight)
- void **ChangeWeight** (**ddVertex**< WT > *from, **ddVertex**< WT > *to, const WT &fwd_del, const WT &bwd_del)

Private Attributes

- **IdGen IDG**
- int **NumV**
- int **NumE**
- **VertexArray VA**
- **DataArray Data**

Friends

- class **sdGraph< DT, WT >**
- istream& **InputBinary** (istream &is, ddGraph< DT, WT > &ddg)
- istream& **operator>>** (istream &is, ddGraph< DT, WT > &ddg)
- ostream& **OutputBinary** (ostream &os, const ddGraph< DT, WT > &ddg)
- ostream& **operator** (ostream &os, const ddGraph< DT, WT > &ddg)

template<class DT, class WT = double> class ddGraph

4.18.1 Member Typedef Documentation

4.18.1.1 template<class DT, class WT = double> typedef ddGraph< DT, WT > ddGraph<DT, WT>::DDG

Definition at line 167 of file ddgraph.h.

4.18.1.2 template<class DT, class WT = double> typedef map< int, DT > ddGraph<DT, WT>::DataArray `[private]`

Definition at line 191 of file ddgraph.h.

4.18.1.3 template<class DT, class WT = double> typedef ddEdge< WT > ddGraph<DT, WT>::Edge `[private]`

Definition at line 184 of file ddgraph.h.

4.18.1.4 template<class DT, class WT = double> typedef ddVertex< WT > ddGraph<DT, WT>::Vertex `[private]`

Definition at line 185 of file ddgraph.h.

4.18.1.5 template<class DT, class WT = double> typedef map< DT, Vertex *> ddGraph<DT, WT>::VertexArray `[private]`

Definition at line 186 of file ddgraph.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 template<class DT, class WT> ddGraph< DT, WT >::ddGraph<DT, WT> () `[inline]`

Definition at line 132 of file ddgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
```

4.18.2.2 template<class DT, class WT> ddGraph< DT, WT >::ddGraph<DT, WT> (const ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 139 of file ddgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
  Copy(ddg);
```

4.18.2.3 template<class DT, class WT> ddGraph< DT, WT >::ddGraph<DT, WT> (const sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 147 of file ddgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
  Copy(sdg);
```

4.18.2.4 template<class DT, class WT> ddGraph< DT, WT >::~ddGraph<DT, WT> () `[inline]`

Definition at line 155 of file ddgraph.inc.

```
{ Clear(); }
```

4.18.3 Member Function Documentation

4.18.3.1 template<class DT, class WT> int ddGraph< DT, WT >::AddEdge (ddVertex< WT > * *from*, ddVertex< WT > * *to*, const WT & *fwd_weight*, const WT & *bwd_weight*) `[inline, private]`

Definition at line 275 of file ddgraph.inc.

```
{
  if(!from && !to)
     return VertexBad;
```

```

    Edge eF, eT;
    eT.V = to;   eT.W = fwd_weight;
    eF.V = from; eF.W = bwd_weight;

    // Check if the edge already exists
    if(from->OutE.count(eT)>0)
        return EdgeInG;

    from->OutE.insert(eT);
    to->InE.insert(eF);
    return ++NumE;

```

4.18.3.2 template<class DT, class WT> int ddGraph< DT, WT >::AddEdge (const DT & *from*, const DT & *to*) [inline]

Definition at line 317 of file ddgraph.inc.

```
  { WT weight; return AddEdge(from,to,weight,weight); }

```

4.18.3.3 template<class DT, class WT> int ddGraph< DT, WT >::AddEdge (const DT & *from*, const DT & *to*, const WT & *weight*) [inline]

Definition at line 313 of file ddgraph.inc.

```
  { return AddEdge(from,to,weight,weight); }

```

4.18.3.4 template<class DT, class WT> int ddGraph< DT, WT >::AddEdge (const DT & *from*, const DT & *to*, const WT & *fwd_weight*, const WT & *bwd_weight*) [inline]

Definition at line 295 of file ddgraph.inc.

Referenced by AddEdge(), Copy(), Input(), and ConstraintManager::add().

```
  {
    ddVertex<WT> * FromV, * ToV;

    // If we dont have the vertices, just add them!
    //
    if((!HasVertex(from) && (AddVertex(from)<0)) ||
       (!HasVertex(to) && (AddVertex(to)<0)))
           return VertexAddFailure;

    FromV = VA[from];
    ToV   = VA[to];

    return AddEdge(FromV,ToV,fwd_weight,bwd_weight);

```

4.18.3.5 template<class DT, class WT> int ddGraph< DT, WT >::AddVertex (const DT & *x*) [inline]

Definition at line 217 of file ddgraph.inc.

Referenced by AddEdge(), Copy(), and ConstraintManager::add().

```
{
  if(HasVertex(x)) return VA[x]->ID;
  int id = IDG.Get();
  Data[id] = x;
  ddVertex<WT> * vert = new ddVertex<WT>(id);
  VA[x] = vert;
  ++NumV;
  return id;
```

4.18.3.6 template<class DT, class WT> void ddGraph< DT, WT >::ChangeWeight (ddVertex< WT > ∗ *from*, ddVertex< WT > ∗ *to*, const WT & *fwd_del*, const WT & *bwd_del*) [inline, private]

Definition at line 367 of file ddgraph.inc.

```
{
  if(!from || !to)
    { cerr << "ddGraph::ChangeWeight\n"; raise(SIGQUIT); }
```

4.18.3.7 template<class DT, class WT> void ddGraph< DT, WT >::ChangeWeight (const DT & *from*, const DT & *to*, const WT & *delta*) [inline]

Definition at line 386 of file ddgraph.inc.

```
{ ChangeWeight(from,to,delta,delta); }
```

4.18.3.8 template<class DT, class WT> void ddGraph< DT, WT >::ChangeWeight (const DT & *from*, const DT & *to*, const WT & *fwd_del*, const WT & *bwd_del*) [inline]

Definition at line 377 of file ddgraph.inc.

Referenced by ChangeWeight().

```
{
  if(!HasEdge(from,to))
    { cerr << "ddGraph::ChangeWeight\n"; raise(SIGQUIT); }
```

4.18.3.9 template<class DT, class WT> void ddGraph< DT, WT >::Clear () [inline]

Definition at line 55 of file ddgraph.inc.

Referenced by Copy(), Input(), and ∼ddGraph().

```
{
  for(VertexArray::iterator it = VA.begin(); it!=VA.end(); ++it)
      delete ($\ast$it).second;
  VA.clear();
  Data.clear();
  IDG.Reset();
  NumV = 0;
  NumE = 0;

```

4.18.3.10 template<class DT, class WT> void ddGraph< DT, WT >::Copy (const sdGraph< DT, WT > & *sdg*) [inline]

Definition at line 94 of file ddgraph.inc.

```
{
  Clear();

  // Add all the edges to the graph:
  //
  int k;
  sdEdge<WT> edge;
  edge.V = new sdVertex<WT>;
  for(int i=0; i<sdg.NumV; ++i)
    {
      if(AddVertex(sdg.Data[i])<0)
          { cerr << "ddGraph::Copy(sdGraph)#1\n"; raise(SIGQUIT); }
      edge.V->ID = i;
      for(int j=0; j<sdg.VA[i]->NumO; ++j)
        {
          k = BinSearch(sdg.VA[i]->OutE[j].V->InE,
                        sdg.VA[i]->OutE[j].V->NumI,
                        edge);
          if(k<0 || k>=sdg.VA[i]->OutE[j].V->NumI)
            { cerr << "ddGraph::Copy(sdGraph)#2\n"; raise(SIGQUIT); }

          AddEdge(sdg.Data[i],
                  sdg.Data[(sdg.VA[i]->OutE[j].V->ID)],
                  sdg.VA[i]->OutE[j].W,
                  sdg.VA[i]->OutE[j].V->InE[k].W);
        }
    }

```

4.18.3.11 template<class DT, class WT> void ddGraph< DT, WT >::Copy (const ddGraph< DT, WT > & *ddg*) [inline]

Definition at line 67 of file ddgraph.inc.

Referenced by ddGraph(), and operator=().

```
{
  if(this == &ddg) return;
  Clear();
  if(ddg.NumV<=0) return;

  // Add all the edges to the graph (this automatically takes care
  // of adding vertices btw.
  //
  set<Edge>::const_iterator eit;
```

```
  Edge edge;
  edge.V = new ddVertex<WT>;
  VertexArray::const_iterator vit;
  for(vit = ddg.VA.begin(); vit!=ddg.VA.end(); ++vit)
    {
      if((edge.V->ID = AddVertex((*vit).first))<0)
          { cerr << "ddGraph::Copy(ddGraph)\n"; raise(SIGQUIT); }
      for(eit = (($\ast$vit).second)->OutE.begin();
          eit!=(($\ast$vit).second)->OutE.end(); ++eit)
          AddEdge((*vit).first,
                  ddg.Data[($\ast$eit).V->ID],
                  eit->W, ((($\ast$eit).V)->InE).find(edge)->W);
    }

```

4.18.3.12 template<class DT, class WT = double> int ddGraph<DT, WT>::Degree (const ddVertex< WT > * *vert*) const [private]

4.18.3.13 template<class DT, class WT> int ddGraph< DT, WT >::Degree (const DT & *x*) const [inline]

Definition at line 205 of file ddgraph.inc.

```
{
  if(!HasVertex(x)) return VertexNotInG;
  ddVertex<WT>* vert = VA[x];
  return (vert->OutDegree() + vert->InDegree());

```

4.18.3.14 template<class DT, class WT> const DT & ddGraph< DT, WT >::GetData (const ddVertex< WT > * *v*) const [inline]

Definition at line 267 of file ddgraph.inc.

```
{
  if(!v)
    { cerr << "ddGraph::GetData(ddVertex<WT>*)\n"; raise(SIGQUIT); }

```

4.18.3.15 template<class DT, class WT> DT & ddGraph< DT, WT >::GetData (const ddVertex< WT > * *v*) [inline]

Definition at line 259 of file ddgraph.inc.

```
{
  if(!v)
    { cerr << "ddGraph::GetData(ddVertex<WT>*)\n"; raise(SIGQUIT); }

```

4.18.3.16 template<class DT, class WT = double> const DT& ddGraph<DT, WT>::GetData (int *vid*) const

4.18.3.17 template<class DT, class WT> DT & ddGraph< DT, WT >::GetData (int *vid*) [inline]

Definition at line 243 of file ddgraph.inc.

Referenced by sdGraph::Copy(), GetData(), ConstraintManager::add(), ConstraintManager::clearAllConstraints(), ConstraintManager::clearConstraints(), ConstraintManager::getConstraint(), and ConstraintManager::setMolecule().

```
{
  if(Data.count(vid)<1)
    { cerr << "ddGraph::GetData("<<vid<<")\n"; raise(SIGQUIT); }

```

4.18.3.18 template<class DT, class WT> ddVertex< WT > * ddGraph< DT, WT >::GetVertex (const DT & *x*) [inline]

Definition at line 229 of file ddgraph.inc.

Referenced by sdGraph::Copy().

```
{
  if(!HasVertex(x)) { cerr << "ddGraph::GetVertex\n"; raise(SIGQUIT); }

```

4.18.3.19 template<class DT, class WT> int ddGraph< DT, WT >::GetVertexId (const DT & *x*) const [inline]

Definition at line 236 of file ddgraph.inc.

```
{
  if(!HasVertex(x)) { cerr << "ddGraph::GetVertexId\n"; raise(SIGQUIT); }

```

4.18.3.20 template<class DT, class WT> const WT & ddGraph< DT, WT >::GetWeight (const DT & *from*, const DT & *to*, int *fwd* = 1) const [inline]

Definition at line 321 of file ddgraph.inc.

Referenced by operator()().

```
{
  Edge edge;

  ddVertex<WT> * FromV, * ToV;
  if(!HasVertex(from) || !HasVertex(to))
    { cerr << "ddGraph::GetWeight\n"; raise(SIGQUIT); }

  FromV = VA[from];
  ToV   = VA[to];

  if(fwd)
    {
      edge.V = ToV;
      return FromV->OutE.Find(edge).W;
    }

```

4.18.3.21 template<class DT, class WT> int ddGraph< DT, WT >::HasEdge (const DT & *from*, const DT & *to*) const [inline]

Definition at line 175 of file ddgraph.inc.

Referenced by ChangeWeight(), and SetWeight().

```
{
  //
  // If either of the vertices is not in the graph then the edge must not be.
  //
  if(!HasVertex(from) || !HasVertex(to)) return 0;

  ddVertex<WT> * FromV = VA[from], * ToV = VA[to];
  Edge eF; eF.V = FromV;
  Edge eT; eT.V = ToV;
  return (FromV->OutE.In(eT) && ToV->InE.In(eF));
```

4.18.3.22 template<class DT, class WT> int ddGraph< DT, WT >::HasVertex (int *n*) const [inline]

Definition at line 171 of file ddgraph.inc.

```
{ return (Data.count(n)>0); }
```

4.18.3.23 template<class DT, class WT> int ddGraph< DT, WT >::HasVertex (const DT & *x*) const [inline]

Definition at line 167 of file ddgraph.inc.

Referenced by AddEdge(), AddVertex(), sdGraph::Copy(), Degree(), GetVertex(), GetVertexId(), GetWeight(), HasEdge(), InDegree(), OutDegree(), RemEdge(), RemEdges(), RemInEdges(), RemOutEdges(), RemVertex(), and ConstraintManager::remove().

```
{ return (VA.count(x)>0); }
```

4.18.3.24 template<class DT, class WT = double> int ddGraph<DT, WT>::InDegree (const ddVertex< WT > * *vert*) const [private]

4.18.3.25 template<class DT, class WT> int ddGraph< DT, WT >::InDegree (const DT & *x*) const [inline]

Definition at line 197 of file ddgraph.inc.

```
{
  if(!HasVertex(x)) return VertexNotInG;
  ddVertex<WT>* vert = VA[x];
  return vert->InDegree();
```

4.18.3.26 template<class DT, class WT> int ddGraph< DT, WT >::Input (istream & *is*) [inline]

Definition at line 512 of file ddgraph.inc.

```
{
  //
  int id, from_id, to_id;
  DT data;
  WT weight;
  ddVertex<WT> * new_v, * from, * to;
  VertexArray::iterator it;
  DynString str;

  map<int,int> IdToOcc;
      // Keeps track of vertex Id occurrences.

  map<int, ddVertex<WT>* > IdToPtr;
      // Keeps a mapping of the id's to their vertices.

  // First we need to delete everything in the graph.
  Clear();

  int error = 0;
  int re_id = 0;

  // Skip any space before the beginning vertex delimiter:
  for(is >> str; str != V_BEGIN_DELIM; is >> str);

  // Read in the vertices until another delimiter is encountered:
  for(is >> str; str != V_END_DELIM; is >> str)
    {
      // The string should now contain an id and the next information
      // in the stream should contain the data item:
      //
      id = str.ToInt();
      if(is.eof() || is.bad()) { error = 1; break; }
      is >> data;

      if(id < 0) re_id = 1;

      // See if this id has already been used.  If it hasnt, record that
      // we've seen it, then create it and place it in the array.
      //
      if(IdToOcc.count(id)>0) { error = 1; break; }
      IdToOcc[id] = NumV;
      new_v = new ddVertex<WT>(id);

      Data[id] = data;
      VA[data] = new_v;
      IdToPtr[id] = new_v;

      ++NumV;
    }

  // Read-in the edges.
  //
  int done = 0;
  if(!error)
    {
      // Again, skip any space before the edge delimiter is encountered:
      for(is >> str; str != E_BEGIN_DELIM; is >> str);

      while(1)
```

```
      {
        for(int i=0; i<3; ++i)
          {
            if(is.eof() || is.bad()) done = 1;
            is >> str;
            if((str == E_END_DELIM) || done)
              {
                done = 1;
                if(i != 0) error = 1;
                break;
              }
            if(i == 0)      from_id = str.ToInt();  // get first v's id.
            else if(i == 1) to_id   = str.ToInt();  // get secnd v's id.
            else            weight  = str.ToDbl();  // get the weight.
          }
         if(error || done) break;

        // Check if the id's are valid.
        if(IdToPtr.count(from_id)<1 || IdToPtr.count(to_id)<1)
          { error = 1; break; }

        from = IdToPtr[from_id];
        to   = IdToPtr[to_id];

        if(AddEdge(from,to,weight,weight) < 0)
            { error = 1; break; }
      }

    // Re-assign the ID's if bad ID's were encountered.
    //
    if(!error)
      {
        if(re_id)
          {
            for(it = VA.begin(); it!=VA.end(); ++it)
                (($\ast$it).second)->ID = IdToOcc[(($\ast$it).second)->ID];
          }
        return 1;
      }
    //
  }

```

4.18.3.27 template<class DT, class WT = double> ddGraph<DT, WT>::MAKE_IO_STREAM_OPERATORS (DDG)

4.18.3.28 template<class DT, class WT> int ddGraph< DT, WT >::NextFreeId () const [inline]

Definition at line 213 of file ddgraph.inc.

```
{ return IDG.Peek(); }

```

4.18.3.29 template<class DT, class WT> int ddGraph< DT, WT >::NumEdges () const [inline]

Definition at line 163 of file ddgraph.inc.

```
{ return NumE; }

```

4.18.3.30 template<class DT, class WT> int ddGraph< DT, WT >::NumVertices () const [inline]

Definition at line 159 of file ddgraph.inc.

Referenced by ConstraintManager::Number(), ConstraintManager::clearAllConstraints(), ConstraintManager::clearConstraints(), and ConstraintManager::setMolecule().

```
{ return NumV; }

```

4.18.3.31 template<class DT, class WT = double> int ddGraph<DT, WT>::OutDegree (const ddVertex< WT > * *vert*) const [private]

4.18.3.32 template<class DT, class WT> int ddGraph< DT, WT >::OutDegree (const DT & *x*) const [inline]

Definition at line 189 of file ddgraph.inc.

```
{
  if(!HasVertex(x)) return VertexNotInG;
  ddVertex<WT>* vert = VA[x];
  return vert->OutDegree();

```

4.18.3.33 template<class DT, class WT> int ddGraph< DT, WT >::Output (ostream & *os*) const [inline]

Definition at line 489 of file ddgraph.inc.

```
{
  VertexArray::const_iterator it;
  // Print the ID and data of all the vertices:
  //
  os << V_BEGIN_DELIM << '\n';
  for(it = VA.begin(); it!=VA.end(); ++it)
      os << '\t' << ((*it).second)->ID << '\t' << ($\ast$it).first << '\n';
  os << V_END_DELIM << '\n';

  // Print the edges of the vertices
  //
  os << E_BEGIN_DELIM << '\n';
  for(it = VA.begin(); it!=VA.end(); ++it)
    {
      if(($\ast$it).second) (($\ast$it).second)->PrintOutEdges(os);
    }

```

4.18.3.34 template<class DT, class WT = double> void ddGraph<DT, WT>::OutputSimple (ostream & *os*)

4.18.3.35 template<class DT, class WT> int ddGraph< DT, WT >::RemEdge (ddVertex< WT > ∗ *from*, ddVertex< WT > ∗ *to*) `[inline, private]`

Definition at line 458 of file ddgraph.inc.

```
{
  if(NumE<=0) return NoEdges;
  if(!from || !to)
      return VertexBad;

  Edge eF, eT;
  eF.V = from; eT.V = to;

  from->OutE.Remove(eT);
  to->InE.Remove(eF);
  return --NumE;

```

4.18.3.36 template<class DT, class WT> int ddGraph< DT, WT >::RemEdge (const DT & *from*, const DT & *to*) `[inline]`

Definition at line 473 of file ddgraph.inc.

```
{
  ddVertex<WT> * FromV, * ToV;

  if(NumV<=0) return NoVertices;
  if(NumE<=0) return NoEdges;

  if(!HasVertex(from) || !HasVertex(to))
      return VertexNotInG;

  FromV = VA[from];
  ToV = VA[to];
  return RemEdge(FromV,ToV);

```

4.18.3.37 template<class DT, class WT> void ddGraph< DT, WT >::RemEdges (const DT & *x*) `[inline]`

Definition at line 413 of file ddgraph.inc.

```
{
  if(!HasVertex(x))
    { cerr << "ddGraph::Remdges\n"; raise(SIGQUIT); }

```

4.18.3.38 template<class DT, class WT> void ddGraph< DT, WT >::RemInEdges (const DT & *x*) `[inline]`

Definition at line 404 of file ddgraph.inc.

```
{
  if(!HasVertex(x))
    { cerr << "ddGraph::RemInEdges\n"; raise(SIGQUIT); }
```

4.18.3.39 template<class DT, class WT> void ddGraph< DT, WT >::RemOutEdges (const DT & *x*) [inline]

Definition at line 395 of file ddgraph.inc.

```
{
  if(!HasVertex(x))
    { cerr << "ddGraph::RemOutEdges\n"; raise(SIGQUIT); }
```

4.18.3.40 template<class DT, class WT> int ddGraph< DT, WT >::RemVertex (const DT & *x*) [inline]

Definition at line 422 of file ddgraph.inc.

Referenced by ConstraintManager::remove().

```
{
  if(NumV <= 0) return NoVertices;

  // ddVertex<WT> doesn't exist in the graph.
  if(!HasVertex(x))
      return VertexNotInG;

  ddVertex<WT>* vert = VA[x];
  if(!vert) return VertexBad;

  //
  // Clear all its edges and the edges it shares with other vertices.
  //
  NumE -= vert->Degree();

  //
  // Remove the vertex from the array.
  //
  VA.erase(x);
  Data.erase(vert->ID);

  //
  // Recycle this vertex's ID
  //
  IDG.Free(vert->ID);

  //
  // And finally, delete the vertex from existence.
  //
  delete vert;

  return --NumV;
```

4.18.3.41 template<class DT, class WT> void ddGraph< DT, WT >::SetWeight (ddVertex< WT > * *from*, ddVertex< WT > * *to*, const WT & *fwd_weight*, const WT & *bwd_weight*) `[inline, private]`

Definition at line 343 of file ddgraph.inc.

```
{
  if(!from || !to)
    { cerr << "ddGraph::SetWeight\n"; raise(SIGQUIT); }
```

4.18.3.42 template<class DT, class WT> void ddGraph< DT, WT >::SetWeight (const DT & *from*, const DT & *to*, const WT & *weight*) `[inline]`

Definition at line 362 of file ddgraph.inc.

```
{ SetWeight(from,to,weight,weight); }
```

4.18.3.43 template<class DT, class WT> void ddGraph< DT, WT >::SetWeight (const DT & *from*, const DT & *to*, const WT & *fwd_weight*, const WT & *bwd_weight*) `[inline]`

Definition at line 353 of file ddgraph.inc.

Referenced by SetWeight().

```
{
  if(!HasEdge(from,to))
    { cerr << "ddGraph::SetWeight\n"; raise(SIGQUIT); }
```

4.18.3.44 template<class DT, class WT> const WT & ddGraph< DT, WT >::operator() (const DT & *from*, const DT & *to*) const `[inline]`

Definition at line 391 of file ddgraph.inc.

```
{ return GetWeight(from,to); }
```

4.18.3.45 template<class DT, class WT> const ddGraph< DT, WT > & ddGraph< DT, WT >::operator= (const sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 627 of file ddgraph.inc.

```
{ Copy(sdg); return *this; }
```

4.18.3.46 template<class DT, class WT> const ddGraph< DT, WT > & ddGraph< DT, WT >::operator= (const ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 623 of file ddgraph.inc.

```
{ Copy(ddg); return *this; }
```

4.18.4 Friends And Related Function Documentation

4.18.4.1 template<class DT, class WT = double> istream& InputBinary (istream & *is*, ddGraph< DT, WT > & *ddg*) `[friend]`

Definition at line 632 of file ddgraph.inc.

```
{
  sdGraph<DT,WT> G;
  if(InputBinary(is,G).bad())
     return is;
  ddg = G;
  return is;
```

4.18.4.2 template<class DT, class WT = double> ostream& OutputBinary (ostream & *os*, const ddGraph< DT, WT > & *ddg*) `[friend]`

Definition at line 642 of file ddgraph.inc.

```
{
  sdGraph<DT,WT> G(ddg);
  return OutputBinary(os,G);
```

4.18.4.3 template<class DT, class WT = double> ostream& operator (ostream & *os*, const ddGraph< DT, WT > & *ddg*) `[friend]`

4.18.4.4 template<class DT, class WT = double> istream& operator>= (istream & *is*, ddGraph< DT, WT > & *ddg*) `[friend]`

Definition at line 649 of file ddgraph.inc.

```
{ return IntputBinary(is,ddg); }
```

4.18.4.5 template<class DT, class WT = double> class sdGraph `[friend]`

Definition at line 73 of file ddgraph.h.

4.18.5 Member Data Documentation

4.18.5.1 template<class DT, class WT = double> DataArray ddGraph<DT, WT>::Data `[private]`

Definition at line 192 of file ddgraph.h.

4.18.5.2 template<class DT, class WT = double> IdGen ddGraph<DT, WT>::IDG `[private]`

Definition at line 177 of file ddgraph.h.

4.18.5.3 template<class DT, class WT = double> int ddGraph<DT, WT>::NumE [private]

Definition at line 181 of file ddgraph.h.

4.18.5.4 template<class DT, class WT = double> int ddGraph<DT, WT>::NumV [private]

Definition at line 179 of file ddgraph.h.

4.18.5.5 template<class DT, class WT = double> VertexArray ddGraph<DT, WT>::VA [private]

Definition at line 187 of file ddgraph.h.

The documentation for this class was generated from the following files:

- **ddgraph.h**
- **ddgraph.inc**

4.19 ddVertex Class Template Reference

`#include <ddvertex.h>`

Public Types

- typedef set<**ddEdge**<WT> > **EdgeSet**

Public Methods

- **ddVertex** (int id)
- **ddVertex** ()
- ~**ddVertex** ()
- int **HaveSelfLoop** () const
- int **OutDegree** () const
- int **InDegree** () const
- int **Degree** () const
- const double& **GetWeightTo** (ddVertex< WT > *v) const
- const double& **GetWeightFrom** (ddVertex< WT > *v) const
- void **SetWeightTo** (ddVertex< WT > *v, const WT &weight)
- void **SetWeightFrom** (ddVertex< WT > *v, const WT &weight)
- void **ChangeWeightTo** (ddVertex< WT > *v, const WT &delta)
- void **ChangeWeightFrom** (ddVertex< WT > *v, const WT &delta)
- int **RemOutEdges** ()
- int **RemInEdges** ()
- int **RemEdges** ()
- void **PrintOutEdges** (ostream &os) const
- void **PrintInEdges** (ostream &os) const
- void **PrintEdges** (ostream &os) const
- int **Compare** (const ddVertex< WT > &v) const
- **MAKECOMPARISONS_WITH_COMPARE** (ddVertex< WT >)

Public Attributes

- int **ID**
- **EdgeSet OutE**
- **EdgeSet InE**

Friends

- ostream& **operator** (ostream &os, const ddVertex< WT > &vert)

template<class WT = double> class ddVertex

4.19.1 Member Typedef Documentation

4.19.1.1 template<class WT = double> typedef set< ddEdge< WT > > ddVertex<WT>::EdgeSet

Definition at line 68 of file ddvertex.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 template<class WT> ddVertex< WT >::ddVertex<WT> (int *id*) `[inline]`

Definition at line 12 of file ddvertex.inc.

```
{
  ID = id>=0 ? id : -1;

```

4.19.2.2 template<class WT> ddVertex< WT >::ddVertex<WT> () `[inline]`

Definition at line 18 of file ddvertex.inc.

```
{
  ID = -1;

```

4.19.2.3 template<class WT> ddVertex< WT >::~ddVertex<WT> () `[inline]`

Definition at line 177 of file ddvertex.inc.

```
{
  RemEdges();

```

4.19.3 Member Function Documentation

4.19.3.1 template<class WT> void ddVertex< WT >::ChangeWeightFrom (ddVertex< WT > * *v*, const WT & *delta*) `[inline]`

Definition at line 85 of file ddvertex.inc.

Referenced by ddGraph::ChangeWeight().

```
{
  ddEdge<WT> e; e.V = v;
  if(!v || InE.count(e)<1) raise(SIGQUIT);
  ((ddEdge<WT>&)(*(InE.find(e)))).W += delta;

```

4.19.3.2 template<class WT> void ddVertex< WT >::ChangeWeightTo (ddVertex< WT > * *v*, const WT & *delta*) `[inline]`

Definition at line 77 of file ddvertex.inc.

Referenced by ddGraph::ChangeWeight().

```
{
  ddEdge<WT> e; e.V = v;
  if(!v || OutE.count(e)<1) raise(SIGQUIT);
  ((ddEdge<WT>&)(*(OutE.find(e)))).W += delta;

```

4.19.3.3 template<class WT = double> int ddVertex<WT>::Compare (const ddVertex< WT > & *v*) const [inline]

Definition at line 100 of file ddvertex.h.

```
                  { return (ID - v.ID); }

```

4.19.3.4 template<class WT> int ddVertex< WT >::Degree () const [inline]

Definition at line 41 of file ddvertex.inc.

Referenced by ddGraph::RemVertex().

```
{ return OutDegree() + InDegree() - HaveSelfLoop(); }

```

4.19.3.5 template<class WT> const double & ddVertex< WT >::GetWeightFrom (ddVertex< WT > ∗ *v*) const [inline]

Definition at line 53 of file ddvertex.inc.

```
{
  ddEdge<WT> e; e.V = v;
  if(!v || InE.count(e)<1) raise(SIGQUIT);
  return (*(InE.find(e))).W;

```

4.19.3.6 template<class WT> const double & ddVertex< WT >::GetWeightTo (ddVertex< WT > ∗ *v*) const [inline]

Definition at line 45 of file ddvertex.inc.

```
{
  ddEdge<WT> e; e.V = v;
  if(!v || OutE.count(e)<1) raise(SIGQUIT);
  return (*(OutE.find(e))).W;

```

4.19.3.7 template<class WT> int ddVertex< WT >::HaveSelfLoop () const [inline]

Definition at line 24 of file ddvertex.inc.

Referenced by Degree().

```
{
  if(OutE.size()==0) return 0;
  for(EdgeSet::const_iterator it = OutE.begin(); it!=OutE.end(); ++it)
      if(($\ast$it).V == this) return 1;
  return 0;

```

4.19.3.8 template<class WT> int ddVertex< WT >::InDegree () const [inline]

Definition at line 37 of file ddvertex.inc.

Referenced by sdGraph::Copy(), Degree(), ddGraph::Degree(), and ddGraph::InDegree().

```
{ return InE.size(); }
```

4.19.3.9 template<class WT = double> ddVertex<WT>::MAKECOMPARISONS_WITH_COMPARE (ddVertex< WT >)

4.19.3.10 template<class WT> int ddVertex< WT >::OutDegree () const [inline]

Definition at line 33 of file ddvertex.inc.

Referenced by sdGraph::Copy(), Degree(), ddGraph::Degree(), and ddGraph::OutDegree().

```
{ return OutE.size(); }
```

4.19.3.11 template<class WT> void ddVertex< WT >::PrintEdges (ostream & *os*) const [inline]

Definition at line 117 of file ddvertex.inc.

```
{ PrintOutEdges(os); PrintInEdges(os); }
```

4.19.3.12 template<class WT> void ddVertex< WT >::PrintInEdges (ostream & *os*) const [inline]

Definition at line 105 of file ddvertex.inc.

Referenced by PrintEdges().

```
{
  EdgeSet::const_iterator it;
  ddEdge<WT> edge;
  for(it = InE.begin(); it!=InE.end(); ++it)
    {
      edge = *it;
      os << '\t' << edge.V->ID << '\t' << ID << '\t' << edge.W << '\n';
    }
```

4.19.3.13 template<class WT> void ddVertex< WT >::PrintOutEdges (ostream & *os*) const [inline]

Definition at line 93 of file ddvertex.inc.

Referenced by PrintEdges().

```
{
  EdgeSet::const_iterator it;
  ddEdge<WT> edge;
  for(it = OutE.begin(); it!=OutE.end(); ++it)
    {
      edge = *it;
      os << '\t' << ID << '\t' << edge.V->ID << '\t' << edge.W << '\n';
    }

```

4.19.3.14 template<class WT> int ddVertex< WT >::RemEdges () [inline]

Definition at line 173 of file ddvertex.inc.

Referenced by ddGraph::RemEdges(), and ~ddVertex().

```
{ return RemOutEdges() + RemInEdges(); }

```

4.19.3.15 template<class WT> int ddVertex< WT >::RemInEdges () [inline]

Definition at line 155 of file ddvertex.inc.

Referenced by RemEdges(), and ddGraph::RemInEdges().

```
{
  ddEdge<WT> t;
  int n = InE.size();
  t.V = this;
  // Remove all our neighbors' outgoing edges to us.
  //
  for(EdgeSet::iterator it = InE.begin(); it!=InE.end(); ++it)
      if(($\ast$it).V)
          ($\ast$it).V->OutE.erase(t);

  // Remove all our incoming edges.
  //
  InE.clear();
  return n;

```

4.19.3.16 template<class WT> int ddVertex< WT >::RemOutEdges () [inline]

Definition at line 132 of file ddvertex.inc.

Referenced by RemEdges(), and ddGraph::RemOutEdges().

```
{
  //
  // This is a one sided removal!  Otherwise it may cause the IterateOver()
  // function to break: we can't remove the edges from the array we've
  // been called from, just those from the vertices we point to.
  //
  ddEdge<WT> f;
  int n = OutE.size();
  f.V = this;
  // Remove all our neighbors' incoming edges from us.
  //
```

```
00144     for(EdgeSet::iterator it = OutE.begin(); it!=OutE.end(); ++it)
00145         if(($\ast$it).V)
00146             ($\ast$it).V->InE.erase(f);
00147
00148     // Remove all our edges.
00149     //
00150     OutE.clear();
00151     return n;
00152
```

4.19.3.17 template<class WT> void ddVertex< WT >::SetWeightFrom (ddVertex< WT > ∗ *v*, const WT & *weight*) `[inline]`

Definition at line 69 of file ddvertex.inc.

Referenced by ddGraph::SetWeight().

```
00070   {
00071     ddEdge<WT> e; e.V = v;
00072     if(!v || InE.count(e)<1) raise(SIGQUIT);
00073     ((ddEdge<WT>&)(*(InE.find(e)))).W = weight;
00074
```

4.19.3.18 template<class WT> void ddVertex< WT >::SetWeightTo (ddVertex< WT > ∗ *v*, const WT & *weight*) `[inline]`

Definition at line 61 of file ddvertex.inc.

Referenced by ddGraph::SetWeight().

```
00062   {
00063     ddEdge<WT> e; e.V = v;
00064     if(!v || OutE.count(e)<1) raise(SIGQUIT);
00065     ((ddEdge<WT>&)(*(OutE.find(e)))).W = weight;
00066
```

4.19.4 Friends And Related Function Documentation

4.19.4.1 template<class WT = double> ostream& operator (ostream & *os*, const ddVertex< WT > & *vert*) `[friend]`

4.19.5 Member Data Documentation

4.19.5.1 template<class WT = double> int ddVertex<WT>::ID

Definition at line 65 of file ddvertex.h.

4.19.5.2 template<class WT = double> EdgeSet ddVertex<WT>::InE

Definition at line 70 of file ddvertex.h.

4.19.5.3 template<class WT = double> EdgeSet ddVertex<WT>::OutE

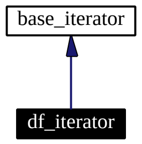
Definition at line 69 of file ddvertex.h.

The documentation for this class was generated from the following files:

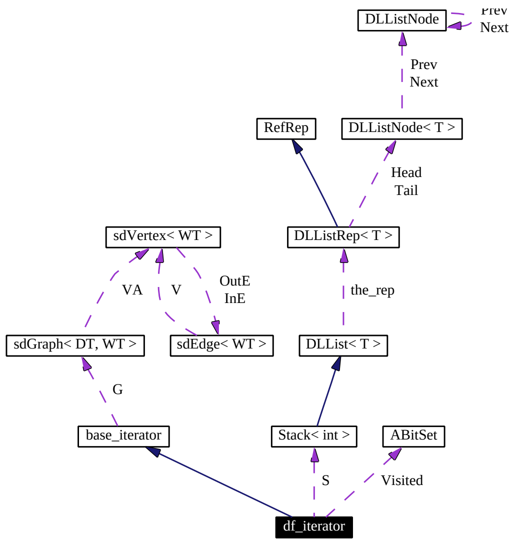
- **ddvertex.h**
- **ddvertex.inc**

4.20 df_iterator Class Reference

Inheritance diagram for df_iterator



Collaboration diagram for df_iterator:



Public Methods

- **df_iterator** (int reverse=0)
- **df_iterator** (const df_iterator &dfi)
- **∼df_iterator** ()
- const df_iterator& **operator=** (const df_iterator &dfi)
- void **reset** ()
- **operator int** () const
- const **base_iterator**& **operator++** ()
- const **base_iterator**& **operator−** ()
- int **operator** ∗ () const

Protected Methods

- **df_iterator** (**sdGraph**< DT, WT > ∗g, int vndx, int reverse=0)

Private Attributes

- **ABS**∗ **Visited**
- **Stack**<int> **S**

Friends

- class **sdGraph< DT, WT >**

4.20.1 Constructor & Destructor Documentation

4.20.1.1 df_iterator::df_iterator (sdGraph< DT, WT > * *g*, int *vndx*, int *reverse* = 0) `[inline, protected]`

Definition at line 114 of file sdgiter.inc.

```
00115                          : base_iterator(g,vndx,reverse)
00116
```

4.20.1.2 df_iterator::df_iterator (int *reverse* = 0) `[inline]`

Definition at line 119 of file sdgiter.inc.

```
00119                                        : base_iterator(reverse)
00120
```

4.20.1.3 df_iterator::df_iterator (const df_iterator & *dfi*) `[inline]`

Definition at line 123 of file sdgiter.inc.

```
00123                                               : base_iterator(dfi)
00124                  {
00125                    Visited = new ABS(*(dfi.Visited));
00126                    S = dfi.S;
00127
```

4.20.1.4 df_iterator::~df_iterator () `[inline]`

Definition at line 129 of file sdgiter.inc.

```
00130                  { delete Visited; }
00131
```

4.20.2 Member Function Documentation

4.20.2.1 int df_iterator::operator * () const `[inline, virtual]`

Reimplemented from **base_iterator** (p. 65).

Definition at line 204 of file sdgiter.inc.

```
00204                                   { return vid; }
00205
```

4.20.2.2 df_iterator::operator int () const [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 156 of file sdgiter.inc.

```
{ return (vid!=-1); }
```

4.20.2.3 const base_iterator & df_iterator::operator++ () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 159 of file sdgiter.inc.

```
{
  int visited, nbrid;

  // We can stop if we've visited all the vertices:
  if(Visited->IsAllSet()) vid = -1;
  if(vid==-1) return *this;

  const sdEdge<WT>* edges = rev ? G->VA[vid]->InE :
                                  G->VA[vid]->OutE;
  int n = rev ? G->VA[vid]->NumI : G->VA[vid]->NumO;

  // Push all the unvisited neighbors onto the stack:
  for(int i=0; i<n; ++i)
    {
      nbrid = edges[i].V->ID;
      if(!Visited->IsSet(nbrid))
          S.Push(nbrid);
    }

  // Skip all the visited vertices in the stack and
  // assign 'vid' to the first unvisited vertex:
  vid = -1;
  visited = 1;
  while(!S.IsEmpty() && visited)
    {
      vid = S.Pop();
      if((visited = Visited->IsSet((uint) vid)))
          vid = -1;
    }
```

4.20.2.4 const base_iterator & df_iterator::operator– () [inline]

Definition at line 196 of file sdgiter.inc.

```
{
  rev = rev ? 0 : 1;
  ++($\ast$this);
  rev = rev ? 0 : 1;
  return *this;
```

4.20.2.5 const df_iterator & df_iterator::operator= (const df_iterator & *dfi*) [inline]

Definition at line 132 of file sdgiter.inc.

```
{
  if(this!=&dfi)
    {
      G=dfi.G; vstart=dfi.vstart; vid=dfi.vid; rev=dfi.rev;
      S = dfi.S;
      delete Visited;
      Visited = new ABS(*(dfi.Visited));
    }

```

4.20.2.6 void df_iterator::reset () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 144 of file sdgiter.inc.

```
{
  if(bad()) vid=-1;
  else
    {
      S.Clear();
      vid = vstart;
      Visited->ClearAll();
      Visited->Set((uint) vid);
    }

```

4.20.3 Friends And Related Function Documentation

4.20.3.1 class sdGraph [friend]

Reimplemented from **base_iterator** (p. 66).

Definition at line 111 of file sdgiter.inc.

4.20.4 Member Data Documentation

4.20.4.1 Stack< int > df_iterator::S [private]

Definition at line 211 of file sdgiter.inc.

4.20.4.2 ABS * df_iterator::Visited [private]

Definition at line 208 of file sdgiter.inc.

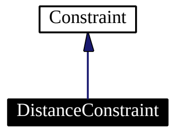
The documentation for this class was generated from the following file:

- **sdgiter.inc**

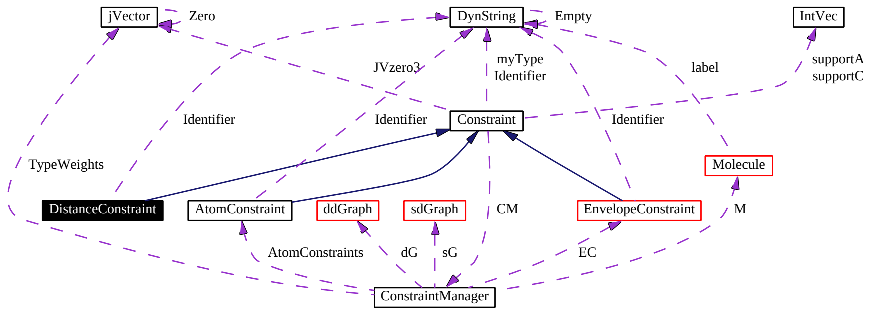
4.21 DistanceConstraint Class Reference

`#include <distance_constraint.h>`

Inheritance diagram for DistanceConstraint



Collaboration diagram for DistanceConstraint:



Public Methods

- **DistanceConstraint** (**ConstraintManager** &cm)
- **DistanceConstraint** (**ConstraintManager** &cm, int atom_i, int atom_j, double distance, double var)
- ∼**DistanceConstraint** ()
- **jVector dStress** (int k, const **jVector** &pos) const
- int **Compare** (const DistanceConstraint &dc) const
- **MAKECOMPARISONS_WITH_COMPARE** (DistanceConstraint)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const

Static Public Attributes

- **DynString Identifier** = "atomdis:"

Protected Methods

- double **computeStress** () const
- double **computeStress** (int k, const **jVector** &pos) const
- double **computeAbsRes** (void) const
- int **Read** (istream &is)
- int **ReadBinary** (istream &is)

  Same scenerio as **Read***()* (p. 157) *only binary version.*

- void **PreSetup** ()

Protected Attributes

- int **envelopeID**
- double **measured**
- double **variance**
- double **stdev**

4.21.1 Detailed Description

//

distance_constraint.h

This is the first constraint type for GRAPE. It constrains the distance between two atoms.

//

Definition at line 20 of file distance_constraint.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 DistanceConstraint::DistanceConstraint (ConstraintManager & *cm*)

Definition at line 3 of file distance_constraint.C.

```
: Constraint(cm,IntVec(2,-1))
{
  // this is an unititialized distance constraint.  The two atoms
  // and the mesured and variance values must be added.

  measured = 0.0;
  variance = INF;

```

4.21.2.2 DistanceConstraint::DistanceConstraint (ConstraintManager & *cm*, int *i*, int *j*, double *distance*, double *var*)

Takes two indices of the atoms, the measured distance between them and the variance associated with this measurement. A molecule, M, is also given to serve as a "template" structure from which the object determines the number (and types) of atoms in the molecule of interest.

Definition at line 13 of file distance_constraint.C.

```
: Constraint(cm, cat( IntVec(1,((i<j) ? i : j)),
                      IntVec(1,((i<j) ? j : i))))
{
measured = distance;
variance = var;
if (variance<gwNumerics::MinDistVariance)
  {gwNumerics::MinDistVariance = variance;}

```

4.21.2.3 DistanceConstraint::~DistanceConstraint () `[inline]`

Definition at line 36 of file distance_constraint.h.

```
                              {}

```

4.21.3 Member Function Documentation

4.21.3.1 int DistanceConstraint::Compare (const DistanceConstraint & *dc*) const [inline]

Definition at line 84 of file distance_constraint.h.

```
{
  if(supportA[0] == dc.supportA[0] && supportA[1] == dc.supportA[1])
    return 0;
  if(supportA[0]<dc.supportA[0])
    return -1;
  if(supportA[0]>dc.supportA[0])
    return 1;
  if(supportA[1]<dc.supportA[1])
    return -1;
  return 1;

```

4.21.3.2 DistanceConstraint::MAKECOMPARISONS_WITH_COMPARE (DistanceConstraint)

4.21.3.3 int DistanceConstraint::Output (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 235 of file distance_constraint.C.

```
{
  os << supportA[0] << ' ' << supportA[1] << ' '
     << measured << ' ' << variance;
  return 2;

```

4.21.3.4 int DistanceConstraint::OutputBinary (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 250 of file distance_constraint.C.

```
{
  os <= supportA <= measured <= variance;
  return 2;

```

4.21.3.5 void DistanceConstraint::PreSetup (void) [protected, virtual]

The setup function is called if the uninialized constraint method is used to create the constraint to allow different data representations within the constraint. Without this, only data read in from the input file will be available to the member functions. For simple constraints, this is really not very useful. However for multi-atom constraints that need to be a little 'smarter' about data representations, the Setup function will be helpful.

Reimplemented from **Constraint** (p. 85).

Definition at line 25 of file distance_constraint.C.

```
{
  //  CString = Identifier;
  stdev = sqrt(variance);
```

4.21.3.6 int DistanceConstraint::Read (istream & *is*) [protected, virtual]

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 157) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 157) function, the derived class should *not* log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented from **Constraint** (p. 85).

Definition at line 242 of file distance_constraint.C.

```
{
  is >> supportA[0] >> supportA[1] >> measured >> variance;
  if (variance<gwNumerics::MinDistVariance)
    {gwNumerics::MinDistVariance = variance;}
```

4.21.3.7 int DistanceConstraint::ReadBinary (istream & *is*) [protected, virtual]

Same scenerio as **Read**() (p. 157) only binary version.

Reimplemented from **Constraint** (p. 85).

Definition at line 256 of file distance_constraint.C.

```
{
  is >= supportA[0] >= supportA[1] >= measured >= variance;
  if (variance<gwNumerics::MinDistVariance)
    {gwNumerics::MinDistVariance = variance;}
```

4.21.3.8 double DistanceConstraint::computeAbsRes (void) const [protected, virtual]

returns abs residual as a Z score, if supported, constraint overwrites

Reimplemented from **Constraint** (p. 89).

Definition at line 227 of file distance_constraint.C.

```
{
  int i=supportA[0], j=supportA[1];
  double calc = euclid(getMolecule()(i),getMolecule()(j));
  return fabs((calc-measured)/stdev);
```

4.21.3.9 double DistanceConstraint::computeStress (int *k*, const jVector & *pos_k*) const [protected, virtual]

Returns the stress on this constraint after "pretending" the atom with index a' moved to position pos_a'. This also must be overwritten by all constraints that inherit from this class

Reimplemented from **Constraint** (p. 89).

Definition at line 56 of file distance_constraint.C.

```
00057 {
00058   double calc;  // calculated (current model) distance between i and j.
00059   int i=supportA[0], j=supportA[1];
00060
00061   // If the atom that has moved does not include either of the two
00062   // atoms this constraint is defined over, then just default to reporting
00063   // how stressed we are about our two atoms.
00064   //
00065   if(k != i && k != j){
00066     cerr << "*** Called computeStress("<<k<<","<<pos_k<<") on dist constraint over atoms " ;
00067     cerr << i << " and " << j << "***" << endl;
00068     DIE("");
00069     return computeStress();
00070   }
00071   // Figure out which atom has moved and report our new satisfaction:
00072   //
00073   if(k == i)
00074       calc = euclid(pos_k, getMolecule()(j));
00075   else
00076       calc = euclid(getMolecule()(i), pos_k);
00077
00078   // Our new satisfaction is the difference between the measured and the
00079   // predicted distance:
00080   //
00081 //    cerr << "ORDER cmputstr-pos: " << Label() << ";calc = " << setprecision(50) << calc << endl;
00082
00083   double R = (calc-measured)/stdev;
00084 //   if (R>DistanceConstraint::MaxAbsResidual)
00085 //     {DistanceConstraint::MaxAbsResidual = R;}
00086
```

4.21.3.10 double DistanceConstraint::computeStress () const [protected, virtual]

One of the only functions that *must* be overwritten by subclasses. Returns a value reflecting how dissatisfied this constraint is with the current molecule residing in CM'.

Reimplemented from **Constraint** (p. 89).

Definition at line 36 of file distance_constraint.C.

Referenced by computeStress().

```
00037 {
00038   int i=supportA[0], j=supportA[1];
00039
00040   // Find the calculated distance in the model
00041   double calc = euclid(getMolecule()(i),getMolecule()(j));
00042 //   cerr << "ORDER cmputstr: " << Label() << ";calc = " << setprecision(50) << calc << endl;
00043 //   cerr << "ORDER cmputstr: " << Label() << ";meas = " << setprecision(50) << measured << endl;
00044   double R = (calc-measured)/stdev;
00045 //   cerr << "ORDER cmputstr: " << Label() << ";R' = " << setprecision(50) << R << endl;
00046   //  R = R/stdev;
00047 //   if (R>DistanceConstraint::MaxAbsResidual)
```

```
//     {DistanceConstraint::MaxAbsResidual = R;}

```

4.21.3.11 jVector DistanceConstraint::dStress (int *k*, const jVector & *pos_k*) const [virtual]

asking constraint number k for its 12 element dStress vector we pass in pos_a vector – corresponding to the atom position if the constraint is an atom, and corresponding to the position of the atom k in question if the call is to a non-atom constraint. EG for a distance constraint atom 12 needs the dStress from the d-constraint, so it will call dStress on the d-constraint(12, atom_pos) so the d_constr can answer the question if atom 12 were in atom_pos, I would say my dStress is blah.

Reimplemented from **Constraint** (p. 89).

Definition at line 171 of file distance_constraint.C.

```
{
  // A HACK and a TEMPORARY FIX to an inherent logic problems in this
  // set of functions.  come see me w/ questions -jd

   int i=supportA[0], j=supportA[1];

  if(k != i && k != j)
    {
      cerr << "Fatal error: distance_constraint.C, dStress("<< k \
           <<", pos) not legal on atom " << k << endl;
      exit(1);
    }

  // Figure out which atom has moved and report our new satisfaction:
  //
  double calc;
  if(k == i)
      calc = euclid(pos_k, getMolecule()(j));
  else
      calc = euclid(getMolecule()(i), pos_k);

//   cerr << "ORDER dstress: " << Label() << ";calc = " << setprecision(50) << calc << endl;

  jVector dist(3,0.0);
  //  jVector return_vect(12,0.0);
  jVector return_vect(3,0.0);
  double R = (calc-measured)/stdev;

  //  cerr << "R=" << R << endl;

  if(k==i)
    dist = ((pos_k - getMolecule()(j))/calc/stdev);
  else if(k==j)
    dist = ((pos_k - getMolecule()(i))/calc/stdev);
  else {
    DIE("Error in distance dstress call.");
    return return_vect;
  }

```

4.21.4 Member Data Documentation

4.21.4.1 DynString DistanceConstraint::Identifier = "atomdis:" [static]

Reimplemented from **Constraint** (p. 95).

Definition at line 100 of file globals.C.

4.21.4.2 int DistanceConstraint::envelopeID [protected]

Definition at line 55 of file distance_constraint.h.

4.21.4.3 double DistanceConstraint::measured [protected]

Definition at line 71 of file distance_constraint.h.

4.21.4.4 double DistanceConstraint::stdev [protected]

Definition at line 73 of file distance_constraint.h.

4.21.4.5 double DistanceConstraint::variance [protected]

Definition at line 72 of file distance_constraint.h.

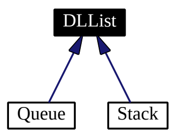
The documentation for this class was generated from the following files:

- **distance_constraint.h**
- **distance_constraint.C**
- **globals.C**

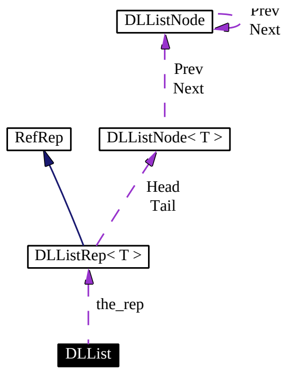
4.22 DLList Class Template Reference

`#include <dllist.h>`

Inheritance diagram for DLList



Collaboration diagram for DLList:



Public Methods

- **DLList** ()
- **DLList** (const DLList< T > &dll)
- ~**DLList** ()
- int **IsEmpty** () const
- **operator int** () const
- void **Clear** ()
- int **Number** () const
- bool **AddToFront** (const T &data)
- bool **AddToBack** (const T &data)
- bool **AddToFront** (const DLList< T > &dll)
- bool **AddToBack** (const DLList< T > &dll)
- bool **Add** (const T &data)
- bool **Add** (const DLList< T > &dll)
- const DLList<T>& **operator+=** (const DLList< T > &dll)
- T **RemoveFromFront** ()
- T **RemoveFromBack** ()
- T **Remove** ()
- const T& **Get** (int i) const
- T& **Get** (int i)
- const T& **PeekFront** () const
- T& **PeekFront** ()
- const T& **PeekBack** () const

- T& **PeekBack** ()
- T **RemoveLocation** (int i)
- T **Remove** (const T &data)
- int **operator==** (const DLList< T > &dll) const
- int **operator!=** (const DLList< T > &dll) const
- T& **Find** (const T &data)
- const T& **Find** (const T &data) const
- int **FindLocation** (const T &data) const
- int **In** (const T &data) const
- ostream& **Print** (ostream &os) const
- DLList<T> **UniqueCopy** () const
- const DLList<T>& **operator=** (const DLList< T > &dll)
- DLList<T>::**Iterator GetIterator** ()
- DLList<T>::**ConstIterator GetConstIterator** () const

Private Attributes

- **DLListRep**<T>* **the_rep**

Friends

- class **Iterator**
- class **ConstIterator**
- ostream& **operator** (ostream &os, const DLList< T > &dll)
- istream& **InputBinary** (istream &is, DLList< T > &dll)
- istream& **operator>=** (istream &is, DLList< T > &dll)
- ostream& **OutputBinary** (ostream &os, const DLList< T > &dll)
- ostream& **operator** (ostream &os, const DLList< T > &dll)

template<class T> class DLList

4.22.1 Constructor & Destructor Documentation

4.22.1.1 template<class T> DLList< T >::DLList<T> () `[inline]`

Definition at line 459 of file dllist.inc.

```
{ the_rep = new DLListRep<T>(); }

```

4.22.1.2 template<class T> DLList< T >::DLList<T> (const DLList< T > & *dll*) `[inline]`

Definition at line 463 of file dllist.inc.

```
{
  the_rep = dll.the_rep;
  if(the_rep) the_rep->Reference();

```

4.22.1.3 template<class T> DLList< T >::~DLList<T> () [inline]

Definition at line 470 of file dllist.inc.

```
{
  if(the_rep)
      the_rep->Dereference();
  the_rep = 0;
```

4.22.2 Member Function Documentation

4.22.2.1 template<class T> bool DLList< T >::Add (const DLList< T > & *dll*) [inline]

Definition at line 162 of file dllist.inc.

```
{ return AddToBack(dll); }
```

4.22.2.2 template<class T> bool DLList< T >::Add (const T & *x*) [inline]

Reimplemented in **Queue** (p. 401).

Definition at line 158 of file dllist.inc.

Referenced by Queue::operator+=(), and Molecule::readFasta().

```
{ return AddToBack(data); }
```

4.22.2.3 template<class T> bool DLList< T >::AddToBack (const DLList< T > & *dll*) [inline]

Definition at line 127 of file dllist.inc.

```
{
  if(!dll.the_rep->Head) return false;
  if(IsEmpty()) { *this = dll.UniqueCopy(); return false; }

  DLListNode<T> * Hop, * New, * ExtPoint, * Previous;

  Hop      = dll.the_rep->Head;
  New      = new DLListNode<T>(Hop->Data);
  ExtPoint = New;

  Hop = Hop->Next;
  while(Hop)
    {
      Previous = New;
      New = new DLListNode<T>(Hop->Data);
      New->Prev = Previous;
      Previous->Next = New;
      Hop = Hop->Next;
    }
```

4.22.2.4 template<class T> bool DLList< T >::AddToBack (const T & *data*) `[inline]`

Definition at line 68 of file dllist.inc.

Referenced by Queue::Add(), Add(), Molecule::DynRead(), InputBinary(), Segments::QSortReduce(), sdGraph::TravLists(), UniqueCopy(), Process::addChild(), jVector::dynread(), IntVec::dynread(), ConstraintManager::getAssocAtoms(), ConstraintManager::getParents(), main(), and operator+=().

```
{
  // Create a new node.
  //
  DLListNode<T> * New = new DLListNode<T>(data);

  // Make the node the new tail of the list.
  //
  if(IsEmpty())
    {
      New->Next = 0;
      New->Prev = 0;
      the_rep->Head = New;
      the_rep->Tail = New;
    }
  else
    {
      the_rep->Tail->Next = New;
      New->Prev  = the_rep->Tail;
      New->Next  = 0;
      the_rep->Tail = New;
    }

```

4.22.2.5 template<class T> bool DLList< T >::AddToFront (const DLList< T > & *dll*) `[inline]`

Definition at line 96 of file dllist.inc.

```
{
  if(!dll.the_rep->Head) return false;
  if(IsEmpty()) { *this = dll.UniqueCopy(); return false; }

  DLListNode<T> * Hop, * New, * NewHead, * Previous;

  Hop     = dll.the_rep->Head;
  New     = new DLListNode<T>(Hop->Data);
  NewHead = New;

  Hop = Hop->Next;
  while(Hop)
    {
      Previous = New;
      New = new DLListNode<T>(Hop->Data);
      New->Prev = Previous;
      Previous->Next = New;
      Hop = Hop->Next;
    }

```

4.22.2.6 template< class T > bool DLList< T >::AddToFront (const T & *data*) [inline]

Definition at line 41 of file dllist.inc.

Referenced by Segments::Add(), Stack::Push(), Process::addChild(), Grid::approx_neighbor_i(), Grid::approx_neighbor_p(), Grid::neighbor_d(), Grid::neighbor_i(), and Grid::neighbor_p().

```
{
  // Create a new node.
  //
  DLListNode<T> * New = new DLListNode<T>(data);

  // Make the node the new head of the list.
  //
  if(IsEmpty())
    {
      New->Next = 0;
      New->Prev = 0;
      the_rep->Head = New;
      the_rep->Tail = New;
    }
  else
    {
      the_rep->Head->Prev = New;
      New->Next  = the_rep->Head;
      New->Prev  = 0;
      the_rep->Head = New;
    }
```

4.22.2.7 template< class T > void DLList< T >::Clear () [inline]

Definition at line 33 of file dllist.inc.

Referenced by Segments::Clear(), InputBinary(), Segments::QSortReduce(), IdGen::Reset(), bf_iterator::reset(), and df_iterator::reset().

```
{ delete the_rep->Head; the_rep->Head=0; the_rep->Tail=0; the_rep->M=0; }
```

4.22.2.8 template< class T > const T & DLList< T >::Find (const T & *data*) const [inline]

Definition at line 360 of file dllist.inc.

```
{ return Get( FindLocation(data) ); }
```

4.22.2.9 template< class T > T & DLList< T >::Find (const T & *data*) [inline]

Definition at line 356 of file dllist.inc.

```
{ return Get( FindLocation(data) ); }
```

4.22.2.10 template<class T> int DLList< T >::FindLocation (const T & *data*) const [inline]

Definition at line 344 of file dllist.inc.

Referenced by Find(), In(), and Remove().

```
{
  DLListNode<T> * Hop = the_rep->Head;
  for(int i=0; Hop; ++i)
    {
      if(Hop->Data == data) return i;
      Hop = Hop->Next;
    }

```

4.22.2.11 template<class T> T & DLList< T >::Get (int *i*) [inline]

Definition at line 252 of file dllist.inc.

```
{
  if((i>=the_rep->M) || (i<0)) i=0;

  DLListNode<T> * Hop;
  if(i<the_rep->M/2)
    {
      Hop = the_rep->Head;
      for(int j=0; j<i; ++j)
          Hop = Hop->Next;
    }
  else
    {
      Hop = the_rep->Tail;
      for(int j=the_rep->M-1; j>i; --j)
          Hop = Hop->Prev;
    }

```

4.22.2.12 template<class T> const T & DLList< T >::Get (int *i*) const [inline]

Definition at line 231 of file dllist.inc.

Referenced by Find(), Segments::GetSeg(), Stack::Peek(), Segments::QSortReduceViol(), and Process::runChild().

```
{
  if((i>=the_rep->M) || (i<0)) i=0;

  DLListNode<T> * Hop;
  if(i<the_rep->M/2)
    {
      Hop = the_rep->Head;
      for(int j=0; j<i; ++j)
          Hop = Hop->Next;
    }
  else
    {
      Hop = the_rep->Tail;
      for(int j=the_rep->M-1; j>i; --j)
          Hop = Hop->Prev;
```

```
00247     }
00248
```

4.22.2.13 template<class T> DLList< T >::ConstIterator DLList<T>::GetConstIterator<T> () const [inline]

Definition at line 170 of file dllist.h.

Referenced by gwNumerics::addVDWSegments(), ConstraintManager::getAssocAtoms(), and Molecule::readFasta().

```
00171                   { return DLList<T>::ConstIterator(*this); }
00172
```

4.22.2.14 template<class T> DLList< T >::Iterator DLList<T>::GetIterator<T> () [inline]

Definition at line 167 of file dllist.h.

Referenced by OutputBinary(), and path_iterator::path_iterator().

```
00168                   { return DLList<T>::Iterator(*this); }
00169
```

4.22.2.15 template<class T> int DLList< T >::In (const T & *data*) const [inline]

Definition at line 364 of file dllist.inc.

```
00365   { return (FindLocation(data) != -1); }
00366
```

4.22.2.16 template<class T> int DLList< T >::IsEmpty () const [inline]

Definition at line 29 of file dllist.inc.

Referenced by AddToBack(), AddToFront(), IdGen::Get(), IdGen::Peek(), main(), operator int(), bf_iterator::operator++(), and df_iterator::operator++().

```
00030   { return (the_rep->Head ? 0 : 1); }
00031
```

4.22.2.17 template<class T> int DLList< T >::Number () const [inline]

Definition at line 37 of file dllist.inc.

Referenced by Molecule::DynRead(), Segments::GetNum(), Segments::Number(), IdGen::NumberFree(), OutputBinary(), Segments::QSortReduce(), Segments::QSortReduceViol(), Process::assertOrder(), jVector::dynread(), IntVec::dynread(), Process::numQueued(), Molecule::readFasta(), and Process::runChild().

```
00038   { return the_rep->M; }
00039
```

4.22.2.18 template<class T> T & DLList< T >::PeekBack () [inline]

Definition at line 285 of file dllist.inc.

```
{ return the_rep->Tail->Data; }

```

4.22.2.19 template<class T> const T & DLList< T >::PeekBack () const [inline]

Definition at line 281 of file dllist.inc.

Referenced by Process::addChild().

```
{ return the_rep->Tail->Data; }

```

4.22.2.20 template<class T> T & DLList< T >::PeekFront () [inline]

Definition at line 277 of file dllist.inc.

```
{ return the_rep->Head->Data; }

```

4.22.2.21 template<class T> const T & DLList< T >::PeekFront () const [inline]

Definition at line 273 of file dllist.inc.

Referenced by Queue::Peek(), and Process::addChild().

```
{ return the_rep->Head->Data; }

```

4.22.2.22 template<class T> ostream & DLList< T >::Print (ostream & *os*) const [inline]

Definition at line 372 of file dllist.inc.

```
{
  DLListNode<T> * Hop = the_rep->Head;
  while(Hop)
  {
    os << Hop->Data << " -> ";
    Hop = Hop->Next;
  }

```

4.22.2.23 template<class T> T DLList< T >::Remove (const T & *data*) [inline]

Definition at line 368 of file dllist.inc.

```
{ return RemoveLocation( FindLocation(data) ); }

```

4.22.2.24 template<class T> T DLList< T >::Remove () [inline]

Reimplemented in **Queue** (p. 401).

Definition at line 227 of file dllist.inc.

Referenced by Molecule::readFasta().

```
00228   { return RemoveFromFront(); }
00229
```

4.22.2.25 template<class T> T DLList< T >::RemoveFromBack () [inline]

Definition at line 202 of file dllist.inc.

Referenced by RemoveLocation(), Process::assertOrder(), Process::cleanQueue(), main(), Process::runChildren(), Process::runChildrenRepeatFirst(), and Process::runNextChild().

```
00203 {
00204   T DeletedData;
00205
00206   //  if(!the_rep->Tail) return DeletedData;  // gives warning
00207   if(the_rep->Tail){
00208
00209     DLListNode<T> * tmp = the_rep->Tail;
00210     the_rep->Tail = the_rep->Tail->Prev;
00211     if(the_rep->Tail) the_rep->Tail->Next = 0;
00212     else        the_rep->Head = 0;
00213
00214     DeletedData = tmp->Data;
00215     tmp->Prev = 0;
00216     delete tmp;
00217     --the_rep->M;
00218     return DeletedData;
00219   } else {
00220     cerr << "called RemoveFromBack from empty dllist. __FILE__, __LINE__" << endl;
00221     raise (SIGQUIT);
00222     return the_rep->Head->Data; // GD useless line -- warning removal only
00223   }
00224
```

4.22.2.26 template<class T> T DLList< T >::RemoveFromFront () [inline]

Definition at line 176 of file dllist.inc.

Referenced by Molecule::DynRead(), Stack::Pop(), Segments::Pop(), Segments::QSortReduce(), Queue::Remove(), Remove(), RemoveLocation(), jVector::dynread(), IntVec::dynread(), and main().

```
00177 {
00178   T DeletedData;
00179
00180   //  if(!the_rep->Head) return DeletedData; // gives warning for non-init
00181
00182   if(the_rep->Head){
00183
00184     DLListNode<T> * tmp = the_rep->Head;
00185     the_rep->Head = the_rep->Head->Next;
00186     if(the_rep->Head) the_rep->Head->Prev = 0;
00187     else           the_rep->Tail = 0;
```

```
    DeletedData = tmp->Data;
    tmp->Next = 0;
    delete tmp;
    --the_rep->M;
    return DeletedData;
  } else {
    cerr << "called RemoveFromFront from empty dllist. __FILE__, __LINE__" << endl;
    raise (SIGQUIT);
    return the_rep->Head->Data; // GD useless line -- warning removal only
  }
```

4.22.2.27 template<class T> T DLList< T >::RemoveLocation (int *i*) [inline]

Definition at line 290 of file dllist.inc.

Referenced by Remove().

```
{
  if(i<=0) { return RemoveFromFront(); }
  if(i>=the_rep->M-1) { return RemoveFromBack(); }

  DLListNode<T> * Hop;
  if(i<the_rep->M/2)
    {
      Hop = the_rep->Head;
      for(int j=0; j<i; ++j)
          Hop = Hop->Next;
    }
  else
    {
      Hop = the_rep->Tail;
      for(int j=the_rep->M-1; j>i; --j)
          Hop = Hop->Prev;
    }

```

4.22.2.28 template<class T> DLList< T > DLList< T >::UniqueCopy () const [inline]

Definition at line 446 of file dllist.inc.

Referenced by AddToBack(), and AddToFront().

```
  {
    DLList<T> dll;
    DLListNode<T> * Hop = the_rep->Head;
    while(Hop)
      {
        dll.AddToBack(Hop->Data);
        Hop = Hop->Next;
      }

```

4.22.2.29 template<class T> DLList<T>::operator int () const [inline]

Definition at line 94 of file dllist.h.

```
                { return !IsEmpty(); }

```

4.22.2.30 template<class T> int DLList< T >::operator!= (const DLList< T > & *dll*) const [inline]

Definition at line 340 of file dllist.inc.

```
{ return !( this->operator==(dll) ); }

```

4.22.2.31 template<class T> const DLList< T > & DLList< T >::operator+= (const DLList< T > & *dll*) [inline]

Definition at line 166 of file dllist.inc.

```
{ AddToBack(dll); return *this; }

```

4.22.2.32 template<class T> const DLList< T > & DLList< T >::operator= (const DLList< T > & *dll*) [inline]

Definition at line 435 of file dllist.inc.

```
{
  if(the_rep == dll.the_rep) return *this;

  if(the_rep) the_rep->Dereference();
  the_rep = dll.the_rep;
  if(the_rep) the_rep->Reference();
  return *this;

```

4.22.2.33 template<class T> int DLList< T >::operator== (const DLList< T > & *dll*) const [inline]

Definition at line 323 of file dllist.inc.

Referenced by operator!=().

```
{
  if(the_rep->M != dll.the_rep->M) return 0;
  if(the_rep->M == 0) return 1;

  DLListNode<T> * Hop1 = the_rep->Head;
  DLListNode<T> * Hop2 = dll.the_rep->Head;
  while(Hop1)
    {
      if(Hop1->Data != Hop2->Data) return 0;
      Hop1 = Hop1->Next;
      Hop2 = Hop2->Next;
    }

```

4.22.3 Friends And Related Function Documentation

4.22.3.1 template<class T> class ConstIterator [friend]

Definition at line 82 of file dllist.h.

Referenced by GetConstIterator().

4.22.3.2 template<class T> istream& InputBinary (istream & *is*, DLList< T > & *dll*) [friend]

Definition at line 389 of file dllist.inc.

```
{
  // Delete everything:
  //
  dll.Clear();

  // Read in the number of elements:
  //
  int numel;
  if(InputBinary(is,numel).bad())
      return is;

  T stuff;
  for(int i=0; i<numel; ++i)
    {
      if(InputBinary(is,stuff).bad())
          return is;
      dll.AddToBack(stuff);
    }
```

4.22.3.3 template<class T> class Iterator [friend]

Definition at line 81 of file dllist.h.

Referenced by GetIterator().

4.22.3.4 template<class T> ostream& OutputBinary (ostream & *os*, const DLList< T > & *dll*) [friend]

Definition at line 412 of file dllist.inc.

```
{
  // Write the number of elements:
  //
  if(OutputBinary(os, dll.Number()).bad())
      return os;

  for(DLList<T>::Iterator it = dll.GetIterator(); it; ++it)
      if(OutputBinary(os, *it).bad())
          return os;
  return os;
```

4.22.3.5 template<class T> ostream& operator (ostream & *os*, const DLList< T > & *dll*) [friend]

4.22.3.6 template<class T> ostream& operator (ostream & *os*, const DLList< T > & *dll*) [friend]

4.22.3.7 template<class T> istream& operator>= (istream & *is*, DLList< T > & *dll*) [friend]

Definition at line 427 of file dllist.inc.

```
{ return InputBinary(os,dll); }

```

4.22.4 Member Data Documentation

4.22.4.1 template<class T> DLListRep< T > * DLList<T>::the_rep [private]

Definition at line 174 of file dllist.h.

The documentation for this class was generated from the following files:

- **dllist.h**
- **dllist.inc**

4.23 DLListNode Class Template Reference

`#include <dllist.h>`

Collaboration diagram for DLListNode:

Public Methods

- **DLListNode** ()
- **DLListNode** (const T &data)
- **~DLListNode** ()

Public Attributes

- T **Data**
- DLListNode* **Next**
- DLListNode* **Prev**

Friends

- class **DLList< T >**
- class **DLListRep< T >**

template<class T> class DLListNode

4.23.1 Constructor & Destructor Documentation

4.23.1.1 template<class T> DLListNode<T>::DLListNode<T> () `[inline]`

Definition at line 36 of file dllist.h.

```
00037                 { Next = 0; Prev = 0; }
00038
```

4.23.1.2 template<class T> DLListNode<T>::DLListNode<T> (const T & *data*) `[inline]`

Definition at line 38 of file dllist.h.

```
00039                 { Next = 0; Prev = 0; Data = data; }
00040
```

4.23.1.3 template<class T> DLListNode<T>::~DLListNode<T> () `[inline]`

Definition at line 42 of file dllist.h.

```
00043                 { if(Next) delete Next; }
00044
```

4.23.2 Friends And Related Function Documentation

4.23.2.1 template<class T> class DLList [friend]

Definition at line 31 of file dllist.h.

4.23.2.2 template<class T> class DLListRep [friend]

Definition at line 32 of file dllist.h.

4.23.3 Member Data Documentation

4.23.3.1 template<class T> T DLListNode<T>::Data

Definition at line 46 of file dllist.h.

4.23.3.2 template<class T> DLListNode<T> * DLListNode<T>::Next

Definition at line 48 of file dllist.h.

4.23.3.3 template<class T> DLListNode<T> * DLListNode<T>::Prev

Definition at line 50 of file dllist.h.

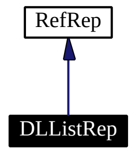
The documentation for this class was generated from the following file:

- **dllist.h**

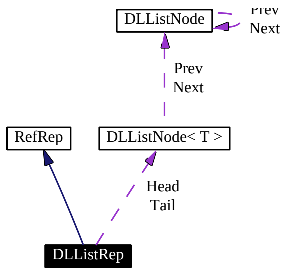
4.24 DLListRep Class Template Reference

`#include <dllist.h>`

Inheritance diagram for DLListRep



Collaboration diagram for DLListRep:



Public Methods

- **DLListRep** ()
- **~DLListRep** ()

Public Attributes

- int **M**
- **DLListNode**<T>∗ **Head**
- **DLListNode**<T>∗ **Tail**

Friends

- class **DLList**< **T** >

template<class T> class DLListRep

4.24.1 Constructor & Destructor Documentation

4.24.1.1 template<class T> DLListRep< T >::DLListRep<T> () `[inline]`

Definition at line 15 of file dllist.inc.

```
00016   { M=0; Head=0; Tail=0; }
00017
```

4.24.1.2 template<class T> DLListRep< T >::~DLListRep<T> () [inline]

Definition at line 19 of file dllist.inc.

```
{ if(Head) delete Head; }

```

4.24.2 Friends And Related Function Documentation

4.24.2.1 template<class T> class DLList [friend]

Definition at line 60 of file dllist.h.

4.24.3 Member Data Documentation

4.24.3.1 template<class T> DLListNode< T > * DLListRep<T>::Head

Definition at line 68 of file dllist.h.

4.24.3.2 template<class T> int DLListRep<T>::M

Definition at line 65 of file dllist.h.

4.24.3.3 template<class T> DLListNode< T > * DLListRep<T>::Tail

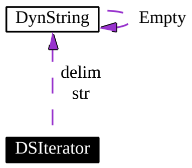
Definition at line 71 of file dllist.h.

The documentation for this class was generated from the following files:

- **dllist.h**
- **dllist.inc**

4.25 DSIterator Class Reference

Collaboration diagram for DSIterator:



Public Methods

- void **Reset** ()
- void **SetString** (const **DynString** &newstring)
- void **SetDelim** (const **DynString** &newdelims)
- **DynString operator** ∗ ()
- **operator int** ()
- const DSIterator& **operator++** ()
- void **Copy** (const DSIterator &it)
- const DSIterator& **operator=** (const DSIterator &it)
- **DSIterator** ()
- **DSIterator** (const DSIterator &it)
- **DSIterator** (const **DynString** &ds, const **DynString** &delims)
- **DSIterator** (const **DynString** &ds)
- **~DSIterator** ()

Private Methods

- int **DelimMatch** (char c)
- void **Position** ()

Private Attributes

- **DynString str**
- **DynString delim**
- int **b**
- int **e**
- int **reset**

4.25.1 Constructor & Destructor Documentation

4.25.1.1 DSIterator::DSIterator () `[inline]`

Definition at line 57 of file dsiter.inc.

```
00057                       { b=0; e=0; reset=0; }
00058
```

4.25.1.2 DSIterator::DSIterator (const DSIterator & *it*) `[inline]`

Definition at line 59 of file dsiter.inc.

```
00060                 { Copy(it); }
00061
```

4.25.1.3 DSIterator::DSIterator (const DynString & *ds*, const DynString & *delims*) `[inline]`

Definition at line 62 of file dsiter.inc.

```
00063                 {
00064                   str = ds;
00065                   delim = delims;
00066                   Reset();
00067                   reset = 0;
00068
```

4.25.1.4 DSIterator::DSIterator (const DynString & *ds*) `[inline]`

Definition at line 70 of file dsiter.inc.

```
00071                 {
00072                   str = ds;
00073                   delim = GetDSDelim();
00074                   Reset();
00075                   reset = 0;
00076
```

4.25.1.5 DSIterator::~DSIterator () `[inline]`

Definition at line 78 of file dsiter.inc.

```
00078                             {}
00079
```

4.25.2 Member Function Documentation

4.25.2.1 void DSIterator::Copy (const DSIterator & *it*) `[inline]`

Definition at line 45 of file dsiter.inc.

Referenced by DSIterator(), and operator=().

```
00046                 {
00047                   str = it.str;
00048                   delim = it.delim;
00049                   b = it.b;
00050                   e = it.e;
00051                   reset = it.reset;
00052
```

4.25.2.2 int DSIterator::DelimMatch (char *c*) `[inline, private]`

Definition at line 89 of file dsiter.inc.

Referenced by Position().

```
{
  for(int i=0; i<delim.Length(); ++i)
      if(delim[i] == c)
          return 1;
  return 0;
```

4.25.2.3 void DSIterator::Position () `[inline, private]`

Definition at line 97 of file dsiter.inc.

Referenced by Reset(), and operator++().

```
{
  // Move the beginning to the next non-delimiter:
  for(b=e+1; b<str.Length() && DelimMatch(str[b]); ++b);

  // Move the end to the position of the last non-delimiter
  // after the beginning:
  for(e=b; e<str.Length()-1 && !DelimMatch(str[e+1]); ++e);
```

4.25.2.4 void DSIterator::Reset () `[inline]`

Definition at line 13 of file dsiter.inc.

Referenced by DSIterator(), operator ∗(), operator int(), and operator++().

```
{ e = -1; Position(); }
```

4.25.2.5 void DSIterator::SetDelim (const DynString & *newdelims*) `[inline]`

Definition at line 20 of file dsiter.inc.

```
{ delim = newdelims; }
```

4.25.2.6 void DSIterator::SetString (const DynString & *newstring*) `[inline]`

Definition at line 16 of file dsiter.inc.

```
{ str = newstring; reset=1; }
```

4.25.2.7 DynString DSIterator::operator * () [inline]

Definition at line 25 of file dsiter.inc.

```
{
  if(reset) { Reset(); reset=0; }
```

4.25.2.8 DSIterator::operator int () [inline]

Definition at line 31 of file dsiter.inc.

```
{
  if(reset) { Reset(); reset=0; }
```

4.25.2.9 const DSIterator & DSIterator::operator++ () [inline]

Definition at line 37 of file dsiter.inc.

```
{
  if(reset) { Reset(); reset=0; }
```

4.25.2.10 const DSIterator & DSIterator::operator= (const DSIterator & *it*) [inline]

Definition at line 54 of file dsiter.inc.

```
{ Copy(it); return *this; }
```

4.25.3 Member Data Documentation

4.25.3.1 int DSIterator::b [private]

Definition at line 85 of file dsiter.inc.

4.25.3.2 DynString DSIterator::delim [private]

Definition at line 83 of file dsiter.inc.

4.25.3.3 int DSIterator::e [private]

Definition at line 85 of file dsiter.inc.

4.25.3.4 int DSIterator::reset [private]

Definition at line 87 of file dsiter.inc.

4.25.3.5 DynString DSIterator::str `[private]`

Definition at line 81 of file dsiter.inc.

Referenced by operator $*$().

The documentation for this class was generated from the following file:

- **dsiter.inc**

4.26 DynString Class Reference

#include <dynstring.h>

Collaboration diagram for DynString:

Public Methods

- **DynString** ()
- **DynString** (const char ∗str)
- **DynString** (const DynString &ds)
- **DynString** (char c)
- **DynString** (int i)
- **DynString** (double d)
- ∼**DynString** ()
- int **IsBad** () const
- int **DestResizeToFit** (int nchars)
- int **NonDestResizeToFit** (int nchars)
- int **Resize** (int newlength)
- int **Set** (const DynString &ds)
- int **Set** (const char ∗str, int numchars)
- int **Set** (int i, char c)
- int **Chop** (int n=1)
- void **Clear** ()
- const DynString& **Cat** (char c)
- const DynString& **Cat** (const char ∗str)
- const DynString& **Cat** (const DynString &ds)
- DynString **Sub** (int i, int j) const
- DynString **operator()** (int i, int j) const
- int **IsSub** (const DynString &substr) const
- int **Replace** (const DynString &oldstr, const DynString &newstr)
- void **ToUpper** ()
- void **ToLower** ()
- double **ToDbl** () const
- int **ToInt** () const
- int **Compare** (const DynString &ds) const
- const char∗ **GetData** () const
- char∗ **GetDataCopy** () const
- int **Length** () const
- int **Bytes** () const
- int **Blocks** () const
- const DynString& **Pad** (char c, int n)
- const DynString& **operator+=** (const DynString &ds)
- const char& **operator**[ ] (int i) const
- char& **operator**[ ] (int i)
- const DynString& **operator=** (const DynString &ds)
- const DynString& **operator=** (const char ∗str)

- const DynString& **operator=** (char c)
- const DynString& **operator=** (double number)
- int **GetChar** (istream &is)
- int **Read** (istream &is, bool append=false)
- int **Split** (DynString *&substr) const
- int **Output** (ostream &os) const
- int **Input** (istream &is)
- int **InputBinary** (istream &is)
- int **OutputBinary** (ostream &os) const
- **MAKECOMPARISONS_WITH_COMPARE** (DynString)
- **MAKE_IO_STREAM_OPERATORS** (DynString)
- **MAKE_BIN_STREAM_OPERATORS** (DynString)
- istream& **ReadLine** (istream &is, bool append=false)
- void **EatLine** (istream &is) const

Static Public Attributes

- char* **Bad** = 0
- const int **BlockSize** = 32
- int **Delimiter** [256]
- DynString **Empty** = ""

Private Methods

- void **MakeBad** ()
- int **DestResize** (int nblocks)
- int **NonDestResize** (int nblocks)

Private Attributes

- char* **innards**
- int **N**
- int **L**

Friends

- void **SetDSDelim** (const DynString &delim)
- void **AddDSDelim** (char delim)
- void **RemDSDelim** (char delim)
- DynString **GetDSDelim** ()
- DynString **IntToDS** (int i)
- DynString **Reproduce** (char c, int n)
- DynString **operator+** (DynString x, const DynString &y)
- int **DSprintf** (DynString &ds, const char *cntl,...)

4.26.1 Constructor & Destructor Documentation

4.26.1.1 DynString::DynString ()

Definition at line 186 of file dynstring.C.

Referenced by EatLine(), and ReadLine().

```
00187   {
00188     innards = 0; N = 0; L = 0;
00189     if(!DestResize(1)) MakeBad();
00190
```

4.26.1.2 DynString::DynString (const char * *str*)

Definition at line 192 of file dynstring.C.

```
00193   {
00194     innards = 0; N = 0; L = 0;
00195     Set(str,strlen(str));
00196
```

4.26.1.3 DynString::DynString (const DynString & *ds*)

Definition at line 198 of file dynstring.C.

```
00199   {
00200     innards = 0; N = 0; L = 0;
00201     Set(ds);
00202
```

4.26.1.4 DynString::DynString (char *c*)

Definition at line 204 of file dynstring.C.

```
00205   {
00206     innards = 0; N = 0; L = 0;
00207     char * str = new char[2];
00208     str[0] = c; str[1] = 0;
00209     Set(str,1);
00210     delete str;
00211
```

4.26.1.5 DynString::DynString (int *i*)

Definition at line 213 of file dynstring.C.

```
00214   {
00215     innards = 0; N = 0; L = 0;
00216     char * str = new char[100];
00217     sprintf(str, "%i", i);
00218     Set(str,strlen(str));
00219     delete str;
00220
```

4.26.1.6 DynString::DynString (double *d*)

Definition at line 222 of file dynstring.C.

```
{
  innards = 0; N = 0; L = 0;
  char * str = new char[100];
  sprintf(str, "%g", d);
  Set(str,strlen(str));
  delete str;

```

4.26.1.7 DynString::∼DynString ()

Definition at line 231 of file dynstring.C.

```
{
  if(!IsBad()) delete[] innards;

```

4.26.2 Member Function Documentation

4.26.2.1 int DynString::Blocks () const `[inline]`

Definition at line 141 of file dynstring.h.

```
{ return N; }

```

4.26.2.2 int DynString::Bytes () const `[inline]`

Definition at line 138 of file dynstring.h.

```
{ return N*BlockSize; }

```

4.26.2.3 const DynString & DynString::Cat (const DynString & *ds*)

Definition at line 106 of file dynstring.C.

```
{
  if(IsBad() || ds.IsBad()) return *this;
  int newlen = L+ds.L;
  if(!NonDestResizeToFit(newlen)) return *this;
  strncat(innards, ds.innards, ds.L);
  L += ds.L;
  innards[L] = '\0';
  return *this;

```

4.26.2.4 const DynString& DynString::Cat (const char * *str*)

4.26.2.5 const DynString & DynString::Cat (char *c*)

Definition at line 97 of file dynstring.C.

Referenced by GetChar(), Pad(), Read(), Split(), and operator+=().

```
00098  {
00099    if(IsBad()) return *this;
00100    if(!NonDestResizeToFit(L+1)) return *this;
00101    innards[L++] = c;
00102    innards[L]   = '\0';
00103    return *this;
00104
```

4.26.2.6 int DynString::Chop (int *n* = 1)

Definition at line 175 of file dynstring.C.

```
00176  {
00177    if(IsBad() || n<0) return 0;
00178    if(n==0) return L;
00179    n = (n<=L) ? n : L;
00180    NonDestResizeToFit(L-n);
00181    L = L-n;
00182    innards[L] = '\0';
00183    return L;
00184
```

4.26.2.7 void DynString::Clear () `[inline]`

Definition at line 83 of file dynstring.h.

Referenced by Read(), Residue::Residue(), and Molecule::readFasta().

```
00084  {
00085    if(!DestResize(1)) MakeBad();
00086    else { L=0; innards[0] = '\0'; }
00087
```

4.26.2.8 int DynString::Compare (const DynString & *ds*) const `[inline]`

Definition at line 122 of file dynstring.h.

```
00123  { return strcmp(innards,ds.innards); }
00124
```

4.26.2.9 int DynString::DestResize (int *nblocks*) `[private]`

Definition at line 69 of file dynstring.C.

Referenced by Clear(), DestResizeToFit(), DynString(), and Set().

```
{
  if(N == nblocks) return 1;
  if(nblocks<1) return 0;
  char * tmp = innards;
  if(!(innards = new char[nblocks*BlockSize]))
    { innards = tmp; return 0; }

```

4.26.2.10 int DynString::DestResizeToFit (int *nchars*) [inline]

Definition at line 45 of file dynstring.h.

Referenced by RanWord::Get(), GetDSDelim(), InputBinary(), Replace(), Reproduce(), Set(), Sub(), and operator=().

```
{ return DestResize((nchars/BlockSize)+1); }

```

4.26.2.11 void DynString::EatLine (istream & *is*) const

Definition at line 770 of file dynstring.C.

Referenced by main().

```
{ DynString().ReadLine(is); }

```

4.26.2.12 int DynString::GetChar (istream & *is*)

Definition at line 451 of file dynstring.C.

Referenced by Molecule::readFasta().

```
{
  char c;
  is.get(c);
  if(is.eof() || is.bad()) return 0;
  Cat(c);
  return 1;

```

4.26.2.13 const char * DynString::GetData () const [inline]

Definition at line 128 of file dynstring.h.

Referenced by main().

```
{ return innards; }

```

4.26.2.14 char * DynString::GetDataCopy () const

Definition at line 412 of file dynstring.C.

```
{
  if(IsBad()) return Bad;
  char * s = new char[L+1];
  strncpy(s,innards,L);
  s[L]='\0';
  return s;
```

4.26.2.15 int DynString::Input (istream & *is*)

Definition at line 742 of file dynstring.C.

```
{ return Read(is); }
```

4.26.2.16 int DynString::InputBinary (istream & *is*)

Definition at line 421 of file dynstring.C.

```
{
  // Delete everything
  //
  delete[] innards; innards = 0;
  N = 0;
  L = 0;

  int num_chars=0;
  is >= num_chars;

  if(!DestResizeToFit(num_chars))
      { cerr << "DynString::InputBinary\n"; raise(SIGQUIT); }
```

4.26.2.17 int DynString::IsBad () const [inline]

Definition at line 41 of file dynstring.h.

Referenced by Cat(), Chop(), GetDataCopy(), IsSub(), Output(), Set(), Sub(), and ~DynString().

```
{ return (innards == Bad); }
```

4.26.2.18 int DynString::IsSub (const DynString & *substr*) const

Definition at line 266 of file dynstring.C.

```
{
  if(IsBad() || substr.IsBad()) return 0;

  int match;
  for(int i=0; i<L-substr.L+1; ++i)
    {
      match = 1;
      for(int j=0; j<substr.L; ++j)
          if(innards[i+j] != substr.innards[j]) { match = 0; break; }
      if(match) return 1;
    }
```

4.26.2.19 int DynString::Length () const `[inline]`

Definition at line 135 of file dynstring.h.

Referenced by DSIterator::DelimMatch(), DSIterator::Position(), Replace(), AtomRegistry::Set-Type(), AnimFrame::Set_Output_Stream(), and DSIterator::operator int().

```
{ return L; }
```

4.26.2.20 DynString::MAKECOMPARISONS_WITH_COMPARE (DynString)

4.26.2.21 DynString::MAKE_BIN_STREAM_OPERATORS (DynString)

4.26.2.22 DynString::MAKE_IO_STREAM_OPERATORS (DynString)

4.26.2.23 void DynString::MakeBad () `[inline, private]`

Definition at line 223 of file dynstring.h.

Referenced by Clear(), DynString(), Set(), Sub(), and operator=().

```
{ N=0; L=0; delete[] innards; innards = Bad; }
```

4.26.2.24 int DynString::NonDestResize (int *nblocks*) `[private]`

Definition at line 81 of file dynstring.C.

Referenced by NonDestResizeToFit().

```
{
  if(N == nblocks) return 1;
  if(nblocks<1) return 0;
  char * tmp = innards;
  if(!(innards = new char[nblocks*BlockSize]))
    { innards = tmp; return 0; }
```

4.26.2.25 int DynString::NonDestResizeToFit (int *nchars*) `[inline]`

Definition at line 47 of file dynstring.h.

Referenced by Cat(), Chop(), Resize(), Set(), and operator[]().

```
{ return NonDestResize((nchars/BlockSize)+1); }
```

4.26.2.26 int DynString::Output (ostream & *os*) const

Definition at line 733 of file dynstring.C.

```
00734 {
00735   if(IsBad())
00736       os << "<<BAD STRING!>>";
00737   else
00738       os << innards;
00739   return L;
00740
```

4.26.2.27 int DynString::OutputBinary (ostream & *os*) const

Definition at line 443 of file dynstring.C.

```
00444 {
00445   os <= L;
00446   if(L>0)
00447       os.write(innards, L*sizeof(char));
00448   return L;
00449
```

4.26.2.28 const DynString & DynString::Pad (char *c*, int *n*)

Definition at line 755 of file dynstring.C.

```
00756 {
00757   DynString padding = Reproduce(c,n);
00758   return Cat(padding);
00759
```

4.26.2.29 int DynString::Read (istream & *is*, bool *append* = false)

Definition at line 460 of file dynstring.C.

Referenced by Input(), and ReadLine().

```
00461 {
00462   char c;
00463   int len=0;
00464
00465   if(!append) Clear();
00466
00467   is.unsetf(ios::skipws);
00468   // Skip all the header delimiter characters.
00469   for(int done=0; !done; )
00470     {
00471       c = is.peek();
00472
00473       // See if the character matches any delimiter character.
00474       if(c == EOF || !Delimiter[(uint) c]) done = 1;
00475
00476       // Otherwise advance to the next character in the stream.
00477       else is.get();
00478     }
00479
00480   // Read in a series of consecutive non-delimiter characters.
00481   for(int done=0; !done; )
00482     {
00483       c = is.peek();
00484       if((c == EOF) || Delimiter[(uint) c]) done = 1;
```

```
      else
        {
          Cat(c);
          is.get();
          ++len;
        }
    }

```

4.26.2.30 istream & DynString::ReadLine (istream & *is*, bool *append* = false)

Definition at line 761 of file dynstring.C.

Referenced by EatLine(), Molecule::readFasta(), Molecule::readXmol(), and Molecule::skipXmol().

```
{
  DynString old_delim = GetDSDelim();
  SetDSDelim(DynString("\n"));
  Read(is,append);
  SetDSDelim(old_delim);
  return is;

```

4.26.2.31 int DynString::Replace (const DynString & *oldstr*, const DynString & *newstr*)

Definition at line 281 of file dynstring.C.

```
{
  int oldlen = oldstr.Length(), newlen = newstr.Length();
  int * match = 0;
  int replace=0;
  int total=-1;

  if(L<1 || oldlen<1) return 0;

  if(!(match = new int[L]))
      { cerr << "DynString::Replace#1\n"; raise(SIGQUIT); }

  if(oldlen < 1) return 0;

  // Find the number of (non-overlapping replacements we need to make):
  for(int i=0; i<L; ++i) match[i] = 0;
  for(int i=0; i<L-oldlen+1; )
    {
      if(!strncmp(innards+i, oldstr.innards, oldlen))
        {
          // Found a match!
          ++replace;
          match[i] = 1;
          i += oldlen;
        }
      else
        { match[i] = 0; ++i; }
    }

  total = L + replace*(newlen-oldlen);

  if(total<1)
    {
```

```
      delete[] match;
      return 0;
    }

  // Copy the old string.
  char * oldinnards = 0;
  if(!(oldinnards = new char[L+1]))
      { cerr << "DynString::Replace#2\n"; raise(SIGQUIT); }
  int oldL = L;
  for(int i=0; i<oldL; ++i) oldinnards[i] = innards[i];
  oldinnards[oldL] = '\0';

  // Make space for the new string.
  if(!DestResizeToFit(total))
      { cerr << "DynString::Replace#3\n"; raise(SIGQUIT); }

  int j=0;
  for(int i=0; i<oldL; )
    {
      if(match[i])
        {
          for(int k=0; k<newlen; ++k)
              innards[j++] = newstr[k];
          i += oldlen;
        }
      else
          innards[j++] = oldinnards[i++];
    }

```

4.26.2.32 int DynString::Resize (int *newlength*) [inline]

Definition at line 49 of file dynstring.h.

```
{ return NonDestResizeToFit(newlength); }

```

4.26.2.33 int DynString::Set (int *i*, char *c*)

Definition at line 154 of file dynstring.C.

```
{
  if(IsBad() || i<0) return 0;
  if(i>=N*BlockSize-1)
    {
      if(!NonDestResizeToFit(i))
        {
          MakeBad();
          return 0;
        }
    }
  if(i>=L)
    {
      innards[L] = innards[N*BlockSize-1];
      L = i+1;
      innards[L] = '\0';
    }

```

4.26.2.34 int DynString::Set (const char * *str*, int *num*)

Definition at line 136 of file dynstring.C.

```
{
  if(!str || str == Bad)
   { MakeBad(); return 0; }

 if(num<=0)
   { DestResize(1); return 0; }

```

4.26.2.35 int DynString::Set (const DynString & *ds*)

Definition at line 117 of file dynstring.C.

Referenced by DynString(), and operator=().

```
{
  if(this == &ds)
      return 1;

  if(ds.L<1)
    { DestResize(1); return 0; }

```

4.26.2.36 int DynString::Split (DynString *& *substr*) const

Definition at line 496 of file dynstring.C.

```
{
  // Number of substrings returned;
  int n=0;

  if(substr)
      delete[] substr;
  substr = 0;

  // Count the number of substrings there are:
  int i=0;
  while(i<L)
    {
      // Skip leading stretch of delimiters:
      for(; i<L; ++i)
        {
          if(!Delimiter[(uint) innards[i]])
              break;
          cout << '<' << innards[i] << '>';
        }
      // Increment the substring count if there's still
      // characters left in the string:
      ++n;

      // Skip over this substring:
      for(; i<L; ++i)
        {
          if(Delimiter[(uint) innards[i]])
              break;
          cout << '[' << innards[i] << ']';
        }
```

```
      }

    if(n < 1)
        return 0;

    // Create space for the substrings:
    substr = new DynString[n];

    i=0;
    int j=-1;
    while(i<L)
      {
        // Skip all the header delimiter characters.
        for(i=0; i<L; ++i)
            // See if the character matches any delimiter.
            if(!Delimiter[(uint) innards[i]])
                break;

        // Read in a series of consecutive non-delimiter.
        substr[++j] = "";
        for(; i<L; ++i)
          {
            if(Delimiter[(uint) innards[i]])
                break;
            else
                substr[j].Cat(innards[i]);
          }
      }

```

4.26.2.37 DynString DynString::Sub (int *i*, int *j*) const

Definition at line 236 of file dynstring.C.

Referenced by operator()().

```
  {
    DynString substr = "";
    if(L <= 0) return substr;
    if(IsBad())
      {
        substr.MakeBad();
        return substr;
      }

    int k;
    if(j<i)
      {
        k = j;
        j = i;
        i = k;
      }

    if(i<0) i = 0;
    if(j >= L) j = L-1;

    if(!substr.DestResizeToFit(j-i+1))
        { cerr << "DynString::Sub\n"; raise(SIGQUIT); }

```

4.26.2.38 double DynString::ToDbl () const [inline]

Definition at line 114 of file dynstring.h.

Referenced by ddGraph::Input(), and main().

```
00115   { return atof(innards); }
00116
```

4.26.2.39 int DynString::ToInt () const [inline]

Definition at line 116 of file dynstring.h.

Referenced by ddGraph::Input(), main(), Molecule::readXmol(), and Molecule::skipXmol().

```
00117   { return ((int) atoi(innards)); }
00118
```

4.26.2.40 void DynString::ToLower () [inline]

Definition at line 110 of file dynstring.h.

```
00111   { for(int i=0; i<L; ++i) innards[i] = tolower(innards[i]); }
00112
```

4.26.2.41 void DynString::ToUpper () [inline]

Definition at line 108 of file dynstring.h.

Referenced by Molecule::readFasta().

```
00109   { for(int i=0; i<L; ++i) innards[i] = toupper(innards[i]); }
00110
```

4.26.2.42 DynString DynString::operator() (int *i*, int *j*) const [inline]

Definition at line 96 of file dynstring.h.

```
00097   { return Sub(i,j); }
00098
```

4.26.2.43 const DynString & DynString::operator+= (const DynString & *ds*) [inline]

Definition at line 157 of file dynstring.h.

```
00158   { return Cat(ds); }
00159
```

4.26.2.44 const DynString & DynString::operator= (double *number*)

Definition at line 383 of file dynstring.C.

```
{
  char * s = new char[30];
  sprintf(s, "%g", number);
  Set(s,strlen(s));
  delete[] s;
  return *this;

```

4.26.2.45 const DynString & DynString::operator= (char *c*)

Definition at line 372 of file dynstring.C.

```
{
  // Get rid of extra space if it is currently allocated.
  //
  if(!DestResizeToFit(1)) { MakeBad(); return *this; }

```

4.26.2.46 const DynString & DynString::operator= (const char * *str*)

Definition at line 369 of file dynstring.C.

```
{ if(innards != str) Set(str,strlen(str)); return *this; }

```

4.26.2.47 const DynString & DynString::operator= (const DynString & *ds*)

Definition at line 362 of file dynstring.C.

```
{
  if(!ds.innards) return ($\ast$this);
  Set(ds);
  return *this;

```

4.26.2.48 char & DynString::operator[ ] (int *i*)

Definition at line 399 of file dynstring.C.

```
{
  if(i<0)
    { cerr << "DynString::operator[]\n"; raise(SIGQUIT); }
  if(i>=L)
    {
      NonDestResizeToFit(i+1);
      L = i+1;
      innards[L] = '\0';
    }

```

4.26.2.49 const char & DynString::operator[] (int *i*) const

Definition at line 392 of file dynstring.C.

```
{
  if((i>=L)||(i<0))
    { cerr << "DynString::operator[]\n"; raise(SIGQUIT); }
```

4.26.3 Friends And Related Function Documentation

4.26.3.1 void AddDSDelim (char *delim*) `[friend]`

Definition at line 45 of file dynstring.C.

```
{ DynString::Delimiter[(uint) delim] = 1; }
```

4.26.3.2 int DSprintf (DynString & *ds*, const char * *cntl*, ...) `[friend]`

4.26.3.3 DynString GetDSDelim () `[friend]`

Definition at line 51 of file dynstring.C.

Referenced by ReadLine().

```
{
  DynString delim;
  delim.DestResizeToFit(256);
  int j=0;
  for(uint i=0; i<256; ++i)
    {
      if(DynString::Delimiter[i])
        {
          delim.innards[j] = (char) i;
          ++j;
        }
    }
```

4.26.3.4 DynString IntToDS (int *i*) `[friend]`

Definition at line 351 of file dynstring.C.

```
{
  char s[100]; s[0] = '\0';
  sprintf(s, "%i", i);
  DynString ds(s);
  return ds;
```

4.26.3.5 void RemDSDelim (char *delim*) `[friend]`

Definition at line 47 of file dynstring.C.

```
{ DynString::Delimiter[(uint) delim] = 0; }
```

4.26.3.6 DynString Reproduce (char *c*, int *n*) [friend]

Definition at line 745 of file dynstring.C.

Referenced by Pad().

```
{
  DynString str;
  if(!str.DestResizeToFit(n)) return str;
  for(int i=0; i<n; ++i) str.innards[i] = c;
  str.innards[n] = '\0';
  str.L = n;
  return str;
```

4.26.3.7 void SetDSDelim (const DynString & *delim*) [friend]

Definition at line 34 of file dynstring.C.

Referenced by ReadLine().

```
{
  uint j;
  for(int i=0; i<256; ++i) DynString::Delimiter[i] = 0;
  for(int i=0; i<delim.L; ++i)
    {
      j = (uint) delim.innards[i];
      DynString::Delimiter[j] = 1;
    }
```

4.26.3.8 DynString operator+ (DynString *x*, const DynString & *y*) [friend]

Definition at line 162 of file dynstring.h.

```
{ return x+=y; }
```

4.26.4 Member Data Documentation

4.26.4.1 char * DynString::Bad = 0 [static]

Definition at line 16 of file dynstring.C.

4.26.4.2 const int DynString::BlockSize = 32 [static]

Definition at line 11 of file dynstring.C.

4.26.4.3 int DynString::Delimiter[256] [static]

Initial value:

```
{
```

```
  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
  1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
}
```

Definition at line 20 of file dynstring.C.

4.26.4.4 DynString DynString::Empty = "" `[static]`

Definition at line 18 of file dynstring.C.

4.26.4.5 int DynString::L `[private]`

Definition at line 243 of file dynstring.h.

4.26.4.6 int DynString::N `[private]`

Definition at line 240 of file dynstring.h.

4.26.4.7 char * DynString::innards `[private]`

Definition at line 238 of file dynstring.h.

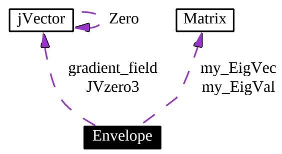
The documentation for this class was generated from the following files:

- **dynstring.h**
- **dynstring.C**

4.27 Envelope Class Reference

`#include <envelope.h>`

Collaboration diagram for Envelope:



Public Methods

- **Envelope** ()
  constructor.
- **Envelope** (const Envelope &e)
  copy constructor.
- ~**Envelope** ()
  destructor.
- const double **operator()** (int x, int y, int **z**) const
  access density by cell.
- const double **operator()** (const **jVector** &J) const
  access density by position.
- const **jVector**& **gradient** (int x, int y, int **z**) const
  access gradient by cell.
- const **jVector**& **gradient** (**jVector** J) const
  access gradient.
- int **Build** (const **Molecule** &Mol, double IW_in=DEFAULT_CUBE_WIDTH)
  create envelope from current molecule.
- void **Get_Eigenvectors** (**Matrix** &EigVec, **Matrix** &EigVal) const
  writes Eigenvalues and Eigenvectors matricies by reference.
- int **WriteDensity** (ostream &outStream, const **Molecule** &Mol) const
  write out a list of atoms and their density in the env.
- int **WriteAvsTxt** (ostream &outStream) const
  write AVS density format(as text).
- int **WriteVRML** (ostream &outStream) const
  write VRML 1.0 format.

- int **WriteVRMLgr** (ostream &outStream) const

  write VRML 1.0 format with gradient arrows.

- void **DiffReport** (ostream &outStream, const Envelope &E)

  Report on differences to another env.

- int **Compare** (const Envelope &ec) const

  default compare function.

- **MAKECOMPARISONS_WITH_COMPARE** (Envelope)
- int **Read** (istream &is)
- int **ReadBinary** (istream &is)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const
- void **Calc_Eigenvectors** ()

  internal call to calc and store Eigenvectors.

- void **CalcGradient** (int d=1)

  uses the current density_array to calculate the gradient_field.

- void **Upsize** (int k)

  increases the box size by a factor k'.

- void **Blur** (int k)

  *performs flat kernel convolution of 3D size (2*kernelSize+1) per side.*

- void **NormalizeMax** ()

  max max density = def's MAX_ENV_DEN' value and scale all cells accordingly.

- double **QueryDensity** (const **jVector** &pos) const

  returns density at 'pos.

- **jVector QueryGradient** (const **jVector** &pos) const
- double& **operator()** (int x, int y, int **z**)

  lvalue (for assigning density).

- **jVector**& **gradient** (int x, int y, int **z**)

  lvalue (for assigning gradient).

- int **ReadAvsTxt** (istream &in)

  read text-converted AVS density format.

- void **Env_Setup1** ()

  allocates memory.

- void **Env_Setup2** ()

  calculates meta information about data.

- void **Center** (void)

  adjust the max[xyz] and min[xyz] of the envelope to position the center of mass at the origin.

Public Attributes

- double∗ **density_array**

  Array of doubles storing density.

- **jVector**∗ **gradient_field**

  Array of jVectors storing gradient.

- int **numPoints**

  Number of cubes in Envelope.

- int **xdim**

  Dimensions in number of cubes.

- int **ydim**

  Dimensions in number of cubes.

- int **zdim**

  Dimensions in number of cubes.

- double **InputWidth**

  Real width of cubes (e.g. 3.7A).

- double **minx**

  Real position of min edges of region.

- double **miny**

  Real position of min edges of region.

- double **minz**

  Real position of min edges of region.

- double **maxx**

  Real position of max edges of region.

- double **maxy**

  Real position of max edges of region.

- double **maxz**

  Real position of max edges of region.

- double **cm_x**

  Center of mass, weighted by probabilities, (as blocks).

- double **cm_y**

  Center of mass, weighted by probabilities, (as blocks).

- double **cm_z**

  Center of mass, weighted by probabilities, (as blocks).

- double **r_cm_x**

  Center of mass, weighted by probabilities, (Real postition!).

- double **r_cm_y**

  Center of mass, weighted by probabilities, (Real postition!).

- double **r_cm_z**

  Center of mass, weighted by probabilities, (Real postition!).

- double **totalDensity**

  Total of all the density numbers in the cubes.

- int **filledPoints**

  Number of non-zero cubes in Envelope.

- **Matrix my_EigVal**

  Eigenvalues for the envelope.

- **Matrix my_EigVec**

  Eigenvectors for the envelope.

- const **jVector JVzero3**

  useful null vector.

Friends

- class **EnvelopeConstraint**

4.27.1 Constructor & Destructor Documentation

4.27.1.1 Envelope::Envelope ()

constructor.

Definition at line 3 of file envelope.C.

```
00005 {
00006   // this is an unititialized envelope.
00007
00008   xdim = ydim = zdim = -1;
00009   minx = miny = minz = maxx = maxy = maxz = 0;
00010
00011   density_array = 0;
00012   gradient_field = 0;
00013
00014   numPoints = 0;
00015   InputWidth = -1;
00016
00017
```

4.27.1.2 Envelope::Envelope (const Envelope & *env_in*)

copy constructor.

Definition at line 20 of file envelope.C.

```
00021 {
00022
00023   xdim = env_in.xdim;
00024   ydim = env_in.ydim;
00025   zdim = env_in.zdim;
00026
00027   minx = env_in.minx;
00028   miny = env_in.miny;
00029   minz = env_in.minz;
00030
00031   maxx = env_in.maxx;
00032   maxy = env_in.maxy;
00033   maxz = env_in.maxz;
00034
00035   InputWidth = env_in.InputWidth;
00036   numPoints = env_in.numPoints;
00037
00038   // allocate
00039   this->Env_Setup1();
00040
00041   // copy data
00042   for (int c = 0; c<numPoints; c++){
00043     density_array[c] = env_in.density_array[c];
00044     gradient_field[c] = env_in.gradient_field[c];
00045   }
00046
```

4.27.1.3 Envelope::∼Envelope ()

destructor.

Definition at line 52 of file envelope.C.

```
00052                   {
00053
00054   if (density_array)
00055     delete [] density_array;
00056
00057   if (gradient_field){
00058     delete [] gradient_field;
00059   }
00060
```

4.27.2 Member Function Documentation

4.27.2.1 void Envelope::Blur (int *k*)

performs flat kernel convolution of 3D size (2∗kernelSize+1) per side.

Definition at line 714 of file envelope.C.

Referenced by EnvelopeConstraint::PreSetup().

```
00714                    {
00715
```

```
//  cerr << "------------ Blur Envelope Start    --------" << endl;

if (k < 1){
  DIE("kernel size must be > 0");
}

int k2 = 2 * k;
int kernelsize = (k2+1)*(k2+1)*(k2+1); // number of cells in the kernel

int new_numPoints = (xdim+k2)*(ydim+k2)*(zdim+k2);
int new_xdim = xdim + k2;
int new_ydim = ydim + k2;
int new_zdim = zdim + k2;

double o_kernelsize = 1./(double)kernelsize;
double d = InputWidth * k;
double * new_density_array = new double[new_numPoints]; // not sparse -- saving lots of zeros here. mem wasted.  FIX
for (int l=0;l<new_numPoints;l++){new_density_array[l] =0;}
jVector * new_gradient_field = new jVector[new_numPoints];
          // need to fix gradients after resize on blur. -- consider RESIZE function.

double spread_val;  // density value to be "spread" to its neighbors

// cycle over old array, processing cells with data
int zh,yh,xh;
for (zh =0;zh<zdim ;zh++){
  for (yh =0;yh<ydim ;yh++){
    for (xh =0;xh<xdim ;xh++){
      spread_val = ($\ast$this)(xh,yh,zh);

      if (spread_val > 0) {

        // spread out value in cell across kernel (normalized to o_kernelsize)
        if (spread_val == 1) {
          spread_val = o_kernelsize;
        } else {
          spread_val = spread_val * o_kernelsize;
        }

        for (int kz = 0; kz<(k2+1);kz++){
          for (int ky = 0; ky<(k2+1);ky++){
            for (int kx = 0; kx<(k2+1);kx++){
              new_density_array[\
                                 (xh+kx)  \
                                 +  ((yh*new_xdim)+(ky*new_xdim))  \
                                 +  ((zh*new_xdim*new_ydim)+(kz*new_ydim*new_xdim)) \
              ] += spread_val;

            }  //kernel x
          }    //kernel y
        }      //kernel z
      } // skip 0 cells in old matrix.
    }   //x
  }     //y
}       //z

//  resize to fit k more cells per side (d is the real size)

minx -= d;
miny -= d;
minz -= d;

maxx += d;
maxy += d;
maxz += d;
```

```


  xdim = new_xdim;
  ydim = new_ydim;
  zdim = new_zdim;

  if (density_array && gradient_field) {
    delete [] density_array;
    density_array = new_density_array;

    delete [] gradient_field;
    gradient_field = new_gradient_field;  // empty

    numPoints = new_numPoints;

  } else {
    DIE("Called Blur on Envelope without intializing internal datastructures.");
  }

```

4.27.2.2 int Envelope::Build (const Molecule & *Mol*, double *IW_in* = DEFAULT_CUBE_WIDTH)

create envelope from current molecule.

Definition at line 248 of file envelope.C.

Referenced by jdOcc_Count::run().

```
{

   //  here we need to build an envelope to describe the CM current molecule

  InputWidth = IW_in;
  Mol.Bounds(minx,miny,minz,maxx,maxy,maxz);
  // figure out dimensions

  xdim = 1 + int ((maxx-minx)/InputWidth);
  ydim = 1 + int ((maxy-miny)/InputWidth);
  zdim = 1 + int ((maxz-minz)/InputWidth);

  maxx = minx + InputWidth*xdim; // reset to match
  maxy = miny + InputWidth*ydim;
  maxz = minz + InputWidth*zdim;

  numPoints = xdim*ydim*zdim;

  this->Env_Setup1();

  int zh, yh, xh;
  for (zh =0;zh<zdim ;zh++){
    for (yh =0;yh<ydim ;yh++){
      for (xh =0;xh<xdim ;xh++){
        ($\ast$this)(xh,yh,zh) = 0;
      }
    }
  }

  int n = Mol.Number();
  jVector center;
  double radius;

  RanDbl RD;
```

```
//  double cenx, ceny, cenz;
double cenx, ceny, cenz;
register double x, y, z;
int x_cell,y_cell,z_cell;
int N = 100000;

// what we really want to count is how much "atom" there
// is overlapping each box.

// we exect 1/frac points per atom to be counted
double frac = 6.02/(N * PI);

// except in cases where the box size is smaller then the atom size
// in this case we want these boxes to have density of 1 so we have
// to adjust the 'frac' accordingly

// assuming carbon VDW radius of 1.4 A here:
if (InputWidth < 2.8) {
  cerr << "Envelope::Build->adjusting fraction for small gridsize" << endl;
  double temp1 = (1.4/InputWidth);
  frac = 8 * temp1* temp1* temp1 / N;

}
WARN_IF(frac > 0.01, "May need to increase counts for good statistics");
  //  cerr << "frac = " << frac << endl;

for (int i =0; i < n; i++){
  cerr << ".";
  center = Mol(i);
  radius = Mol.getAtom(i).VDW();

  cenx = center[0];
  ceny = center[1];
  cenz = center[2];

  for (int j = 0;j<N;j++){
    x = RD.Get(-1,1);       y = RD.Get(-1,1);       z = RD.Get(-1,1);
    // XYZ ON (0,1) range
    if (x*x+y*y+z*z < 1){
      // point is inside unit sphere
      x = cenx + (x * radius);
      y = ceny + (y * radius);
      z = cenz + (z * radius);

      x_cell = (int) floor((x-minx)/InputWidth);
      y_cell = (int) floor((y-miny)/InputWidth);
      z_cell = (int) floor((z-minz)/InputWidth);

      ($\ast$this)(x_cell,y_cell,z_cell) = ($\ast$this)(x_cell,y_cell,z_cell) + frac;

    }// inside sphere?
  } // gen points
}

cerr << endl;

for (zh =0;zh<zdim ;zh++){
  for (yh =0;yh<ydim ;yh++){
    for (xh =0;xh<xdim ;xh++){
      if (($\ast$this)(xh,yh,zh) > MAX_ENV_DEN)
        ($\ast$this)(xh,yh,zh) = MAX_ENV_DEN;
    }
  }
}

```

4.27.2.3 void Envelope::CalcGradient (int $d = 1$)

uses the current density_array to calculate the gradient_field.

Definition at line 440 of file envelope.C.

Referenced by Blur(), ReadAvsTxt(), and Upsize().

```
00440                                 {
00441 
00442   if (d < 1){
00443     DIE ("gradient distance must be > 0");
00444   }
00445 
00446   double den;
00447   double val;
00448   jVector direction;
00449   int x,y,z;
00450   jVector J;
00451   int zh,yh,xh;
00452 
00453   for (zh =0;zh<zdim ;zh++){
00454     for (yh =0;yh<ydim ;yh++){
00455       for (xh =0;xh<xdim ;xh++){
00456 
00457         den =($\ast$this)(xh,yh,zh);
00458 
00459         // setting gradient at xh,yh,zh
00460         (($\ast$this).gradient(xh,yh,zh)).zero();
00461 
00462         for (int kz = (-1*d); kz<=d;kz++){
00463           for (int ky = (-1*d); ky<=d;ky++){
00464             for (int kx = (-1*d); kx<=d;kx++){
00465 
00466 
00467 
00468               // add-from coords
00469               x = xh+kx;
00470               y = yh+ky;
00471               z = zh+kz;
00472 
00473               if ((x >= 0) && (x < xdim) &&
00474                   (y >= 0) && (y < ydim) &&
00475                   (z >= 0) && (z < zdim))  {
00476                 // bounds OK
00477 
00478                 J.zero();
00479                 if ((val = ($\ast$this)(x,y,z))){ //sic
00480                   J[0] = kx*val;
00481                   J[1] = ky*val;
00482                   J[2] = kz*val;
00483 
00484                   J.norm_unit(); // J div by magnitude squared
00485 
00486                   (($\ast$this).gradient(xh,yh,zh)) = (($\ast$this).gradient(xh,yh,zh)) + J;
00487                 } // else add nothing -- no density here
00488 
00489               } // bounds ok on add-from box
00490 
00491               }  //kernel x
00492             }    //kernel y
00493           }      //kernel z
00494 
00495       }   //x
00496     }     //y
```

```
00497   }
00498
```

4.27.2.4 void Envelope::Calc_Eigenvectors (void)

internal call to calc and store Eigenvectors.

Definition at line 510 of file envelope.C.

Referenced by Env_Setup2().

```
00510                                 {
00511
00512   Matrix Spread(3,3); // real symmetric
00513   Spread.zero();
00514
00515   //  jVector tempV;
00516   double val;
00517   // Calc spread
00518   double dx, dy, dz = 0;
00519
00520
00521   for (int zh =0;zh<zdim ;zh++){
00522     for (int yh =0;yh<ydim ;yh++){
00523       for (int xh =0;xh<xdim ;xh++){
00524
00525         val = ($\ast$this)(xh,yh,zh);
00526
00527         if (val > 0) {
00528
00529 //        dx = (xh - cm_x)*InputWidth*val/totalDensity;
00530 //        dy = (yh - cm_y)*InputWidth*val/totalDensity;
00531 //        dz = (zh - cm_z)*InputWidth*val/totalDensity;
00532           dx = (xh - cm_x)*InputWidth*val;
00533           dy = (yh - cm_y)*InputWidth*val;
00534           dz = (zh - cm_z)*InputWidth*val;
00535
00536
00537           Spread (1,1) += dx*dx/filledPoints;
00538           Spread (2,1) += dx*dy/filledPoints;
00539           Spread (2,2) += dy*dy/filledPoints;
00540           Spread (3,1) += dx*dz/filledPoints;
00541           Spread (3,2) += dy*dz/filledPoints;
00542           Spread (3,3) += dz*dz/filledPoints;
00543 //        Spread (1,1) += dx*dx;
00544 //        Spread (2,1) += dx*dy;
00545 //        Spread (2,2) += dy*dy;
00546 //        Spread (3,1) += dx*dz;
00547 //        Spread (3,2) += dy*dz;
00548 //        Spread (3,3) += dz*dz;
00549
00550         } // end if
00551       } // x
00552     } // y
00553   }
00554
```

4.27.2.5 void Envelope::Center (void)

adjust the max[xyz] and min[xyz] of the envelope to position the center of mass at the origin.

Definition at line 111 of file envelope.C.

Referenced by EnvelopeConstraint::Read().

```
00111                           {
00112
00113 #ifdef NOISY_CONSTRAINTS
00114   WARN("Centering Envelope data on (0,0,0)");
00115   cerr << "original center_of_mass: (" << r_cm_x << "," << r_cm_y << "," << r_cm_z << ")" << endl;
00116 #endif
00117   maxx -= r_cm_x;
00118   minx -= r_cm_x;
00119   maxy -= r_cm_y;
00120   miny -= r_cm_y;
00121   maxz -= r_cm_z;
00122   minz -= r_cm_z;
00123
00124   // re calc metadata
00125   Env_Setup2();
00126
00127 #ifdef NOISY_CONSTRAINTS
00128   cerr << "new      center_of_mass: (" << r_cm_x << "," << r_cm_y << "," << r_cm_z << ")" << endl;
00129   cerr << "done centering envelope." << endl;
00130 #endif
00131
```

4.27.2.6 int Envelope::Compare (const Envelope & *e*) const [inline]

default compare function.

Definition at line 190 of file envelope.h.

```
00191 {
00192   DIE("Envelope Compare not Implemented");
00193   return 0;
00194
```

4.27.2.7 void Envelope::DiffReport (ostream & *outStream*, const Envelope & *E*)

Report on differences to another env.

Definition at line 352 of file envelope.C.

Referenced by main().

```
00352                                                               {
00353
00354   if (xdim != E.xdim || ydim != E.ydim || zdim != E.zdim ){
00355     outStream << "dimensions differ:\n" << xdim << " " <<ydim << " " <<zdim  << endl
00356               << E.xdim << " " <<E.ydim << " " <<E.zdim  << endl;
00357     outStream << "aborting";
00358     return;
00359   } else {
00360     outStream << "dimensions OK " << endl;
00361   }
00362
00363   double THRESHOLD = 0.01;
00364
00365   if (fabs(r_cm_x-E.r_cm_x)>THRESHOLD ||
00366       fabs(r_cm_y-E.r_cm_y)>THRESHOLD ||
00367       fabs(r_cm_z-E.r_cm_z)>THRESHOLD ){
```

```
    outStream << "COM differs: " << endl
              << r_cm_x << " " << r_cm_y << " " << r_cm_z << endl
              << E.r_cm_x << " " << E.r_cm_y << " " << E.r_cm_z << endl;
  } else {
    outStream << "COM OK " << endl;
  }

  if (fabs(totalDensity -E.totalDensity) > 50* THRESHOLD)
    outStream << "totalDensity differs: " << totalDensity << "        " << E.totalDensity << endl;
  else {
    outStream << "totalDensity OK " << endl;
  }

  if (abs(filledPoints - E.filledPoints) > 2)
    outStream << "filledPoints differs: " << filledPoints << "        " << E.filledPoints << endl;
  else {
    outStream << "filledPoints OK " << endl;
  }

  double totalError = 0;
  double maxError = 0;
  int diff_count = 0;
  double diff;


  int zh,yh,xh;

  for (zh =0;zh<zdim ;zh++){
    for (yh =0;yh<ydim ;yh++){
      for (xh =0;xh<xdim ;xh++){
        diff = fabs ((*this)(xh,yh,zh) - E(xh,yh,zh)) ;
        totalError += diff;
        if (diff > maxError)
          maxError = diff;
        if (diff > THRESHOLD) {
          diff_count++;
        }
      }
    }
  }

```

4.27.2.8 void Envelope::Env_Setup1 ()

allocates memory.

Definition at line 63 of file envelope.C.

Referenced by Build(), Envelope(), and ReadAvsTxt().

```
00063                          {
00064
00065   density_array = new double[numPoints]; // not sparse -- saving lots of zeros here. mem wasted.  FIX
00066   //  cerr << " Allocating Envelope for "<< numPoints << endl;
00067   gradient_field = new jVector[numPoints];
00068
```

4.27.2.9 void Envelope::Env_Setup2 ()

calculates meta information about data.

Definition at line 70 of file envelope.C.

Referenced by Blur(), Center(), Envelope(), ReadAvsTxt(), and Upsize().

```
                         {

double val;
int zh, yh, xh; // blocks

filledPoints =  0;
totalDensity = cm_z = cm_y = cm_x = 0;

// first calc in blocks, then translate to real pos.
for ( zh =0;zh<zdim ;zh++){
  for ( yh =0;yh<ydim ;yh++){
    for ( xh =0;xh<xdim ;xh++){
      val = ($\ast$this)(xh,yh,zh) ;

      if (val > 0) {
        filledPoints++;
        totalDensity += val;


        cm_x += xh * val;
        cm_y += yh * val;
        cm_z += zh * val;

      }
    }
  }
}

```

4.27.2.10 void Envelope::Get_Eigenvectors (Matrix & *EigVal*, Matrix & *EigVec*) const

writes Eigenvalues and Eigenvectors matricies by reference.

Definition at line 502 of file envelope.C.

Referenced by EnvelopeConstraint::FindEnvelopeEigenvectors().

```
                                                              {

EigVec = my_EigVec;
EigVal = my_EigVal;


```

4.27.2.11 Envelope::MAKECOMPARISONS_WITH_COMPARE (Envelope)

4.27.2.12 void Envelope::NormalizeMax ()

max max density = def's MAX_ENV_DEN' value and scale all cells accordingly.

Definition at line 813 of file envelope.C.

Referenced by Blur().

```
                         {
```

```

double spread_val;
double max_grid_val = -INF;

// find max
for (int zh =0;zh<zdim ;zh++){
  for (int yh =0;yh<ydim ;yh++){
    for (int xh =0;xh<xdim ;xh++){
      spread_val = ($\ast$this)(xh,yh,zh);

      if (spread_val > max_grid_val) {
        max_grid_val = spread_val;
      }
    }
  }
}

double o_max_grid_val = MAX_ENV_DEN / max_grid_val;

// reset
for (int zh =0;zh<zdim ;zh++){
  for (int yh =0;yh<ydim ;yh++){
    for (int xh =0;xh<xdim ;xh++){
      spread_val = ($\ast$this)(xh,yh,zh);

      ($\ast$this)(xh,yh,zh) = spread_val * o_max_grid_val;
      //      cerr << spread_val << "   " << spread_val * o_max_grid_val << endl;
    }
  }
}

```

4.27.2.13 int Envelope::Output (ostream & *os*) const

Definition at line 215 of file envelope.C.

```
{

  os << "\nxdim = " << xdim
     << "\nydim = " << ydim
     << "\nzdim = " << zdim

     << "\nminx = " << minx
     << "\nminy = " << miny
     << "\nminz = " << minz

     << "\nmaxx = " << maxx
     << "\nmaxy = " << maxy
     << "\nmaxz = " << maxz << endl;

  os <<   "\nnumPoints = "  << numPoints
     <<   "\nInputWidth = " << InputWidth << endl<< endl;

  int zh, yh, xh;
  for (zh =0;zh<zdim ;zh++){
    for (yh =0;yh<ydim ;yh++){
      for (xh =0;xh<xdim ;xh++){
        os << ($\ast$this)(xh,yh,zh) << " ";
      }
        os << endl;
    }
    os << "--" << endl;
  }
```

```
00243
```

4.27.2.14 int Envelope::OutputBinary (ostream & *os*) const

Definition at line 424 of file envelope.C.

```
00425 {
00426   // NOT IMPLEMENTED
00427   //  os <= supportA <= measured <= variance;
00428   return 2;
00429
```

4.27.2.15 double Envelope::QueryDensity (const jVector & *pos*) const

returns density at ‘pos.

Definition at line 849 of file envelope.C.

```
00849                                                {
00850
00851   int xh = (int)((pos[0] - minx)/InputWidth);
00852   int yh = (int)((pos[1] - miny)/InputWidth);
00853   int zh = (int)((pos[2] - minz)/InputWidth);
00854
00855   return ($\ast$this)(xh,yh,zh);
00856   return 0;
00857
```

4.27.2.16 jVector Envelope::QueryGradient (const jVector & *pos*) const

4.27.2.17 int Envelope::Read (istream & *is*)

Definition at line 418 of file envelope.C.

Referenced by EnvelopeConstraint::Read(), and main().

```
00419 {
00420   // assume avs file now, later check here and call appropriate read-in function.
00421   return ReadAvsTxt(is);
00422
```

4.27.2.18 int Envelope::ReadAvsTxt (istream & *inStream*)

read text-converted AVS density format.

Definition at line 1032 of file envelope.C.

Referenced by Read().

```
01032                                           {
01033
01034
01035   //  cerr << "------------ READ Envelope start    --------" << endl;
01036
01037   if (density_array) {
```

```
    DIE("Envelope Read Error! It appears you have already read in density data into this envelope!");
  }

  char d[255];  // used for reading header lines

  inStream.getline(d,255); // # AVS
  inStream.getline(d,255); // ndim

  //dim1
  inStream.ignore(5);
  inStream >>  xdim;
  //dim2
  inStream.ignore(6); // +\n
  inStream >>  ydim;
  //dim3
  inStream.ignore(6); // +\n
  inStream >>  zdim;

  //  cerr << "dim: " << xdim << ", "<< ydim << ", "<< zdim << endl;

  inStream.getline(d,255); // \n

  inStream.getline(d,255); // nspace
  inStream.getline(d,255); // vclean
  inStream.getline(d,255); // data
  inStream.getline(d,255); // field

  // mins
  inStream.ignore (9);
  inStream >>  minx >> miny >> minz;

  //  cerr << "min: " <<  minx << ", " << miny << ", " << minz << endl;

  // maxs
  inStream.ignore (10); // +\n
  inStream >>  maxx >> maxy >> maxz;

  //  cerr << "max: "  <<  maxx << ", " << maxy << ", " << maxz << endl;

  inStream.getline(d,255); // nspace
  inStream.getline(d,255); // # end of header

  InputWidth = (maxx-minx)/xdim;

  if (InputWidth > 50 || InputWidth < 0.01){
    cerr << "Input Grid Width is "<< InputWidth << " :Invalid.  Quitting." << endl;
    DIE("");
  }

  if ((fabs(InputWidth - (maxy-miny)/ydim) > 0.01) ||
      (fabs(InputWidth - (maxz-minz)/zdim) > 0.01)) {
    cerr << "AVS Input is not cubic" << endl;
    cerr << "dim: " << xdim << ", "<< ydim << ", "<< zdim << endl;
    cerr << "min: " <<  minx << ", " << miny << ", " << minz << endl;
    cerr << "max: "  <<  maxx << ", " << maxy << ", " << maxz << endl;
    DIE("");
  }

  double val;
  int readCount = 0;
  //  jVector Pos(3,0.0);

  numPoints = xdim*ydim*zdim;

  if (numPoints >= INT_MAX || numPoints < 0){
```

```
    cerr << "Envelope size error!\n\n Envelope Class uses signed ints to index data, so "<< INT_MAX << " is the maximum number of cells possible.  Your number of cells is "<< numPoints << ", which is too large."<< endl;
    DIE("");
  }

  this->Env_Setup1();

  int zh, yh, xh;
  for ( zh =0;zh<zdim ;zh++){
    for ( yh =0;yh<ydim ;yh++){
      for ( xh =0;xh<xdim ;xh++){
        inStream >> val;

        if (val > 0) {
          totalDensity += val;
        }

        if (!inStream.good()){
          cerr << "Envelope reading error!\n\n read "<< readCount << " entries, expected "<< numPoints << endl;
          DIE("");
        }

        ($\ast$this)(xh,yh,zh) = val;             // not sparse -- saving lots of zeros here. mem wasted.  FIX
        readCount++;
      }
    }
  }

```

4.27.2.19 int Envelope::ReadBinary (istream & *is*)

Definition at line 431 of file envelope.C.

```
00432 {
00433   // NOT IMPLEMENTED
00434   //  is >= supportA[0] >= supportA[1] >= measured >= variance;
00435   return 2;
00436
```

4.27.2.20 void Envelope::Upsize (int *k*)

increases the box size by a factor k'.

Definition at line 569 of file envelope.C.

Referenced by EnvelopeConstraint::PreSetup().

```
00569                        {
00570
00571 //   if (k != 2 && k != 3 && k!=4){
00572 //     DIE("upsize k factor must be 2,3 or 4");
00573 //   }
00574    if (k != 2){
00575      DIE("upsize k factor must be 2");
00576    }  // 3 and 4 require more login in resetting real bounds.
00577
00578   int new_xdim,new_ydim,new_zdim;
00579   // figure out new bounds
00580
00581   int err;
00582   err = (xdim%k ? 1: 0);
00583   new_xdim = xdim/k+err;
```

```
  err = (ydim%k ? 1: 0);
  new_ydim = ydim/k+err;

  err = (zdim%k ? 1: 0);
  new_zdim = zdim/k+err;


  int new_numPoints = new_xdim * new_ydim * new_zdim;
  double new_InputWidth = k * InputWidth;

  // begin: this whole section is more complex for k = 3 or 4
  //  double new_minx, new_miny, new_minz;
  double new_maxx, new_maxy, new_maxz;

  new_maxx = minx + new_InputWidth*new_xdim;
  new_maxy = miny + new_InputWidth*new_ydim;
  new_maxz = minz + new_InputWidth*new_zdim;
  // end:   this whole section is more complex for k = 3 or 4

  double * new_density_array = new double[new_numPoints]; // not sparse
  for (int l=0;l<new_numPoints;l++){new_density_array[l] =0;}
  jVector * new_gradient_field = new jVector[new_numPoints];
  for (int l=0;l<new_numPoints;l++){new_gradient_field[l].zero();}

  double old_val;  // old density value

  // cycle over old array, processing cells with data
  int zh,yh,xh;
  int new_index;
  int new_index_x, new_index_y, new_index_z;
  for (zh =0;zh<zdim ;zh++){
    //    new_index_z = (zh/k+(zh%k?1:0))*new_ydim*new_zdim;
    new_index_z = (zh/k)*new_ydim*new_xdim;
    for (yh =0;yh<ydim ;yh++){
      //      new_index_y = (yh/k+(yh%k?1:0))*new_xdim;
      new_index_y = (yh/k)*new_xdim;
      for (xh =0;xh<xdim ;xh++){
        //      new_index_x = xh/k+(xh%k?1:0);
        new_index_x = xh/k;
        old_val = ($\ast$this)(xh,yh,zh);

//      cerr << " xh=" << xh
//           << " yh=" << yh
//           << " zh=" << zh << endl;
//
//      cerr << " new_index_x=" << new_index_x
//           << " new_index_y=" << new_index_y
//           << " new_index_z=" << new_index_z << endl;
//
//      int aa = (xh)+(yh)*xdim+(zh)*xdim*ydim ;
//      cerr << " old index = " << aa << " of " << numPoints << endl;
//
        new_index = new_index_x+new_index_y+new_index_z;
//      cerr << " new index = " << new_index << " of " << new_numPoints << endl << endl;
//

        if (old_val > 0) {
          old_val = old_val / (k*k*k);

          if (new_index > new_numPoints) {
            DIE ("ACK");
          }
          new_density_array[new_index] += old_val;
        }
```

```
      }
    }
  }

  maxx = new_maxx;
  maxy = new_maxy;
  maxz = new_maxz;

//   minx = new_minx;
//   miny = new_miny;
//   minz = new_minz;

  xdim = new_xdim;
  ydim = new_ydim;
  zdim = new_zdim;

  if (density_array && gradient_field) {
    delete [] density_array;
    density_array = new_density_array;
    new_density_array = 0;

    delete [] gradient_field;
    gradient_field = new_gradient_field;  // empty
    new_gradient_field = 0;

    InputWidth = new_InputWidth;
    numPoints = new_numPoints;

  } else {
    DIE("Called Upsize on Envelope without intializing internal datastructures.");
  }

```

4.27.2.21 int Envelope::WriteAvsTxt (ostream & *outStream*) const

write AVS density format(as text).

Definition at line 1003 of file envelope.C.

Referenced by jdOcc_Count::run().

```
01003                                                 {
01004
01005   outStream << "envelope: # AVS field file" << endl;
01006   outStream << "ndim=3" << endl;
01007   outStream << "xdim= " << xdim << endl;
01008   outStream << "ydim= " << ydim << endl;
01009   outStream << "zdim= " << zdim << endl;
01010   outStream << "nspace=3" << endl;
01011   outStream << "veclen=1" << endl;
01012   outStream << "data=float" << endl;
01013   outStream << "field=uniform" << endl;
01014   outStream << "min_ext= " << minx << " "  << miny  << " " << minz << endl;
01015   outStream << "max_ext= " << maxx  << " " << maxy  << " " << maxz << endl;
01016   outStream << "# end of header" << endl;
01017
01018   int zh, yh, xh;
01019   for (zh =0;zh<zdim ;zh++){
01020     for (yh =0;yh<ydim ;yh++){
01021       for (xh =0;xh<xdim ;xh++){
01022         outStream << ($\ast$this)(xh,yh,zh) << " ";
01023
01024       }
```

```
01025     }
01026   }
01027
```

4.27.2.22 int Envelope::WriteDensity (ostream & *outStream*, const Molecule & *Mol*) const

write out a list of atoms and their density in the env.

Definition at line 988 of file envelope.C.

Referenced by jdOcc_Count::run().

```
00988                                                                      {
00989
00990   jVector center;
00991
00992   int n = Mol.Number();
00993   for (int i=0;i<n;i++){
00994
00995     center = Mol(i);
00996     outStream << ($\ast$this)(center)  << ": " << i << endl;
00997
00998   }
00999
```

4.27.2.23 int Envelope::WriteVRML (ostream & *outStream*) const

write VRML 1.0 format.

Definition at line 859 of file envelope.C.

Referenced by EnvelopeConstraint::WriteVRML_OrigData(), and jdOcc_Count::run().

```
00859                                                  {
00860
00861   outStream << "#VRML V1.0 ascii" << endl;
00862
00863   double x,y,z;
00864   double i_half = InputWidth/2;
00865   double val, t;
00866
00867   int zh, yh, xh;
00868   for (zh =0;zh<zdim ;zh++){
00869     for (yh =0;yh<ydim ;yh++){
00870       for (xh =0;xh<xdim ;xh++){
00871         val = ($\ast$this)(xh,yh,zh);
00872
00873
00874         if (val> 0) {
00875           t  = 0.5-val/2;
00876
00877           x = minx + i_half + xh * InputWidth;
00878           y = miny + i_half + yh * InputWidth;
00879           z = minz + i_half + zh * InputWidth;
00880
00881
00882
00883           outStream << "Separator{" << endl;
00884           if (val > .9)
00885             outStream << "Material{specularColor 0.8 0.8 0.8 diffuseColor 1 0.2 0.2 transparency 0.1" \
```

```
                    <<"}" << endl;
        else if (val > .5)
          outStream << "Material{specularColor 0.5 0.5 0.5 diffuseColor 0.2 0.8 0.2 transparency " \
                    << t <<"}" << endl;
        else
          outStream << "Material{specularColor 0.2 0.2 0.2 diffuseColor 0.3 0.3 0.3 transparency " \
                    << t <<"}" << endl;

        outStream << "Transform{translation "<< x << " " << y << " "\
                  << z << " " <<"}" << endl;
        outStream << "Cube{width " << InputWidth << " height " \
                  << InputWidth << " depth " << InputWidth <<"}" << endl;
        outStream << "}" << endl;

      }

    }
  }
}

```

4.27.2.24 int Envelope::WriteVRMLgr (ostream & *outStream*) const

write VRML 1.0 format with gradient arrows.

Definition at line 910 of file envelope.C.

```
00910                                                     {
00911
00912   outStream << "#VRML V1.0 ascii\nSeparator{" << endl;
00913
00914   double x,y,z;
00915   double xr,yr,zr, rad;
00916   double i_half = InputWidth/2;
00917   double val, t;
00918   jVector G;
00919   int scale;
00920
00921
00922   int zh, yh, xh;
00923   for (zh =0;zh<zdim ;zh++){
00924     for (yh =0;yh<ydim ;yh++){
00925       for (xh =0;xh<xdim ;xh++){
00926         val = ($\ast$this)(xh,yh,zh);
00927         G = gradient(xh,yh,zh);
00928
00929         if (val> 0) {
00930           t  = 0.5-val/2;
00931
00932           x = minx + i_half + xh * InputWidth;
00933           y = miny + i_half + yh * InputWidth;
00934           z = minz + i_half + zh * InputWidth;
00935
00936           xr = G[2];
00937           yr = 0;
00938           zr = -1 * G[0];
00939           if((rad = G.mag())) // sic
00940             rad =  acos (G[1]/(rad));
00941
00942           scale = ((int) i_half)+1;
00943
00944           outStream << " Separator{" << endl;
00945
```

```
          outStream << "  Transform{translation "<< x << " " << y << " "\
                    << z << " " <<"}" << endl;

          outStream << "  Separator{" << endl;
          if (val > .9)
            outStream << "   Material{specularColor 0.8 0.8 0.8 diffuseColor 1 0.2 0.2 transparency 0.1" \
                      <<"}" << endl;
          else if (val > .5)
            outStream << "   Material{specularColor 0.5 0.5 0.5 diffuseColor 0.2 0.8 0.2 transparency " \
                      << t <<"}" << endl;
          else
            outStream << "   Material{specularColor 0.2 0.2 0.2 diffuseColor 0.3 0.3 0.3 transparency " \
                      << t <<"}" << endl;

          outStream << "   Cube{width " << InputWidth << " height " \
                    << InputWidth << " depth " << InputWidth <<"}" << endl;
          outStream << "  }" << endl;

          outStream << "  Separator{" << endl;

          outStream << "   Material {diffuseColor 0 0 1} " << endl;
          outStream << "   Transform {rotation " << xr << " "<< yr << " "<< zr <<" "<< rad <<"}" << endl;
          outStream << "   # gradient vector =" << G   << endl;
          outStream << "   Scale {scaleFactor " << scale <<" " << scale <<" " << scale <<"}" << endl;
          outStream << "   Cylinder {radius  .2 height  1 }" << endl;
          outStream << "   Transform {translation 0 1 0}" << endl;
          outStream << "   Cone { parts SIDES bottomRadius .2 height 1}" << endl;

          outStream << "  }" << endl;

          outStream << " }" << endl;

        }

      }
    }
  }

  outStream << "}
```

4.27.2.25 jVector & Envelope::gradient (int x, int y, int z)

lvalue (for assigning gradient).

Definition at line 135 of file envelope.C.

```
00135                                       {
00136   if ((x < 0) || (x >= xdim) ||
00137       (y < 0) || (y >= ydim) ||
00138       (z < 0) || (z >= zdim))  {
00139     cerr << "Envelope gradient access out of bounds.  I got "
00140          << x << ", "<< y << ", "<< z << endl;
00141     raise (SIGQUIT);
00142
00143   }
00144
```

4.27.2.26 const jVector & Envelope::gradient (jVector J) const

access gradient.

Definition at line 164 of file envelope.C.

```
00164                                   {
00165   int x,y,z;
00166 
00167   // check this
00168   x = (int) floor((J[0]-minx)/InputWidth);
00169   y = (int) floor((J[1]-miny)/InputWidth);
00170   z = (int) floor((J[2]-minz)/InputWidth);
00171 
00172   return gradient(x,y,z);
00173 
```

4.27.2.27 const jVector & Envelope::gradient (int *x*, int *y*, int *z*) const

access gradient by cell.

Definition at line 150 of file envelope.C.

Referenced by WriteVRMLgr(), EnvelopeConstraint::getGradient(), and gradient().

```
00150                                             {
00151   if ((x < 0) || (x >= xdim) ||
00152       (y < 0) || (y >= ydim) ||
00153       (z < 0) || (z >= zdim))  {
00154 //     cerr << "Envelope const gradient access out of bounds.  I got "
00155 //       << x << ", "<< y << ", "<< z << endl;
00156 //     raise (SIGQUIT);
00157     return JVzero3;
00158   }
00159 
```

4.27.2.28 double & Envelope::operator() (int *x*, int *y*, int *z*)

lvalue (for assigning density).

Definition at line 178 of file envelope.C.

```
00178                                         {
00179   if ((x < 0) || (x >= xdim) ||
00180       (y < 0) || (y >= ydim) ||
00181       (z < 0) || (z >= zdim))  {
00182     cerr << "Tried to assign to Envelope out of bounds.  I got "
00183          << x << ", "<< y << ", "<< z << endl;
00184     DIE("");
00185   }
00186 
```

4.27.2.29 const double Envelope::operator() (const jVector & *J*) const

access density by position.

Definition at line 203 of file envelope.C.

```
00203                                            {
00204   int x,y,z;
00205 
00206   // check this
```

```
  x = (int) floor((J[0]-minx)/InputWidth);
  y = (int) floor((J[1]-miny)/InputWidth);
  z = (int) floor((J[2]-minz)/InputWidth);

  return ($\ast$this)(x,y,z);


```

4.27.2.30 const double Envelope::operator() (int x, int y, int z) const

access density by cell.

Definition at line 190 of file envelope.C.

```
00190                                               {
00191   if ((x < 0) || (x >= xdim) ||
00192       (y < 0) || (y >= ydim) ||
00193       (z < 0) || (z >= zdim))  {
00194 //     cerr << "Envelope const access out of bounds.  I got "
00195 //       << x << ", "<< y << ", "<< z << endl;
00196 //     raise (SIGQUIT);
00197     return MIN_ENV_DEN;
00198   }
00199
```

4.27.3 Friends And Related Function Documentation

4.27.3.1 class EnvelopeConstraint [friend]

Definition at line 34 of file envelope.h.

4.27.4 Member Data Documentation

4.27.4.1 double Envelope::InputWidth

Real width of cubes (e.g. 3.7A).

Definition at line 106 of file envelope.h.

4.27.4.2 const jVector Envelope::JVzero3

useful null vector.

Definition at line 180 of file envelope.h.

4.27.4.3 double Envelope::cm_x

Center of mass, weighted by probabilities, (as blocks).

Definition at line 119 of file envelope.h.

4.27.4.4 double Envelope::cm_y

Center of mass, weighted by probabilities, (as blocks).

Definition at line 119 of file envelope.h.

4.27.4.5 double Envelope::cm_z

Center of mass, weighted by probabilities, (as blocks).

Definition at line 119 of file envelope.h.

4.27.4.6 double * Envelope::density_array

Array of doubles storing density.

Definition at line 94 of file envelope.h.

4.27.4.7 int Envelope::filledPoints

Number of non-zero cubes in Envelope.

Definition at line 128 of file envelope.h.

4.27.4.8 jVector * Envelope::gradient_field

Array of jVectors storing gradient.

Definition at line 97 of file envelope.h.

4.27.4.9 double Envelope::maxx

Real position of max edges of region.

Definition at line 112 of file envelope.h.

4.27.4.10 double Envelope::maxy

Real position of max edges of region.

Definition at line 112 of file envelope.h.

4.27.4.11 double Envelope::maxz

Real position of max edges of region.

Definition at line 112 of file envelope.h.

4.27.4.12 double Envelope::minx

Real position of min edges of region.

Definition at line 109 of file envelope.h.

4.27.4.13 double Envelope::miny

Real position of min edges of region.

Definition at line 109 of file envelope.h.

4.27.4.14 double Envelope::minz

Real position of min edges of region.

Definition at line 109 of file envelope.h.

4.27.4.15 Matrix Envelope::my_EigVal

Eigenvalues for the envelope.

Definition at line 131 of file envelope.h.

4.27.4.16 Matrix Envelope::my_EigVec

Eigenvectors for the envelope.

Definition at line 134 of file envelope.h.

4.27.4.17 int Envelope::numPoints

Number of cubes in Envelope.

Definition at line 100 of file envelope.h.

4.27.4.18 double Envelope::r_cm_x

Center of mass, weighted by probabilities, (Real postition!).

Definition at line 122 of file envelope.h.

4.27.4.19 double Envelope::r_cm_y

Center of mass, weighted by probabilities, (Real postition!).

Definition at line 122 of file envelope.h.

4.27.4.20 double Envelope::r_cm_z

Center of mass, weighted by probabilities, (Real postition!).

Definition at line 122 of file envelope.h.

4.27.4.21 double Envelope::totalDensity

Total of all the density numbers in the cubes.

Definition at line 125 of file envelope.h.

4.27.4.22 int Envelope::xdim

Dimensions in number of cubes.

Definition at line 103 of file envelope.h.

4.27.4.23 int Envelope::ydim

Dimensions in number of cubes.

Definition at line 103 of file envelope.h.

4.27.4.24 int Envelope::zdim

Dimensions in number of cubes.

Definition at line 103 of file envelope.h.

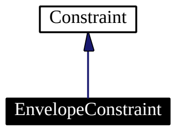
The documentation for this class was generated from the following files:

- **envelope.h**
- **envelope.C**

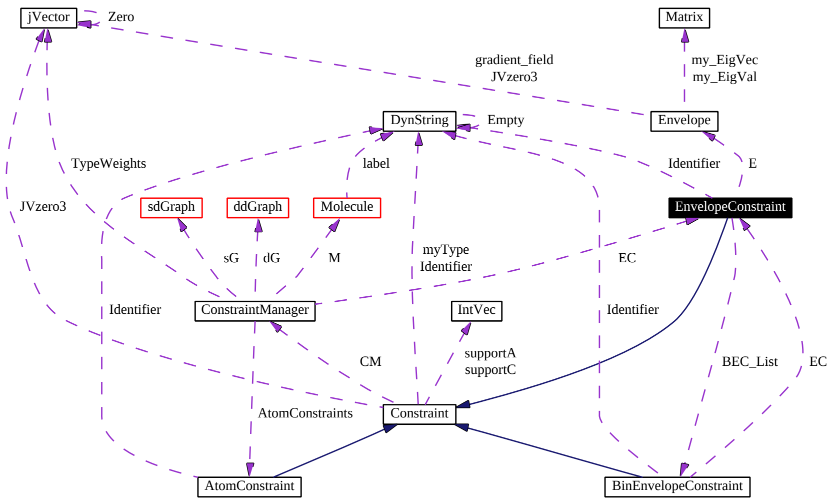
4.28 EnvelopeConstraint Class Reference

#include <envelope_constraint.h>

Inheritance diagram for EnvelopeConstraint



Collaboration diagram for EnvelopeConstraint:



Public Methods

- **EnvelopeConstraint** (**ConstraintManager** &cm)

  constructor.

- ~**EnvelopeConstraint** ()
- **jVector dStress** (int k, const **jVector** &pos) const
- double **getDensity** (const **jVector** &J) const

  return the density at **jVector** (p. 325) *J.*

- **jVector getGradient** (const **jVector** &J) const

  return the gradient at **jVector** (p. 325) *J.*

- int **Compare** (const EnvelopeConstraint &ec) const
- **MAKECOMPARISONS_WITH_COMPARE** (EnvelopeConstraint)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const
- int **WriteVRML_OrigData** (ostream &outStream) const

  write VRML 1.0 format for initial envelope data (env_A).

- void **MoveModelbyGrad** (double d) const

  move model distance d according to gradient of atoms outsie E[0].

- void **AlignModeltoCOM** () const

  calling this translates & rotates all atoms in the model.

- void **FindEnvelopeEigenvectors** (**Matrix** &EigVec, **Matrix** &EigVal) const

  Determine the EigenVectors for the ""A"" envelope.

- **Matrix FindModelRotation** () const

  Determine the roation matrix.

- void **AlignPrincComp** (void) const

  Rotate current model via pricipal components.

- **jVector FindModelTranslationToCOM** () const

  Determine the translation of model points to COM of envelope.

- double **computeStress** () const

  returns stress on constraint.

- double **computeStress** (int k, const **jVector** &pos) const

  returns stress on constraint if atom k' were in position pos'.

- int **Read** (istream &is)
- int **ReadBinary** (istream &is)

  Same scenerio as **Read**() (p. 236) *only binary version.*

- int **ReadAvsTxt** (istream &in)

  read text-converted AVS density format.

- void **PreSetup** ()

  aligns the grid.

- void **PostSetup** ()

Public Attributes

- **Envelope∗∗ E**

  Array of envelope pointers. [0] points to read-in data, rest are blurred copies.

- double **K**

  factor used to combine E[0], E[1], etc. when they overlap.

- **BinEnvelopeConstraint∗∗ BEC_List**

  array of pointers to BEC's; for access.

Static Public Attributes

- **DynString Identifier** = "envelope:"
- int **num_evelopes** = 2

  Number of envelopes to use. 0 is original data, 1..n-1 are blurred versions.

- double **translation_threshold** = 0.5

  If the translate vector nrm() > this, then a translation occurs.

- double **rotation_threshold** = 0.08

  If the sum(abs(Rotation Matrix)) > this, then a rotation occurs.

- double **density_threshold** = 0.125

  greater than this => atom fits envelope.

- int **blur_kernelSize** = 2

  size of blur kernel.

Friends

- class **BinEnvelopeConstraint**

4.28.1 Constructor & Destructor Documentation

4.28.1.1 EnvelopeConstraint::EnvelopeConstraint (ConstraintManager & *cm*)

constructor.

Definition at line 4 of file envelope_constraint.C.

```
00005   : Constraint(cm,IntVec(cm.numAtoms(),0,1)) // range constructor for intvec
00006   : Constraint(cm)
00007   // NOTE NO SUPPORTS -- will support all the BinEnvConstratints after they're added
00008 {
00009
00010   E = new Envelope*[EnvelopeConstraint::num_evelopes];
00011
00012   E[0] = new Envelope;
00013   for(int i =1;i<EnvelopeConstraint::num_evelopes;i++)
00014     E[i] =0;
00015
00016   BEC_List = 0;
00017   K = 1;
00018
```

4.28.1.2 EnvelopeConstraint::~EnvelopeConstraint ()

Definition at line 22 of file envelope_constraint.C.

```
00022                                         {
00023
00024   for(int i =0;i<EnvelopeConstraint::num_evelopes;i++){
00025     if (E[i])
```

```
00026       delete E[i];
00027   }
00028
```

4.28.2 Member Function Documentation

4.28.2.1 void EnvelopeConstraint::AlignModeltoCOM (void) const

calling this translates & rotates all atoms in the model.

Definition at line 167 of file envelope_constraint.C.

Referenced by computeStress().

```
00167                                                   {
00168
00171
00172   // Translate to COM
00173   jVector T = FindModelTranslationToCOM();
00174   if (T.nrm() > translation_threshold){
00175     CM->TranslateModel(T);
00176 #ifdef NOISY_CONSTRAINTS
00177     cerr << "finished translate by " << T << endl;
00178   } else {
00179     WARN ("not translated -- too close");
00180 #endif
00181   }
00182
```

4.28.2.2 void EnvelopeConstraint::AlignPrincComp (void) const

Rotate current model via pricipal components.

Definition at line 85 of file envelope_constraint.C.

Referenced by AlignModeltoCOM().

```
00085                                                  {
00086
00087   jVector tempV1;
00088   jVector tempV2;
00089   int num = CM->numAtoms();
00090
00091
00092   // Rotate
00093   Matrix Rot(3,3);
00094   Rot = FindModelRotation();
00095
00096   Matrix I(3,3);
00097   I.identity();
00098
00099   if ((Rot - I).SumAbsoluteValue() > rotation_threshold){
00100
00101     for (int a = 0; a < num; a++){
00102       if (CM->constraintON(a)){
00103         for (int i =0;i<3;i++){   //matrix multiply into a jVector
00104
00105           tempV1 = CM->getPosition(a);
00106
00107           //    tempV2[i] =0;
00108           //    for (int j =0;j<3;j++){
```

```
00109           //      tempV2[i] += Rot(i+1,j+1)*tempV1[j];
00110           //    }
00111
00112           tempV2[i]  = Rot(i+1,1)*tempV1[0];
00113           tempV2[i] += Rot(i+1,2)*tempV1[1];
00114           tempV2[i] += Rot(i+1,3)*tempV1[2];
00115
00116         }  // end matrix multiply
00117         CM->setPosition(a, tempV2);
00118       }
00119     } // end loop on num' atoms
00120
00121
00122     //   } else {
00123     // do not move if matrix is almost Identitiy
00124     //     cerr << "Rotate Aborted -- too close to Identity." << endl;
00125
00126   }
00127
```

4.28.2.3 int EnvelopeConstraint::Compare (const EnvelopeConstraint & *ec*) const `[inline]`

Definition at line 154 of file envelope_constraint.h.

```
00155 {
00156   DIE("Envelope Compare not Implemented");
00157
00158   return 0;
00159
```

4.28.2.4 void EnvelopeConstraint::FindEnvelopeEigenvectors (Matrix & *EigVal*, Matrix & *EigVec*) const

Determine the EigenVectors for the ""A"" envelope.

Definition at line 324 of file envelope_constraint.C.

Referenced by FindModelRotation().

```
00324                                                                                        {
00325
00326   E[0]->Get_Eigenvectors(EigVal, EigVec);
00327
00328
```

4.28.2.5 Matrix EnvelopeConstraint::FindModelRotation (void) const

Determine the roation matrix.

Definition at line 352 of file envelope_constraint.C.

Referenced by AlignPrincComp().

```
00352                                                {
00353
00354   Matrix Spread(3,3); //lower triangular matrix of 3x3
00355
```

```
00356   Matrix R; //rotation matrix
00357
00358   jVector tempV;
00359
00360   int in_Points = CM->numAtomsON();
00361
00362   // Calc spread
00363   double dx, dy, dz = 0;
00364   double meanx = 0, meany = 0, meanz = 0;
00365
00366   for (int a = 0; a < in_Points; a++){
00367     if (CM->constraintON(a)){
00368       tempV = CM->getPosition(a);
00369
00370       meanx += tempV[0];
00371       meany += tempV[1];
00372       meanz += tempV[2];
00373     }
00374   }
00375   meanx /= in_Points;
00376   meany /= in_Points;
00377   meanz /= in_Points;
00378
00379   // calc dx dy dz
00380
00381   Spread.zero();
00382
00383
00384   for (int a = 0; a < in_Points; a++){
00385     tempV = CM->getPosition(a);
00386     dx = tempV[0] - meanx;
00387     dy = tempV[1] - meany;
00388     dz = tempV[2] - meanz;
00389
00390     Spread (1,1) += dx*dx/in_Points;
00391     Spread (2,1) += dx*dy/in_Points;
00392     Spread (2,2) += dy*dy/in_Points;
00393     Spread (3,1) += dx*dz/in_Points;
00394     Spread (3,2) += dy*dz/in_Points;
00395     Spread (3,3) += dz*dz/in_Points;
00396   }
00397
```

4.28.2.6 jVector EnvelopeConstraint::FindModelTranslationToCOM (void) const

Determine the translation of model points to COM of envelope.

Definition at line 331 of file envelope_constraint.C.

Referenced by AlignModeltoCOM().

```
00331                                                         {
00332   jVector rv;
00333   jVector model_center = CM->molecule_center();
00334
00335   rv[0] = E[0]->r_cm_x - model_center[0];
00336   rv[1] = E[0]->r_cm_y - model_center[1];
00337   rv[2] = E[0]->r_cm_z - model_center[2];
00338
00339 #ifdef NOISY_CONSTRAINTS
00340   cerr <<"Model COM: "<< model_center << endl;
00341
00342   cerr <<"Env COM: " << E[0]->r_cm_x <<" " << E[0]->r_cm_y <<" "<< E[0]->r_cm_z << endl;
```

```
  cerr << "Translation vector: "  << rv << " (mag=" << rv.mag() << ")" <<endl<< endl;
#endif
  return rv;
```

4.28.2.7 EnvelopeConstraint::MAKECOMPARISONS_WITH_COMPARE (EnvelopeConstraint)

4.28.2.8 void EnvelopeConstraint::MoveModelbyGrad (double *d*) const

move model distance d according to gradient of atoms outsie E[0].

Definition at line 132 of file envelope_constraint.C.

Referenced by AlignModeltoCOM().

```
00132                                                       {
00133
00136
00137   jVector dir; dir.zero();
00138   // cycle over my BEC, asking them if their atom is outside
00139   // if so, take position of point, determine gradient there,
00140   // and += to dir.
00141   //  move model by d*dir.unit();
00142
00143   jVector pos;
00144   int num = CM->numAtoms(); // ASSUMES all atoms in CM are in env. <?>
00145   for (int i = 0; i < num; i++){
00146     if (CM->constraintON(i)){
00147       //    cerr << "atom = " << i << endl;
00148       //      pos = CM->getPosition(BEC_List[i]->Atom_id);
00149       pos = CM->getPosition(i);
00150       if ((*E[0])(pos) <= EnvelopeConstraint::density_threshold) {// outside orig data
00151         //      cerr << "location = " << pos << endl;
00152         //      cerr << "gradient = " << getGradient(pos) << endl;
00153         dir += getGradient(pos);
00154       }
00155     }
00156   }
00157
```

4.28.2.9 int EnvelopeConstraint::Output (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 439 of file envelope_constraint.C.

```
{
  /*
    os << supportA[0] << ' ' << supportA[1] << ' '
    << measured << ' ' << variance;
    return 2;
  */
  return 0;
```

4.28.2.10 int EnvelopeConstraint::OutputBinary (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 458 of file envelope_constraint.C.

```
00459 {
00460   //  os <= supportA <= measured <= variance;
00461   return 2;
00462
```

4.28.2.11 void EnvelopeConstraint::PostSetup (void) [virtual]

constraints get their ID when added to the constraint graph. this addition is done in the Input function between the pre- and post setup functions. constraint setup that needs to occur with an ID and an added constraint go in post setup. (like adding parent constraints, etc.)

Reimplemented from **Constraint** (p. 84).

Definition at line 57 of file envelope_constraint.C.

```
00057                                   {
00058
00059   BinEnvelopeConstraint * BECptr;
00060
00061   int num = CM->numAtoms(); // ASSUMES all atoms in CM are in env. <?> FIX
00062   BEC_List = new BinEnvelopeConstraint * [num];
00063
00064   for (int i = 0; i < num; i++){
00065
00066     BECptr = new BinEnvelopeConstraint(*CM,i,this);
00067     // to be deleted by ~ConstraintManager
00068
00069     BEC_List[i] = BECptr; // record it.
00070
00071     BECptr->Input(cin); // INPUT STREAM NOT USED (is there a null stream?)
00072
00073     // each bin_envelope is supported by the envelope_constraint
00074     // When the envelope_constraint is "valid" the atoms have been
00075     // aligned ok against the envelope.  Asserting the highlevel EnvelopeConstraint
00076     // invalid will force further alignment
00077
00078   }
00079
```

4.28.2.12 void EnvelopeConstraint::PreSetup (void) [virtual]

aligns the grid.

Reimplemented from **Constraint** (p. 85).

Definition at line 36 of file envelope_constraint.C.

```
00036                                  {
00037
00038   // create blurred envelopes
00039
00040   if (E[0]->numPoints > 1) {
00041
00042     for(int i =1;i<EnvelopeConstraint::num_evelopes;i++){
```

```
00043       E[i] = new Envelope(*E[i-1]); // copy
00044       E[i]->Blur(1);                // FIX -- make a variable in globals
00045       E[i]->Upsize(2);
00046     }
00047
00048   } else {
00049     cerr << " Envelope setup failed.  Envelope has " << E[0]->numPoints << " points." << endl;
00050     DIE("");
00051   }
00052
```

4.28.2.13 int EnvelopeConstraint::Read (istream & *is*) `[virtual]`

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 236) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 236) function, the derived class should ∗not∗ log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented from **Constraint** (p. 85).

Definition at line 449 of file envelope_constraint.C.

```
00450 {
00451   // tell A to read away!
00452   int rv = E[0]->Read(is);
00453   E[0]->Center();
00454   return rv;
00455
00456
```

4.28.2.14 int EnvelopeConstraint::ReadAvsTxt (istream & *in*)

read text-converted AVS density format.

4.28.2.15 int EnvelopeConstraint::ReadBinary (istream & *is*) `[virtual]`

Same scenerio as **Read**() (p. 236) only binary version.

Reimplemented from **Constraint** (p. 85).

Definition at line 464 of file envelope_constraint.C.

```
00465 {
00466   //  is >= supportA[0] >= supportA[1] >= measured >= variance;
00467   return 2;
00468
```

4.28.2.16 int EnvelopeConstraint::WriteVRML_OrigData (ostream & *outStream*) const `[inline]`

write VRML 1.0 format for initial envelope data (env_A).

Definition at line 162 of file envelope_constraint.h.

Referenced by jdErr_Eval::run().

```
00162                                                                            {
00163   return E[0]->WriteVRML(outStream);
00164
```

4.28.2.17 double EnvelopeConstraint::computeStress (int *k*, const jVector & *pos_k*) const [virtual]

returns stress on constraint if atom k' were in position pos'.

Reimplemented from **Constraint** (p. 89).

Definition at line 225 of file envelope_constraint.C.

```
00226 {
00227
00228   DIE("");
00229     return 0;
00230
00231
00232
```

4.28.2.18 double EnvelopeConstraint::computeStress () const [virtual]

returns stress on constraint.

Reimplemented from **Constraint** (p. 89).

Definition at line 210 of file envelope_constraint.C.

```
00210                                     {
00211
00212   // align centroid
00213
00214   // calc rotation matirix
00215
00216   // rotate atoms
00217
00218   AlignModeltoCOM();
00219   return 0;
00220
00221
```

4.28.2.19 jVector EnvelopeConstraint::dStress (int *k*, const jVector & *pos_k*) const [virtual]

asking constraint number k for its 12 element dStress vector we pass in pos_a vector – corresponding to the atom position if the constraint is an atom, and corresponding to the position of the atom k in question if the call is to a non-atom constraint. EG for a distance constraint atom 12 needs the dStress from the d-constraint, so it will call dStress on the d-constraint(12, atom_pos) so the d_constr can answer the question if atom 12 were in atom_pos, I would say my dStress is blah.

Reimplemented from **Constraint** (p. 89).

Definition at line 235 of file envelope_constraint.C.

```
00236 {
00237
00238   //  DIE("");
```

```
return JVzero3;


```

4.28.2.20 double EnvelopeConstraint::getDensity (const jVector & J) const

return the density at **jVector** (p. 325) J.

Definition at line 244 of file envelope_constraint.C.

Referenced by BinEnvelopeConstraint::computeStress(), and BinEnvelopeConstraint::dStress().

```
                                                           {


  double D = (*E[0])(J) ;
  //  cerr << D << " = E[0] density \n" ;
  if (D >  EnvelopeConstraint::density_threshold)
    return MAX_ENV_DEN;

  int i = 1;
  while ((i < EnvelopeConstraint::num_evelopes) && (D < EnvelopeConstraint::density_threshold))
    {
      D = (*E[i])(J) ;
      //      cerr << "next, " << D << " = E[" <<i<<"] density \n" ;
      i++;

    }

// else {
//   //  double rv = (1/(1+K))*(((*E[0])(J))+(K*((*env_B)(J))));
//     rv = util_max( a_density , K*(*env_B)(J) );
//     }
//   // DEBUG
// //   cerr << endl << "EC::getDensity: pos{" << J[0] << ", " << J[1]
// //        << ", " << J[2] << "}

```

4.28.2.21 jVector EnvelopeConstraint::getGradient (const jVector & J) const

return the gradient at **jVector** (p. 325) J.

Definition at line 274 of file envelope_constraint.C.

Referenced by MoveModelbyGrad(), and BinEnvelopeConstraint::dStress().

```
                                                              {

  double D = (*E[0])(J);
  jVector G = E[0]->gradient(J) ;
  //  cerr << D << " = E[0] density " ;
  if (D >  EnvelopeConstraint::density_threshold)
    return G;

  int i = 1;
  while ((i < EnvelopeConstraint::num_evelopes) && (D < EnvelopeConstraint::density_threshold))
    {
      D = (*E[i])(J) ;
      G = E[i]->gradient(J);
      //  cerr << D << " = E[" <<i<<"] density " ;
      i++;
```

```
    }

   //
//   jVector G;
//   double p = (*E[0])(J);
//   if ( p >= EnvelopeConstraint::density_threshold ) {
//     if ( p >= K*(*env_B)(J) ) {
//       //    cerr << p <<"(A),  ";
//       G = E[0]->gradient(J);
//     }  else {
//       G = env_B->gradient(J);
//     }
//   } else {
//     G = JVzero;
//   }

```

4.28.3 Friends And Related Function Documentation

4.28.3.1 class BinEnvelopeConstraint [friend]

Definition at line 32 of file envelope_constraint.h.

Referenced by PostSetup().

4.28.4 Member Data Documentation

4.28.4.1 BinEnvelopeConstraint ** EnvelopeConstraint::BEC_List

array of pointers to BEC's; for access.

Definition at line 75 of file envelope_constraint.h.

4.28.4.2 Envelope ** EnvelopeConstraint::E

Array of envelope pointers. [0] points to read-in data, rest are blurred copies.

Definition at line 69 of file envelope_constraint.h.

4.28.4.3 DynString EnvelopeConstraint::Identifier = "envelope:" [static]

Reimplemented from **Constraint** (p. 95).

Definition at line 105 of file globals.C.

4.28.4.4 double EnvelopeConstraint::K

factor used to combine E[0], E[1], etc. when they overlap.

Definition at line 72 of file envelope_constraint.h.

4.28.4.5 int EnvelopeConstraint::blur_kernelSize = 2 [static]

size of blur kernel.

Definition at line 110 of file globals.C.

4.28.4.6 double EnvelopeConstraint::density_threshold = 0.125 [static]

greater than this => atom fits envelope.

Definition at line 107 of file globals.C.

4.28.4.7 int EnvelopeConstraint::num_evelopes = 2 [static]

Number of envelopes to use. 0 is original data, 1..n-1 are blurred versions.

Definition at line 106 of file globals.C.

4.28.4.8 double EnvelopeConstraint::rotation_threshold = 0.08 [static]

If the sum(abs(Rotation Matrix)) > this, then a rotation occurs.

Definition at line 108 of file globals.C.

4.28.4.9 double EnvelopeConstraint::translation_threshold = 0.5 [static]

If the translate vector nrm() > this, then a translation occurs.

Definition at line 109 of file globals.C.

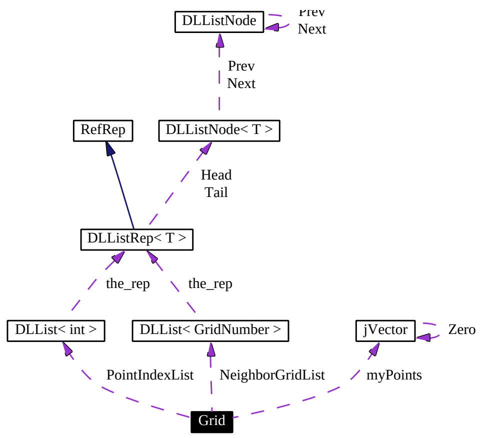
The documentation for this class was generated from the following files:

- **envelope_constraint.h**
- **envelope_constraint.C**
- **globals.C**

4.29 Grid Class Reference

`#include <grid.h>`

Collaboration diagram for Grid:



Public Methods

- **Grid** ()
- **Grid** (**pointList** plist, int numP, double **delta**, double **Length**=0)
- **Grid** (int in_numPoints, double in_delta, double in_Length)
- **Grid** (const Grid &G)
- ~**Grid** ()
- double **getBounds** (void) const
- double **getDelta** (void) const
- int **getNumPoints** (void) const
- const **pointList getPoints** (void) const
- const **GridNumber*** **getGrids** (void) const
- **GridNumber getGrid** (int point_index) const
- **GridNumber getGridDimension** () const
- double **getDistance** (int point1_index, int point2_index) const
- double **getDistance** (int point1_index, const **jVector** &p) const
- double **getDistance** (const **jVector** &p, int point2_index) const
- const **jVector**& **getPosition** (int point_index) const
- bool **JVoverlap** (const **jVector** &point) const
- bool **GNoverlap** (**GridNumber** GN) const
- **DLList**<double> **neighbor_d** (int index, double distance) const
- **DLList**<double> **neighbor_d** (const **jVector** &p, double distance) const
- **DLList<jVector> neighbor_p** (int index, double distance) const
- **DLList<jVector> neighbor_p** (const **jVector** &p, double distance) const
- **DLList<jVector> approx_neighbor_p** (int index, double distance) const
- **DLList<jVector> approx_neighbor_p** (const **jVector** &p, double distance) const
- **DLList**<int> **neighbor_i** (int index, double distance) const
- **DLList**<int> **neighbor_i** (const **jVector** &p, double distance) const
- **DLList**<int> **approx_neighbor_i** (int index, double distance) const
- **DLList**<int> **approx_neighbor_i** (const **jVector** &p, double distance) const

- double **min_neighbor_d** (const **jVector** &p, double distance) const
- void **ReGrid** (**pointList** plist, int numP, double in_delta, double in_Length=0)
- void **ReSize** (int numP, double in_delta, double in_Length)
- void **setPosition** (int index_num, const **jVector** &NewPosition)
- int **addPoints** (**pointList** pList, int numP)
- int **addPoint** (const **jVector** &Position)
- int **deletePoint** (int index_num)
- void **clear** (void)

Private Methods

- void **Initialize** (**pointList** pList, int numP)
- void **setupNeighborLists** ()
- double **calcDistance3** (const **jVector** &j1, const **jVector** &j2) const
- **GridNumber createHash** (const **jVector** &P) const

Private Attributes

- double **Length**
- double **TwiceLength**
- double **N1Dist**
- double **N2Dist**
- double **delta**
- int **numPoints**
- **pointList myPoints**
- **hashmap_type GridHashMap**
- **DLList<GridNumber> NeighborGridList**
- **DLList**<int> **PointIndexList**
- **GridNumber* PointToGridNumArray**
- **GridNumber GridDimension**
- **GridNumber GridDimensionSq**
- hashmap_type::iterator** **PointToHashIter**
- **GridNumber Neighbor1** [27]
- **GridNumber Neighbor2** [125]

4.29.1 Constructor & Destructor Documentation

4.29.1.1 Grid::Grid ()

Definition at line 9 of file grid.C.

```
00009         {
00010     Length = 0;
00011     delta = 0.0;
00012     numPoints = 0;
00013
```

4.29.1.2 Grid::Grid (pointList *plist*, int *in_numPoints*, double *in_delta*, double *in_Length* = 0)

Definition at line 16 of file grid.C.

```
00017 {
00018 
00019   numPoints = in_numPoints;
00020   delta = in_delta;
00021   Length = in_Length;
00022   Initialize(plist,numPoints);
00023 
```

4.29.1.3 Grid::Grid (int *in_numPoints*, double *in_delta*, double *in_Length*)

Definition at line 34 of file grid.C.

```
00035 {
00036   jVector * tempList = new jVector [in_numPoints];
00037 
00038   // create dummy starting posistions
00039 
00040   for (int i=0; i<in_numPoints;i++){
00041     tempList[i].Resize(3);
00042     tempList[i].zero();
00043     //    tempPos = ((((.5)*i)/in_numPoints)+(0.25))*in_Length;
00044   }
00045 
```

4.29.1.4 Grid::Grid (const Grid & *G*)

Definition at line 25 of file grid.C.

```
00025                     {
00026 
00027   numPoints = G.getNumPoints();
00028   delta = G.getDelta();
00029   Length = G.getBounds();
00030   Initialize(G.getPoints(),numPoints);
00031 
00032 
```

4.29.1.5 Grid::~Grid ()

Definition at line 56 of file grid.C.

```
00057 {
00058   clear();
00059 
```

4.29.2 Member Function Documentation

4.29.2.1 bool Grid::GNoverlap (GridNumber *GN*) const

Definition at line 189 of file grid.C.

Referenced by JVoverlap().

```
                                       {

pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
iter_pair = GridHashMap.equal_range(GN);
if (iter_pair.first != iter_pair.second){
  // an atom exits in this grid
  return true;
}

```

4.29.2.2 void Grid::Initialize (pointList *plist*, int *numP*) [private]

Definition at line 61 of file grid.C.

Referenced by Grid(), and ReGrid().

```
00061                                             {
00062
00063   // Reset bounding box width (length’) if too small
00064   double temp;
00065   double tempLength  = 0;
00066   for(int i=0; i<numP; i++) {
00067     for(int j=0; j<3; j++) {
00068       //      if (plist[i]){ //sanity check-- make sure a jVecotr is there LEAKING
00069         temp = fabs(plist[i][j]);
00070         if (tempLength < temp ) tempLength = temp;
00071         //      }
00072     }
00073   }
00074   if (tempLength >= Length) {
00075     // minimum bounds condition
00076     Length = tempLength;
00077   }
00078
00079   numPoints = numP; // different if resze or regrid is called.
00080   TwiceLength = 2 * Length;  //Commonly reused multiples, calculated once.
00081   N1Dist = delta;
00082   N2Dist = 2 * delta;
00083
00084   GridDimension = (GridNumber) ((TwiceLength+delta)/delta);
00085
00086   if (GridDimension > 1290) { // fix to check actual limit == (max long int) ^-3
00087     cerr << "You have attempted to create a Grid with more" << endl
00088          << "than 1290 elements in each dimension.  This is" << endl
00089          << "not allowed because of the limitations on the" << endl
00090          << "of long integers used on this system." << endl;
00091     cerr << "The number of elements you tried to use was "
00092          << GridDimension << " ." << endl << endl;
00093     exit(1);
00094   }
00095
00096   GridDimensionSq =  GridDimension*GridDimension;
00097
00098   setupNeighborLists();
00099
00100   GridHashMap.clear();
00101   GridHashMap.resize(5*numP);  // MAGIC NUMBER (use this many hash buckets)
00102
00103   PointToGridNumArray = new GridNumber[numP];
00104   myPoints = new jVector[numP];
00105   PointToHashIter = new hashmap_type::iterator * [numP];
00106   GridNumber TempGridNumber;
00107
```

```
00108   // Go through all the input points
00109   for(int i=0; i<numP; i++) {
00110     // Save a copy of the location vector
00111     myPoints[i] = plist[i];
00112
00113     // Create a long int to hash the filled grids
00114     TempGridNumber = createHash(myPoints[i]);
00115
00116     // Add elements to the GridHashMap
00117     PointToHashIter[i] = new hashmap_type::iterator;
00118     *PointToHashIter[i] = GridHashMap.insert(hashmap_type::value_type(TempGridNumber, i));
00119
00120     // Save the hash number for the grid
00121     PointToGridNumArray[i] = TempGridNumber;
00122   }
00123
```

4.29.2.3 bool Grid::JVoverlap (const jVector & *point*) const

Definition at line 201 of file grid.C.

```
00201                                                {
00202
00203   GridNumber boxNumber = createHash(point);
00204   return GNoverlap(boxNumber);
00205
00206
```

4.29.2.4 void Grid::ReGrid (pointList *plist*, int *numP*, double *in_delta*, double *in_Length* = 0)

Definition at line 158 of file grid.C.

Referenced by ReSize().

```
00158                                                                                {
00159   // RE-allocate new Grid box containing all points in the list
00160   //  This should be  called if a point is moved outside the current
00161   //  bounds
00162
00163   clear();
00164
00165   delta = in_delta;
00166   Length = in_Length;
00167   Initialize(plist, numP);
00168
00169
00170
```

4.29.2.5 void Grid::ReSize (int *numP*, double *in_delta*, double *in_Length*)

Definition at line 172 of file grid.C.

```
00172                                                             {
00173
00174   jVector * tempList = new jVector [numP];
00175
00176   // create dummy starting posistions
```

```

for (int i=0; i<numP;i++){
  tempList[i].Resize(3);
  tempList[i].zero();
}

```

4.29.2.6 int Grid::addPoint (const jVector & *Position*) [inline]

Definition at line 150 of file grid.h.

```
                                        {/*not implemented yet*/return 0;}

```

4.29.2.7 int Grid::addPoints (pointList *pList*, int *numP*) [inline]

Definition at line 142 of file grid.h.

```
                                           {/*Not yet*/ return 0;}

```

4.29.2.8 DLList< int > Grid::approx_neighbor_i (const jVector & *p*, double *distance*) const

Definition at line 683 of file grid.C.

```
                                                                          {
// returns a doublely-linked list of point **indicies** within all the
// grids that are 'distance' away from point p.
// All points in qualifying grids are returned.  This means it is a
// correct list, yet may contain extra elements.
  DLList<int> ReturnList;
  if (distance <= N1Dist) {
    // Go through 27 one-neighbor grids and add up contributions
    pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
    hashmap_type::const_iterator out_iter;
    GridNumber TempGridNumber = createHash(p);
    for (int i = 0; i < 27; i++){
      iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
      if (iter_pair.first != iter_pair.second){
        // an atom exits in this grid
        for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
          ReturnList.AddToFront((*out_iter).second); // index
        }
      }
    } // end of neighbor1 cycles
  } else if (distance <= N2Dist){
    // Go through 27 one-neighbor grids and add up contributions
    pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
    hashmap_type::const_iterator out_iter;
    GridNumber TempGridNumber = createHash(p);
    for (int i = 0; i < 125; i++){
      iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
      if (iter_pair.first != iter_pair.second){
        // an atom exits in this grid
        for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
          ReturnList.AddToFront((*out_iter).second); // index
```

```
          }
        }
      } // end of neighbor2 cycles
    } else { // outside near neighbors, return all
      for (int i = 0; i < numPoints; i++){
        ReturnList.AddToFront(i);                       // index
      }
    }

```

4.29.2.9 DLList< int > Grid::approx_neighbor_i (int *index*, double *distance*) const

Definition at line 633 of file grid.C.

Referenced by Molecule::approx_neighbor_i().

```
00633                                                                     {
00634   jVector p = myPoints[index];
00635 // returns a doublely-linked list of point **indicies** within all the
00636 // grids that are 'distance' away from point p.
00637 // All points in qualifying grids are returned.  This means it is a
00638 // correct list, yet may contain extra elements.
00639   DLList<int> ReturnList;
00640   if (distance <= N1Dist) {
00641     // Go through 27 one-neighbor grids and add up contributions
00642     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00643     hashmap_type::const_iterator out_iter;
00644     GridNumber TempGridNumber = createHash(p);
00645     for (int i = 0; i < 27; i++){
00646       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00647       if (iter_pair.first != iter_pair.second){
00648         // an atom exits in this grid
00649         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00650           if (out_iter != *(PointToHashIter[index])){
00651             ReturnList.AddToFront((*out_iter).second); // index
00652           }
00653         }
00654       }
00655     } // end of neighbor1 cycles
00656   } else if (distance <= N2Dist){
00657     // Go through 27 one-neighbor grids and add up contributions
00658     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00659     hashmap_type::const_iterator out_iter;
00660     GridNumber TempGridNumber = createHash(p);
00661     for (int i = 0; i < 125; i++){
00662       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
00663       if (iter_pair.first != iter_pair.second){
00664         // an atom exits in this grid
00665         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00666           if (out_iter != *(PointToHashIter[index])){
00667             ReturnList.AddToFront((*out_iter).second); // index
00668           }
00669         }
00670       }
00671     } // end of neighbor2 cycles
00672   } else { // outside near neighbors, return all
00673     for (int i = 0; i < numPoints; i++){
00674       if (i != index ) {
00675         ReturnList.AddToFront(i);                       // index
00676       }
00677     }
00678   }
00679
```

4.29.2.10 DLList< jVector > Grid::approx_neighbor_p (const jVector & *p*, double *distance*) const

Definition at line 491 of file grid.C.

```
00491                                                                             {
00492 // Returnsa an approximate list of **positions** to other points within
00493 // a given distance.  This is approximate because the distances are
00494 // not calculated here.  All points in qualifying grids are
00495 // returned.  This means it is a correct list, yet may contain extra
00496 // elements.
00497
00498   DLList<jVector> ReturnList;
00499   if (distance <= N1Dist) {
00500     // Go through 27 one-neighbor grids and add up contributions
00501     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00502     hashmap_type::const_iterator out_iter;
00503     GridNumber TempGridNumber = createHash(p);
00504     for (int i = 0; i < 27; i++){
00505       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00506       if (iter_pair.first != iter_pair.second){
00507         // an atom exits in this grid
00508         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00509           ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
00510         }
00511       }
00512     } // end of neighbor1 cycles
00513   } else if (distance <= N2Dist){
00514     // Go through 27 one-neighbor grids and add up contributions
00515     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00516     hashmap_type::const_iterator out_iter;
00517     GridNumber TempGridNumber = createHash(p);
00518     for (int i = 0; i < 125; i++){
00519       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
00520       if (iter_pair.first != iter_pair.second){
00521         // an atom exits in this grid
00522         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00523           ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
00524         }
00525       }
00526     } // end of neighbor2 cycles
00527   } else { // outside near neighbors, return all
00528     for (int i = 0; i < numPoints; i++){
00529       ReturnList.AddToFront(myPoints[i]); // jVector position
00530     }
00531   }
00532
```

4.29.2.11 DLList< jVector > Grid::approx_neighbor_p (int *index*, double *distance*) const

Definition at line 438 of file grid.C.

```
00438                                                                      {
00439   jVector p = myPoints[index];
00440 // Returnsa an approximate list of **positions** to other points within
00441 // a given distance.  This is approximate because the distances are
00442 // not calculated here.  All points in qualifying grids are
00443 // returned.  This means it is a correct list, yet may contain extra
00444 // elements.
00445
00446   DLList<jVector> ReturnList;
```

```
if (distance <= N1Dist) {
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 27; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        if (out_iter != *(PointToHashIter[index])){
          ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
        }
      }
    }
  } // end of neighbor1 cycles
} else if (distance <= N2Dist){
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 125; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        if (out_iter != *(PointToHashIter[index])){
          ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
        }
      }
    }
  } // end of neighbor2 cycles
} else { // outside near neighbors, return all
  for (int i = 0; i < numPoints; i++){
    if (i != index){
      ReturnList.AddToFront(myPoints[i]); // jVector position
    }
  }
}
```

4.29.2.12 double Grid::calcDistance3 (const jVector & *j1*, const jVector & *j2*) const [inline, private]

Definition at line 247 of file grid.h.

Referenced by getDistance().

```
00247                                                                              {
00248
```

4.29.2.13 void Grid::clear (void)

Definition at line 728 of file grid.C.

Referenced by ReGrid(), and ~Grid().

```
00729 {
00730   // Empties the Grid, deletes and zeros all dynamic memory.
00731
```

```
GridHashMap.clear();

for (int i = 0; i< numPoints; i++){
  delete PointToHashIter[i];
}

```

4.29.2.14 GridNumber Grid::createHash (const jVector & *P*) const [inline, private]

Definition at line 223 of file grid.h.

Referenced by Initialize(), JVoverlap(), approx_neighbor_i(), approx_neighbor_p(), neighbor_d(), neighbor_i(), neighbor_p(), and setPosition().

```
                                                        {
  GridNumber Hashx, Hashy, Hashz;
  Hashx = (GridNumber) (((P[0] + Length)/TwiceLength)*(GridDimension-1));
  Hashy = (GridNumber) (((P[1] + Length)/TwiceLength)*(GridDimension-1));
  Hashz = (GridNumber) (((P[2] + Length)/TwiceLength)*(GridDimension-1));
  return Hashx + GridDimension * Hashy + GridDimensionSq * Hashz;

```

4.29.2.15 int Grid::deletePoint (int *index_num*) [inline]

Definition at line 156 of file grid.h.

```
                                   {/*not implemented yet*/return 0;}

```

4.29.2.16 double Grid::getBounds (void) const [inline]

Definition at line 261 of file grid.h.

Referenced by Grid().

```
  { return Length; }

```

4.29.2.17 double Grid::getDelta (void) const [inline]

Definition at line 264 of file grid.h.

Referenced by Grid().

```
{ return delta; }

```

4.29.2.18 double Grid::getDistance (const jVector & *p*, int *point2_index*) const [inline]

Definition at line 257 of file grid.h.

```
                                                                          {

```

4.29.2.19 double Grid::getDistance (int *point1_index*, const jVector & *p*) const [inline]

Definition at line 254 of file grid.h.

```
00254                                                                        {
00255
```

4.29.2.20 double Grid::getDistance (int *point1_index*, int *point2_index*) const [inline]

Definition at line 72 of file grid.h.

Referenced by neighbor_d(), neighbor_i(), and neighbor_p().

```
00072                                                              {
00073
```

4.29.2.21 GridNumber Grid::getGrid (int *point_index*) const [inline]

Definition at line 276 of file grid.h.

```
00276                                                     {
00277     return PointToGridNumArray[point_index];
00278
```

4.29.2.22 GridNumber Grid::getGridDimension () const [inline]

Definition at line 281 of file grid.h.

```
00281                                               {
00282     return GridDimension;
00283
```

4.29.2.23 const GridNumber ∗ Grid::getGrids (void) const [inline]

Definition at line 273 of file grid.h.

```
00274   { return PointToGridNumArray; }
00275
```

4.29.2.24 int Grid::getNumPoints (void) const [inline]

Definition at line 267 of file grid.h.

Referenced by Grid().

```
00268   { return numPoints; }
00269
```

4.29.2.25 const pointList Grid::getPoints (void) const [inline]

Definition at line 270 of file grid.h.

Referenced by Grid().

```
00271   { return myPoints; }
00272
```

4.29.2.26 const jVector & Grid::getPosition (int *point_index*) const [inline]

Definition at line 250 of file grid.h.

Referenced by Molecule::Bounds(), Molecule::Copy(), Molecule::getCOM(), Molecule::getPosition(), Molecule::operator()(), Molecule::writeXmol(), and Molecule::writeXmolData().

```
00250                                                               {
00251
```

4.29.2.27 double Grid::min_neighbor_d (const jVector & *p*, double *distance*) const

Definition at line 726 of file grid.C.

```
00726                                                                    {/*not yet*/return 0.0;}
00727
```

4.29.2.28 DLList< double > Grid::neighbor_d (const jVector & *p*, double *distance*) const

Definition at line 298 of file grid.C.

```
00298                                                                      {
00299 // Returns a list of **distances** to other points within a given
00300 // distance to the first point.
00301   DLList<double> ReturnList;
00302   double d;
00303   if (distance <= N1Dist) {
00304     // Go through 27 one-neighbor grids and add up contributions
00305     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00306     hashmap_type::const_iterator out_iter;
00307     GridNumber TempGridNumber = createHash(p);
00308     for (int i = 0; i < 27; i++){
00309       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00310       if (iter_pair.first != iter_pair.second){
00311         // an atom exits in this grid
00312         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00313           d = getDistance((*out_iter).second,p); // distance
00314           if (d < distance) ReturnList.AddToFront(d);
00315         }
00316       }
00317     } // end of neighbor1 cycles
00318   } else if (distance <= N2Dist){
00319     // Go through 27 one-neighbor grids and add up contributions
00320     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00321     hashmap_type::const_iterator out_iter;
00322     GridNumber TempGridNumber = createHash(p);
00323     for (int i = 0; i < 125; i++){
```

```
      iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
      if (iter_pair.first != iter_pair.second){
        // an atom exits in this grid
        for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
          d = getDistance((*out_iter).second,p); // distance
          if (d < distance) ReturnList.AddToFront(d);
        }
      }
    } // end of neighbor2 cycles
  } else { // outside near neighbors, return all
    for (int i = 0; i < numPoints; i++){
      d = getDistance(i,p); // distance
      if (d < distance) ReturnList.AddToFront(d);
    }
  }

```

4.29.2.29 DLList< double > Grid::neighbor_d (int *index*, double *distance*) const

Definition at line 246 of file grid.C.

```
00246                                                               {
00247   jVector p = myPoints[index];
00248 // Returns a list of **distances** to other points within a given
00249 // distance to the first point.
00250   DLList<double> ReturnList;
00251   double d;
00252   if (distance <= N1Dist) {
00253     // Go through 27 one-neighbor grids and add up contributions
00254     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00255     hashmap_type::const_iterator out_iter;
00256     GridNumber TempGridNumber = createHash(p);
00257     for (int i = 0; i < 27; i++){
00258       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00259       if (iter_pair.first != iter_pair.second){
00260         // an atom exits in this grid
00261         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00262           if (out_iter != *(PointToHashIter[index])){
00263             d = getDistance((*out_iter).second,p); // distance
00264             if (d < distance) ReturnList.AddToFront(d);
00265           }
00266         }
00267       }
00268     } // end of neighbor1 cycles
00269   } else if (distance <= N2Dist){
00270     // Go through 27 one-neighbor grids and add up contributions
00271     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00272     hashmap_type::const_iterator out_iter;
00273     GridNumber TempGridNumber = createHash(p);
00274     for (int i = 0; i < 125; i++){
00275       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
00276       if (iter_pair.first != iter_pair.second){
00277         // an atom exits in this grid
00278         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00279           if (out_iter != *(PointToHashIter[index])){
00280             d = getDistance((*out_iter).second,p); // distance
00281             if (d < distance) ReturnList.AddToFront(d);
00282           }
00283         }
00284       }
00285     } // end of neighbor2 cycles
00286   } else { // outside near neighbors, return all
00287     for (int i = 0; i < numPoints; i++){
```

```
    if (i != index){
      d = getDistance(i,p); // distance
      if (d < distance) ReturnList.AddToFront(d);
    }
  }
}

```

4.29.2.30 DLList< int > Grid::neighbor_i (const jVector & *p*, double *distance*) const

Definition at line 588 of file grid.C.

```
00588                                                                   {
00589   // returns a doublely-linked list of point **indicies** within all the
00590   // grids that are 'distance' away from point p. NB this calclulates
00591   // and discards distances. (SLOW)
00592   DLList<int> ReturnList;
00593   double d;
00594   if (distance <= N1Dist) {
00595     // Go through 27 one-neighbor grids and add up contributions
00596     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00597     hashmap_type::const_iterator out_iter;
00598     GridNumber TempGridNumber = createHash(p);
00599     for (int i = 0; i < 27; i++){
00600       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00601       if (iter_pair.first != iter_pair.second){
00602         // an atom exits in this grid
00603         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00604           d = getDistance((*out_iter).second,p); // distance
00605           if (d< distance) ReturnList.AddToFront((*out_iter).second); // index
00606         }
00607       }
00608     } // end of neighbor1 cycles
00609   } else if (distance <= N2Dist){
00610     // Go through 27 one-neighbor grids and add up contributions
00611     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00612     hashmap_type::const_iterator out_iter;
00613     GridNumber TempGridNumber = createHash(p);
00614     for (int i = 0; i < 125; i++){
00615       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
00616       if (iter_pair.first != iter_pair.second){
00617         // an atom exits in this grid
00618         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00619           d = getDistance((*out_iter).second,p); // distance
00620           if (d< distance) ReturnList.AddToFront((*out_iter).second); // index
00621         }
00622       }
00623     } // end of neighbor2 cycles
00624   } else { // outside near neighbors, return all
00625     for (int i = 0; i < numPoints; i++){
00626       d = getDistance(i,p); // distance
00627       if (d < distance) ReturnList.AddToFront(i); // index
00628     }
00629   }
00630
```

4.29.2.31 DLList< int > Grid::neighbor_i (int *index*, double *distance*) const

Definition at line 535 of file grid.C.

Referenced by Molecule::neighbor_i().

```
                                                               {
jVector p = myPoints[index];
// returns a doublely-linked list of point **indicies** within all the
// grids that are 'distance' away from point p. NB this calclulates
// and discards distances. (SLOW)
DLList<int> ReturnList;
double d;
if (distance <= N1Dist) {
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 27; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        if (out_iter != *(PointToHashIter[index])){
          d = getDistance((*out_iter).second,p); // distance
          if (d< distance) ReturnList.AddToFront((*out_iter).second); // index
        }
      }
    }
  } // end of neighbor1 cycles
} else if (distance <= N2Dist){
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 125; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        if (out_iter != *(PointToHashIter[index])){
          d = getDistance((*out_iter).second,p); // distance
          if (d< distance) ReturnList.AddToFront((*out_iter).second); // index
        }
      }
    }
  } // end of neighbor2 cycles
} else { // outside near neighbors, return all
  for (int i = 0; i < numPoints; i++){
    if (i != index){
      d = getDistance(i,p); // distance
      if (d < distance) ReturnList.AddToFront(i); // index
    }
  }
}
```

4.29.2.32 DLList< jVector > Grid::neighbor_p (const jVector & *p*, double *distance*) const

Definition at line 394 of file grid.C.

```
00394                                                                     {
00395   // Returns a list of **positions** to other points within a given
00396   // distance NB this calclulates and discards distances. (SLOW)
00397   DLList<jVector> ReturnList;
```

```
double d;
if (distance <= N1Dist) {
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 27; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        d = getDistance((*out_iter).second,p); // distance
        if (d< distance) ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
      }
    }
  } // end of neighbor1 cycles
} else if (distance <= N2Dist){
  // Go through 27 one-neighbor grids and add up contributions
  pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
  hashmap_type::const_iterator out_iter;
  GridNumber TempGridNumber = createHash(p);
  for (int i = 0; i < 125; i++){
    iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
    if (iter_pair.first != iter_pair.second){
      // an atom exits in this grid
      for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
        d = getDistance((*out_iter).second,p); // distance
        if (d< distance) ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
      }
    }
  } // end of neighbor2 cycles
} else { // outside near neighbors, return all
  for (int i = 0; i < numPoints; i++){
    d = getDistance(i,p); // distance
    if (d < distance) ReturnList.AddToFront(myPoints[i]); // jVector position
  }
}
```

4.29.2.33 DLList< jVector > Grid::neighbor_p (int *index*, double *distance*) const

Definition at line 342 of file grid.C.

```
00342                                                              {
00343   jVector p = myPoints[index];
00344     // Returns a list of **positions** to other points within a given
00345   // distance NB this calclulates and discards distances. (SLOW)
00346   DLList<jVector> ReturnList;
00347   double d;
00348   if (distance <= N1Dist) {
00349     // Go through 27 one-neighbor grids and add up contributions
00350     pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
00351     hashmap_type::const_iterator out_iter;
00352     GridNumber TempGridNumber = createHash(p);
00353     for (int i = 0; i < 27; i++){
00354       iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor1[i]);
00355       if (iter_pair.first != iter_pair.second){
00356         // an atom exits in this grid
00357         for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
00358           if (out_iter != *(PointToHashIter[index])){
00359             d = getDistance((*out_iter).second,p); // distance
00360             if (d< distance) ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
00361           }
```

```
        }
      }
    } // end of neighbor1 cycles
  } else if (distance <= N2Dist){
    // Go through 27 one-neighbor grids and add up contributions
    pair<hashmap_type::const_iterator, hashmap_type::const_iterator> iter_pair;
    hashmap_type::const_iterator out_iter;
    GridNumber TempGridNumber = createHash(p);
    for (int i = 0; i < 125; i++){
      iter_pair = GridHashMap.equal_range(TempGridNumber-Neighbor2[i]);
      if (iter_pair.first != iter_pair.second){
        // an atom exits in this grid
        for (out_iter = iter_pair.first; out_iter != iter_pair.second; ++out_iter){
          if (out_iter != *(PointToHashIter[index])){
            d = getDistance((*out_iter).second,p); // distance
            if (d< distance) ReturnList.AddToFront(myPoints[(*out_iter).second]); // jVector position
          }
        }
      }
    } // end of neighbor2 cycles
  } else { // outside near neighbors, return all
    for (int i = 0; i < numPoints; i++){
      if (i != index){
        d = getDistance(i,p); // distance
        if (d < distance) ReturnList.AddToFront(myPoints[i]); // jVector position
      }
    }
  }

```

4.29.2.34 void Grid::setPosition (int *index_num*, const jVector & *NewPosition*)

Definition at line 208 of file grid.C.

Referenced by Molecule::Copy(), Molecule::DynRead(), Molecule::Input(), Molecule::InputBinary(), Molecule::Molecule(), Molecule::readXmol(), Molecule::setPosition(), and Molecule::setRandom().

```
00208                                                                 {
00209   // Moves the point with index == index_num to the new position
00210   // spoecified by NewPosition  This has to maintain the
00211 
00212   myPoints[index_num] = NewPosition;
00213 
00214 
00215   // DEBUG
00216   // bounds check
00217   if ( (fabs(NewPosition[0]) > Length) ||
00218        (fabs(NewPosition[1]) > Length) ||
00219        (fabs(NewPosition[2]) > Length) ){
00220     cerr << "Grid::setPostion moved out of bounds: MaxGridLength = "
00221          << Length
00222          << " New Position = "
00223          << NewPosition
00224          << endl;
00225     raise (SIGQUIT);
00226   }
00227 
00228   // Should check bounds here.  Not done now.  May cause errors if
00229   // max(fabs(NewPosition[0,1,2])) > Length.
00230   // need to call regrid if bounds are not OK.
00231 
00232   GridNumber TempGridNumber = createHash(NewPosition);
```

```
00233   if (TempGridNumber !=  PointToGridNumArray[index_num]){
00234     // only unpdate hash if point moves to new grid.
00235     // delete old point in hash
00236     GridHashMap.erase(*(PointToHashIter[index_num]));
00237 
00238     // add point in new grid to hash
00239     *PointToHashIter[index_num] = GridHashMap.insert(hashmap_type::value_type(TempGridNumber, index_num));
00240 
00241     PointToGridNumArray[index_num] = TempGridNumber;
00242   }
00243 
```

4.29.2.35 void Grid::setupNeighborLists (void) [private]

Definition at line 125 of file grid.C.

Referenced by Initialize().

```
00126 {
00127   register long x,y,z;
00128   int index = 0;
00129   for (z = -1; z < 2; z++){
00130     for (y = -1; y < 2; y++){
00131       for (x = -1; x < 2; x++){
00132         Neighbor1[index++] = z*GridDimensionSq + y*GridDimension + x;
00133       }
00134     }
00135   }
00136 
00137   index = 0;
00138   for (z = -2; z < 3; z++){
00139     for (y = -2; y < 3; y++){
00140       for (x = -2; x < 3; x++){
00141         Neighbor2[index++] = z*GridDimensionSq + y*GridDimension + x;
00142       }
00143     }
00144   }
00145 
```

4.29.3 Member Data Documentation

4.29.3.1 GridNumber Grid::GridDimension [private]

Definition at line 191 of file grid.h.

4.29.3.2 GridNumber Grid::GridDimensionSq [private]

Definition at line 192 of file grid.h.

4.29.3.3 hashmap_type Grid::GridHashMap [private]

Definition at line 180 of file grid.h.

4.29.3.4 double Grid::Length [private]

Definition at line 170 of file grid.h.

4.29.3.5 double Grid::N1Dist [private]

Definition at line 172 of file grid.h.

4.29.3.6 double Grid::N2Dist [private]

Definition at line 173 of file grid.h.

4.29.3.7 GridNumber Grid::Neighbor1[27] [private]

Definition at line 213 of file grid.h.

4.29.3.8 GridNumber Grid::Neighbor2[125] [private]

Definition at line 214 of file grid.h.

4.29.3.9 DLList< GridNumber > Grid::NeighborGridList [private]

Definition at line 185 of file grid.h.

4.29.3.10 DLList< int > Grid::PointIndexList [private]

Definition at line 186 of file grid.h.

4.29.3.11 GridNumber * Grid::PointToGridNumArray [private]

Definition at line 188 of file grid.h.

4.29.3.12 hashmap_type::iterator ** Grid::PointToHashIter [private]

Definition at line 196 of file grid.h.

4.29.3.13 double Grid::TwiceLength [private]

Definition at line 171 of file grid.h.

4.29.3.14 double Grid::delta [private]

Definition at line 174 of file grid.h.

4.29.3.15 pointList Grid::myPoints [private]

Definition at line 177 of file grid.h.

4.29.3.16 int Grid::numPoints `[private]`

Definition at line 175 of file grid.h.

The documentation for this class was generated from the following files:

- **grid.h**
- **grid.C**

4.30 gwNumerics Class Reference

`#include <gw_numerics.h>`

Public Methods

- **gwNumerics** ()
- ~**gwNumerics** ()

Static Public Methods

- void **SolveVector** (const **jVector** &gradient, const **jVector** &\hessian, **jVector** &result)
- double **CalculateDerivative** (const **jVector** &direction, const\ **jVector** &gradient)
- bool **CubicLineSearch** (const **jVector** &position, const **jVector** &\direction, const double &derivative, const\ **ConstraintManager** &CM, int atom_index,\ **jVector** &NewPosition, double &qmin)
- **jVector SetHCPosition2** (const **jVector** &start_position, const **jVector** &curr_position,\const **jVector** &direction, **ConstraintManager** &\CM, int atom_index, double qmin_dist, double &max_lpert_r)
- void **addVDWSegments** (**Segments** &Seglist, const **jVector** &curr_position,\const **jVector** &direction, **ConstraintManager** &\CM, int atom_index, double offset=0)
- void **addLPERTSegments** (**Segments** &Seglist, const **jVector** &curr_position,\const **jVector** &direction, **ConstraintManager** &\CM, int atom_index, double qmin_dist,\const **jVector** &start_position, const **jVector** &new_position, double &max_r)
- double **LinearThreshold** (const double &stress, double *RadArray)
- **Segment LineIntersectSphere** (const **jVector** &atom_position, const double &\radius, const **jVector** &vec_position, const\ **jVector** &atom_direction)
- void **bfgs_step** (const **jVector** &x, const **jVector** &xp, const **jVector** &gf, const **jVector** &gfp, **jVector** &H, **jVector** &direction)
- void **LocalPert** (int I, **ConstraintManager** &CM, double min_dist, **jVector** start_position)
- int **SpreadR** (int I, double radius, **DLList**< int > Moved, int depth, **ConstraintManager** &CM, **jVector** start_position, const **jVector** pref_dir)
- **DLList**<int> **CheckViol** (int I, **DLList**< int > NeighborList, bool UsePrevious, const **ConstraintManager** &CM)
- int **MoveOnSingleViol** (int I, double radius, int otherAtom, **ConstraintManager** &CM)
- **jVector CalcForce** (int I, double radius, **DLList**< int > ViolList, **DLList**< int > NeighborList, **DLList**< int > Moved, const **ConstraintManager** &CM, const **jVector** &start_position, double min_dist, const **jVector** pref_dir)
- void **reset_LP_Count** (void)

Static Public Attributes

- int **NumLP** = 0
- int **NumNotLP** = 0
- bool **UseOnlyMoved** = true

  Flag to set vdw exclusion based on previously moved atoms only.

- double **ls_dist_tolerance** = 0.0001
- double **ls_succ_tolerance** = 0.001

- double **ls_deriv_tolerance** = 0.0000001
- bool **LocalPerturb** = true

  Setting for local perturbation activation.

- int **MaxNCuts** = 10

  Maximum number of cuts used in the cubic line search.

- double **LPertFactor**

  Multiplication factor for local pert radius.

- double **MaxAbsDistRes** = -1 ∗ INF

  the max residual over distance constraints.

- double **AveAbsDistRes** = 0

  the average residual over distance constraints.

- double **MinDistVariance** = INF

  the minimum variance fron distance constraint data.

4.30.1 Detailed Description

//

File: **gw_numerics.h**

Created by: Jon Dugan

This class is never instantiated. It containts static, public functions available for use by other classes (in this case GRAPE Jobs). Several of the numerical techniques originally came from Glenn Williams, hence the 'gw_' in the name.

//

Definition at line 33 of file gw_numerics.h.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 gwNumerics::gwNumerics ()

Definition at line 3 of file gw_numerics.C.

```
00004 {
00005
```

4.30.2.2 gwNumerics::∼gwNumerics ()

Definition at line 7 of file gw_numerics.C.

```
00008 {
00009
```

4.30.3 Member Function Documentation

4.30.3.1 jVector gwNumerics::CalcForce (int *I*, double *radius*, DLList< int > *ViolList*, DLList< int > *NeighborList*, DLList< int > *Moved*, const ConstraintManager & *CM*, const jVector & *start_position*, double *min_dist*, const jVector *pref_dir*) `[static]`

4.30.3.2 double gwNumerics::CalculateDerivative (const jVector & *direction*, const\jVector & *gradient*) `[static]`

Returns the derivative of the evaluation function at a position. GGG This return variable is used to check for convergence in the iterations GGG

4.30.3.3 DLList<int> gwNumerics::CheckViol (int *I*, DLList< int > *NeighborList*, bool *UsePrevious*, const ConstraintManager & *CM*) `[static]`

4.30.3.4 bool gwNumerics::CubicLineSearch (const jVector & *position*, const jVector &\ *direction*, const double & *derivative*, const\ConstraintManager & *CM*, int *atom_index*, \jVector & *NewPosition*, double & *qmin*) `[static]`

Returns bool flag indication success(true) or failure (false). The function also resets the value in 'NewPosition' to the best position on the line if the search was successful.

Referenced by gwRelaxAtom::run().

4.30.3.5 Segment gwNumerics::LineIntersectSphere (const jVector & *atom_position*, const double &\ *radius*, const jVector & *vec_position*, const\jVector & *atom_direction*) `[static]`

using the vec_position and vec_direction, a line in three space is tested if an atom at atom_-position with radius specified by radius intersects that line. **Segment** (p. 480) returned has units of direction.mag()

Referenced by addLPERTSegments().

4.30.3.6 double gwNumerics::LinearThreshold (const double & *X*, double * *Array*) `[static]`

Determines the 'radius' of an atom for hard constraint purposes using the linear interpolation method between the max and min settings in RadArray.

Definition at line 466 of file gw_numerics.C.

```
{
  // The Array specifies the linear function that maps one space
  // into another, like from X to Y.  Array[0] = Xmin,
  // Array[1] = Yval for X < Xmin.  Array[2]= Xmax-Xmin, and Array[3]=
  // Yval for X > Xmax.  Linear interpolation is done if X is between
  // Xmin and Xmax.

  if (X <= Array[0]) return Array[1];
  if (X >= Array[0]+Array[2]) return Array[3]; // Array[2]==0 => step fntn
  if (Array[2]!=0){ //just in case
    return Array[1]+(Array[3]-Array[1])*((X-Array[0])/Array[2]);
  }

```

4.30.3.7 void gwNumerics::LocalPert (int *I*, ConstraintManager & *CM*, double *min_dist*, jVector *start_position*) [static]

4.30.3.8 int gwNumerics::MoveOnSingleViol (int *I*, double *radius*, int *otherAtom*, ConstraintManager & *CM*) [static]

4.30.3.9 jVector gwNumerics::SetHCPosition2 (const jVector & *start_position*, const jVector & *curr_position*, \const jVector & *direction*, ConstraintManager &\ *CM*, int *atom_index*, double *qmin_dist*, double & *max_lpert_r*) [static]

This returns a [3x1] vector used to move an atom in its final iteration, conforming to the Van Der Waal constraints.

Definition at line 88 of file gw_numerics.C.

Referenced by gwRelaxAtom::run().

```
00090                                                              {
00091
00092
00094   //    overview
00095   //
00096   // add vdw to seglist
00097   // add lpert to seglist
00098   // copy seglist
00099   // check viol
00100   // if viol -- loop:
00101   //   // get neighbors at best
00102   //   // add vdw
00103   //   // check viol
00104   // until !viol
00105   //
00107
00108   Segments SegList;
00109   Segment TempSegment;
00110   double func_eval1, func_eval2;
00111
00112   jVector new_position = curr_position +direction*qmin_dist;
00113
00114 //   cerr << "*1 new_position =>" << new_position << endl;
00115 //   cerr << "*1 curr_position =>" << curr_position << endl;
00116
00117 //   cerr << "new_position" << new_position << endl;
00118 //   cerr << "curr_position" << curr_position << endl;
00119 //   cerr << "qmin_dist"  << qmin_dist << endl;
00120
00121 #ifdef GNOMAD_OUTPUT
00122       printf("  Initial qmin:\n");
00123       printf("    initial qmin = %lf\n", qmin_dist);
00124 #endif
00125
00126
00127   // add vdw to seglist
00128       addVDWSegments(SegList, curr_position,  \
00129                      direction, CM, atom_index, 0);
00130
00131        //  cerr << "*2 new_position =>" << new_position << endl;
00132
00133   // add lpert to seglist
00134   if (LocalPerturb)
00135     addLPERTSegments (SegList, curr_position,  \
00136                      direction, CM, atom_index, qmin_dist, \
```

```
                      start_position, new_position, max_lpert_r);

  //  cerr << "*3 new_position =>" << new_position << endl;

  // check viol  -- write seglist.qsortviol
  TempSegment = SegList.QSortReduceViol(qmin_dist);

  if ((TempSegment.start == 0) && (TempSegment.end == 0)){
    // no violation

#ifdef GNOMAD_OUTPUT
    printf("  After vdw_segs\n");
    //    printf("    lbound, ubound = %lf %lf\n", *lbound, *ubound);
    //    printf("    num_segments = %d\n", *num_segments);
    printf ("Segment List:\n");
    for(int ixx=0; ixx < SegList.GetNum(); ixx++)
      printf("      start_ineq, end_ineq = %lf %lf\n", SegList.GetSeg(ixx).start, SegList.GetSeg(ixx).end );
    printf("\n");

    printf("  After viol_check\n");
    printf("    qmin = %lf\n\n", qmin_dist);
#endif

    //    cerr << "*4 new_position =>" << new_position << endl;
    return new_position;

  } else {     // BEGIN Violation else


    while ((TempSegment.start != 0) && (TempSegment.end != 0)){
      // BEGIN location loop:

      //find current best
      func_eval1 = CM.stress(atom_index, curr_position+direction*TempSegment.start);
      func_eval2 = CM.stress(atom_index, curr_position+direction*TempSegment.end);
      qmin_dist = (func_eval1 < func_eval2)? TempSegment.start : TempSegment.end;

      //change qmin_dist to be there
      new_position = curr_position + (direction*qmin_dist);

      //      cerr << "*5 new_position =>" << new_position << endl;
//       cerr << "*5 direction =>" << direction << endl;
//       cerr << "*5 direction.mag() =>" << direction.mag() << endl;
//       cerr << "*5 qmin =>" << qmin_dist << endl;

      // add new vdw
      addVDWSegments(SegList, new_position,  \
                     direction, CM, atom_index, qmin_dist);

      //      cerr << "*6 new_position =>" << new_position << endl;

      //  check viol
      TempSegment = SegList.QSortReduceViol(qmin_dist);

      // if no violation at current qmin_dist, then TempSegment will be 0,0 -- break out of loop.

    }   // END location loop


#ifdef GNOMAD_OUTPUT
    printf("  After vdw_segs\n");
    //    printf("    lbound, ubound = %lf %lf\n", *lbound, *ubound);
    //    printf("    num_segments = %d\n", *num_segments);
    printf ("Segment List:\n");
    for(int ixx=0; ixx < SegList.GetNum(); ixx++)
```

```
      printf("      start_ineq, end_ineq = %lf %lf\n", SegList.GetSeg(ixx).start, SegList.GetSeg(ixx).end );
    printf("\n");

    printf("  After viol_check\n");
    printf("    qmin = %lf\n\n", qmin_dist);
#endif

    //    cerr << "*7 new_position =>" << new_position << endl;
    return new_position;

  }

```

4.30.3.10 void gwNumerics::SolveVector (const jVector & *gradient*, const jVector &\ *hessian*, jVector & *result*) [static]

Result pointer is a [3x1] array of doubles that represent are the solution to Hessian (x1,y1,z1) = Gradient. GGG in Gnomad, x1,y1,z1 (result*) is variable p GGG

Referenced by bfgs_step().

4.30.3.11 int gwNumerics::SpreadR (int *I*, double *radius*, DLList< int > *Moved*, int *depth*, ConstraintManager & *CM*, jVector *start_position*, const jVector *pref_dir*) [static]

4.30.3.12 void gwNumerics::addLPERTSegments (Segments & *SegList*, const jVector & *curr_position*, \const jVector & *direction*, ConstraintManager &\ *CM*, int *atom_index*, double *qmin_dist*, \const jVector & *start_position*, const jVector & *new_position*, double & *max_r*) [static]

Definition at line 308 of file gw_numerics.C.

```
                                                                               {


#ifdef GNOMAD_OUTPUT
  printf("  In pert_seg\n");
  printf("    initial x0, y0, z0 = %lf  %lf  %lf\n", start_position[0], start_position[1], start_position[2]);
  printf("    initial x, y, z = %lf  %lf  %lf\n", curr_position[0], curr_position[1], curr_position[2]);
  printf("    initial direction vector = %lf  %lf  %lf\n", direction[0], direction[1], direction[2]);
  //  printf("    initial lbound, ubound = %lf %lf\n", *lbound, *ubound);
  //  printf("    num_segments = %d\n\n", *num_segments);
#endif

#ifdef JOB_VERBOSE
  cerr << "addLPERTSegments: starting " << endl;
#endif

  Segment TempSegment;
  //  double radius = LinearThreshold(sqrt(CM.stress(atom_index, curr_position)), LPertRadius);
  //double radius = sqrt(0.5*CM.stress(atom_index, curr_position));
  double radius = sqrt(0.5*MinDistVariance*CM.stress(atom_index, new_position));

  if (radius > max_r){
#ifdef JOB_VERBOSE2
    cerr << "addLPERTSegments: pert radius =    " << radius << endl;
    cerr << "RESETTING PERT RADIUS ARBITRARILY to " << radius << endl ;
#endif
    radius = max_r;
```

```
  }

#ifdef JOB_VERBOSE2
  cerr << "addLPERTSegments: pert stress =    " << CM.stress(atom_index, new_position)  << endl;
  cerr << "addLPERTSegments: pert dif vector =" << curr_position-start_position << endl;
  cerr << "addLPERTSegments: pert radius =    " << radius << endl;
  cerr << "addLPERTSegments: pert max_radius cutoff =    " << max_r << endl;
  cerr << "addLPERTSegments: pert mindistvar =    " << MinDistVariance << endl;
#endif
  //cerr << "r " << radius << endl;
  //radius = LinearThreshold(log10(CM.stress(atom_index, curr_position)+1), LPertRadius);
  //radius = LinearThreshold(LPertRadius[2]-(1/(CM.stress(atom_index, curr_position)+LPertRadius[0])), LPertRadius);
  // radius = radius * LPertFactor;
  if (radius>0) {
    TempSegment = LineIntersectSphere(start_position, radius, curr_position, direction);

#ifdef JOB_VERBOSE2
    cerr << "addLPERTSegments: pert segement =" << TempSegment << endl;
#endif

    if (TempSegment.start < -1e8 && TempSegment.end > 1e8) { // two number from Glenn's ass
      return;
    }

    if ((qmin_dist > TempSegment.start) && (qmin_dist < TempSegment.end)) {
#ifdef JOB_VERBOSE2
      cerr << "addLPERTSegments: segment caused perturbation" << endl;
#endif
      NumLP++;
      SegList.Add(TempSegment);
      // here is where Local Pert has bumped atom number atom_index'
    } else {
#ifdef JOB_VERBOSE2
      cerr << "addLPERTSegments: pert segment irrelevant" << endl;
#endif
      NumNotLP++;
    }

  }

```

4.30.3.13 void gwNumerics::addVDWSegments (Segments & *SegList*, const jVector & *curr_position*, \const jVector & *direction*, ConstraintManager &\ *CM*, int *atom_index*, double *offset* = 0) [static]

Definition at line 217 of file gw_numerics.C.

```
                                    {

#ifdef JOB_VERBOSE
  cerr << "addVDWSegments: starting " << endl;
  cerr << "addVDWSegments: offset = "<<offset << endl;
  if (direction.nrm() == 0)
    cerr << "addVDWSegments: Zero dir! (abort)" << endl;
#endif

  if (direction.nrm() == 0)
    return ;


  DLList<int> NeighborList = CM.approx_neighbor_i(curr_position, 3.7);
  DLList<int>::ConstIterator Nei_Iter;
```

```
  int other_index;
  double msd;  // Minimum Separation Distance (center to center)
  Segment TempSegment;

  for(Nei_Iter = NeighborList.GetConstIterator(); Nei_Iter; ++Nei_Iter){
    // BEGIN iterate over local neighbors
    other_index = *Nei_Iter;

#ifdef JOB_VERBOSE2
    cerr << "addVDWSegments: checking atom " << other_index << endl;
#endif

    //    msd = LinearThreshold((CM.stress(other_index)),VDWRadius);
    //msd = 3;
    //msd    moved into loop to prevent msd on atom = atom

    //    cerr << "Bonds for atom " << atom_index << ": " << CM.getMolecule().GetBonds(atom_index) << endl;

    // DEBUG
    //cerr << "vdw atom:     "<< other_index;
    //cerr << "vdw msd:   "<< msd;
    //cerr << "vdw position: "<< (CM.getMolecule())(other_index) << endl;

    // here use the Moved() atom info to only check VDW against moved atoms
    if ((other_index != atom_index) && (CM.constraintON(other_index))) {
      if ((UseOnlyMoved == false)
          || ((UseOnlyMoved == true)
              && (CM.getAtom(other_index).Moved() == true)
              ))
        {
          msd = (CM.getMolecule()).MSD(atom_index, other_index);

#ifdef JOB_VERBOSE2
          cerr << "addVDWSegments: atom " << other_index <<
            " passed move and self-atom criteria. msd = " << msd << endl;

          cerr << "x            " << curr_position << endl;
          cerr << "other atom x " << (CM.getMolecule())(other_index) << endl;
          cerr << "dir          " << direction << endl;
#endif

          TempSegment = LineIntersectSphere((CM.getMolecule())(other_index),
                                            msd, curr_position ,direction);
          // returns segment on 0,0 if not intersect
          //cerr << "vdw segment: " << TempSegment << endl;

#ifdef JOB_VERBOSE2
          cerr << "addVDWSegments: segment returned " << TempSegment
               << " for atom " << other_index << "; d = "
               << euclid ((CM.getMolecule())(other_index), curr_position)
               << endl;
#endif

          if (offset){
            TempSegment.start += offset;
            TempSegment.end += offset;
#ifdef JOB_VERBOSE2
          cerr << "addVDWSegments: segment offset to " << TempSegment
               << " for atom " << other_index << endl;
#endif

          }
          if (TempSegment.start != TempSegment.end) SegList.Add(TempSegment);
        }
    }
```

```
}

```

4.30.3.14 void gwNumerics::bfgs_step (const jVector & *x*, const jVector & *xp*, const jVector & *gf*, const jVector & *gfp*, jVector & *H*, jVector & *direction*) [static]

Definition at line 921 of file gw_numerics.C.

Referenced by gwRelaxAtom::run().

```
                                                      {

//      cerr << "D: -  H-row0= " << H[0] << " " << H[1] << " " << H[2] << endl;
//      cerr << "D: -  H-row1= " << H[1] << " " << H[3] << " " << H[4] << endl;
//      cerr << "D: -  H-row2= " << H[2] << " " << H[4] << " " << H[5] << endl;
//      cerr << "D:  -  x= " << x << endl;
//      cerr << "D:  -  xp= " << xp << endl;
//      cerr << "D:  -  gf= " << gf << endl;
//      cerr << "D:  -  gfp= " << gfp << endl;
//      cerr << "D:  -  p= " << direction << endl;
  // solves for p, the direction vector

  register int i;
  // double Acopy[9], bcopy[3], d1, d2, hs[3], h1[6], h2[6], s[3], st, temp, v[3];
  double st, d1, d2, temp, h1[6], h2[6], hs[3];
  jVector Acopy(9,0);
  jVector s;
  jVector v;

  st = 0.0;

  s = x - xp;
//   cerr << "D:  -  s= " << s << endl;
  v = gf - gfp;
//   cerr << "D:  -  v= " << v << endl;

  d1 = 1/(v*s);
//   cerr << "D:  -  d1= " << d1 << endl;

  h1[0] = d1*v[0]*v[0];
  h1[1] = d1*v[0]*v[1];
  h1[2] = d1*v[0]*v[2];
  h1[3] = d1*v[1]*v[1];
  h1[4] = d1*v[1]*v[2];
  h1[5] = d1*v[2]*v[2];

  temp = H[1]*s[0]*s[1] + H[2]*s[0]*s[2] + H[4]*s[1]*s[2];
  temp *= 2.0;
  d2 = H[0]*s[0]*s[0] + H[3]*s[1]*s[1] + H[5]*s[2]*s[2];
  d2 += temp;
  d2 = 1.0/d2;

//   cerr << "D:  -  d2= " << d2 << endl;

  hs[0] = H[0]*s[0] + H[1]*s[1] + H[2]*s[2];
  hs[1] = H[1]*s[0] + H[3]*s[1] + H[4]*s[2];
  hs[2] = H[2]*s[0] + H[4]*s[1] + H[5]*s[2];

  h2[0] = d2*hs[0]*hs[0];
  h2[1] = d2*hs[0]*hs[1];
  h2[2] = d2*hs[0]*hs[2];
```

```
h2[3] = d2*hs[1]*hs[1];
h2[4] = d2*hs[1]*hs[2];
h2[5] = d2*hs[2]*hs[2];

for(i=0; i<6; i++) H[i] = H[i] + h1[i] - h2[i];

Acopy[0] = H[0];
Acopy[1] = H[1];
Acopy[2] = H[2];
Acopy[3] = H[1];
Acopy[4] = H[3];
Acopy[5] = H[4];
Acopy[6] = H[2];
Acopy[7] = H[4];
Acopy[8] = H[5];


gwNumerics::SolveVector((-1)*gf, Acopy, direction);


```

4.30.3.15 void gwNumerics::reset_LP_Count (void) [inline, static]

Definition at line 191 of file gw_numerics.h.

```
                                            {
  NumLP = NumNotLP = 0;

```

4.30.4 Member Data Documentation

4.30.4.1 double gwNumerics::AveAbsDistRes = 0 [static]

the average residual over distance constraints.

Definition at line 11 of file globals.C.

4.30.4.2 double gwNumerics::LPertFactor [static]

Multiplication factor for local pert radius.

Definition at line 223 of file gw_numerics.h.

4.30.4.3 bool gwNumerics::LocalPerturb = true [static]

Setting for local perturbation activation.

Definition at line 20 of file globals.C.

4.30.4.4 double gwNumerics::MaxAbsDistRes = -1 * INF [static]

the max residual over distance constraints.

Definition at line 10 of file globals.C.

4.30.4.5 int gwNumerics::MaxNCuts = 10 [static]

Maximum number of cuts used in the cubic line search.

Definition at line 21 of file globals.C.

4.30.4.6 double gwNumerics::MinDistVariance = INF [static]

the minimum variance fron distance constraint data.

Definition at line 12 of file globals.C.

4.30.4.7 int gwNumerics::NumLP = 0 [static]

Definition at line 22 of file globals.C.

4.30.4.8 int gwNumerics::NumNotLP = 0 [static]

Definition at line 23 of file globals.C.

4.30.4.9 bool gwNumerics::UseOnlyMoved = true [static]

Flag to set vdw exclusion based on previously moved atoms only.

Definition at line 19 of file globals.C.

4.30.4.10 double gwNumerics::ls_deriv_tolerance = 0.0000001 [static]

Tolerance of convergence. This value is used to test the derivative of the evaluation function duringthe line search. (tau)

Definition at line 8 of file globals.C.

4.30.4.11 double gwNumerics::ls_dist_tolerance = 0.0001 [static]

Epsilon of useless movement. If an atom is moved less than this amount, it is assumed to be a trivial movement. (1e-4)

Definition at line 7 of file globals.C.

4.30.4.12 double gwNumerics::ls_succ_tolerance = 0.001 [static]

Tolerance value used to test if the line search function was successful. (alph in glenn wiliams code)

Definition at line 6 of file globals.C.

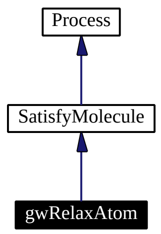
The documentation for this class was generated from the following files:

- **gw_numerics.h**
- **globals.C**
- **gw_numerics.C**

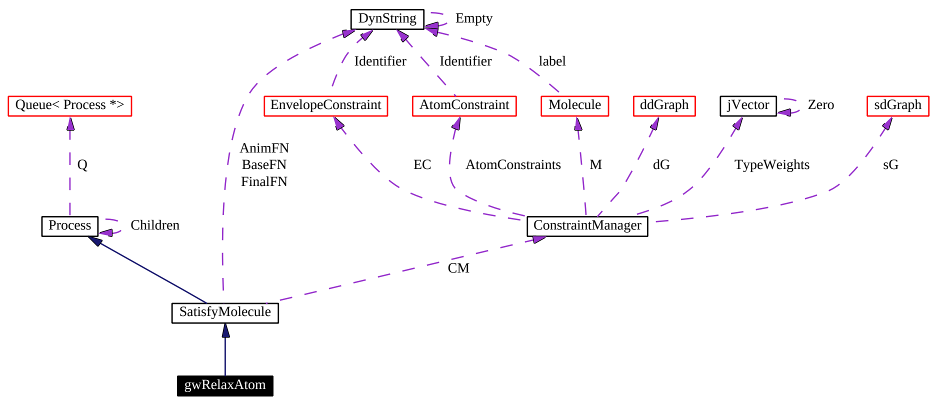
4.31 gwRelaxAtom Class Reference

`#include <gw_relax_atom.h>`

Inheritance diagram for gwRelaxAtom



Collaboration diagram for gwRelaxAtom:



Public Methods

- **gwRelaxAtom** (const gwRelaxAtom &ra)

  Copy Constructor.

- **gwRelaxAtom** (**ConstraintManager** &cm, int index)

  Constructor takes a reference to a constraint manager, and the index of the atom.

- **~gwRelaxAtom** ()
- int **run** ()
- int **getIndex** (void) const

  return index of the atom the class is acting on.

Static Public Attributes

- int **max_iterations** = 40

  Maximum nuber of times to perform linesearch while updating atom.

Private Attributes

- int **atom_index**

  The index of the atom in the molecule this process is associated with.

4.31.1 Detailed Description

///

File: **gw_relax_atom.h**

This class re-implements exactly the atom relaxtion code from Glenn Williams' Gnomad software. A modified Gauss Newton using the gradient and Hessian determine the optimal direction of travel for the atom, and a 2nd oder polynomial line search determines the best new position. Hard constraint provides both Van del Waals excusion and the ability to perform local pertubation.

///

Definition at line 25 of file gw_relax_atom.h.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 gwRelaxAtom::gwRelaxAtom (const gwRelaxAtom & *ra*)

Copy Constructor.

Definition at line 4 of file gw_relax_atom.C.

```
00005   : SatisfyMolecule(*(ra.CM))
00006 {
00007   atom_index = ra.atom_index;
00008
```

4.31.2.2 gwRelaxAtom::gwRelaxAtom (ConstraintManager & *cm*, int *temp_index*)

Constructor takes a reference to a constraint manager, and the index of the atom.

Definition at line 10 of file gw_relax_atom.C.

```
00011   : SatisfyMolecule(cm)
00012 {
00013   atom_index = temp_index;
00014   // gwRA assumes that static double ls_deriv_tolerance,
00015   // static double ls_dist_tolerance,
00016
```

4.31.2.3 gwRelaxAtom::~gwRelaxAtom ()

Definition at line 18 of file gw_relax_atom.C.

```
00019 {
00020   // Do *not* delete constraint manager (or M for that matter)!
00021
```

4.31.3 Member Function Documentation

4.31.3.1 int gwRelaxAtom::getIndex (void) const [inline]

return index of the atom the class is acting on.

Definition at line 71 of file gw_relax_atom.h.

```
00071                                          {return atom_index;}
00072
```

4.31.3.2 int gwRelaxAtom::run () [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 23 of file gw_relax_atom.C.

```
00024 {
00025
00026 #ifdef JOB_VERBOSE
00027   cerr <<endl<< "Start: \tgwRelaxAtom::run() atom=" << atom_index << endl;
00028 #endif
00029
00030 #ifdef JOB_VERBOSE2
00031   cerr << "Value: \tgwRelexAtom::run() priority atom " << atom_index << "before ="<< getPriority() << endl;
00032 #endif
00033
00034
00035 // #ifdef GNOMAD_OUTPUT
00036 //   printf("  In move_atom for atom %d\n\n", atom_index);
00037 // #endif
00038
00039   //  cerr << ";" << atom_index ;
00040
00041   jVector hessian(6,0.0);
00042   // 3x3 Hessian symmetric matrix of partial derivatives dr/di*dr/dj, where
00043   // i and j iterate over x,y,z dimension, and r is the residual
00044   // of this atom.
00045   // 0 1 2
00046   // 1 3 4
00047   // 2 4 5
00048
00049   jVector gradient(3,0.0);
00050   // Gradient vector.
00051
00052   jVector prev_gradient(3,0.0);
00053   // previous Gradient vector.
00054
00055   jVector direction(3,0.0);
00056   // Direction atom moves during each step inside iteration
00057
00058   jVector prev_direction(3,0.0);
00059   // previous Direction atom moves during each step inside iteration
00060
00061   jVector start_position = CM->getPosition(atom_index);
00062
00063   jVector step_position = start_position;
00064   // pulls the x,y,z coordinate of the atom before relaxation.
00065
00066   jVector prev_position = start_position;
```

```

  jVector new_position = start_position;

   //  jVector temp_vector(12,0.0);
  // FIX  a hack to hold the dStress(int) returnval.

  double step_length = 0;
  double new_step_length = 0;
  double derivative;
  int iteration = 0;
  register int i;
  bool ContinueFlag = true;

  for(i=0; i<3; i++) gradient[i] = 0.0;  // reset jVectors
  for(i=0; i<6; i++) hessian[i] = 0.0;

#ifdef GNOMAD_OUTPUT
  printf("    x0, y0, z0 = %lf, %lf, %lf\n", start_position[0], start_position[1], start_position[2]);
  printf("    f0 = %lf\n\n", CM->stress(atom_index));
#endif

  // 'iteration' loop
  while (ContinueFlag){

#ifdef JOB_VERBOSE2
    cerr << "Working\tgwRelaxAtom::run() start iteration number " << iteration << endl << endl;
    cerr << "Value: \tgwRelaxAtom::run() step_position (start of iteration)=  " << step_position <<endl;
    cerr << "Value: \tgwRelaxAtom::run() prev_position (start of iteration)=  " << prev_position <<endl;
    cerr << "Stress = "<< setprecision(25) <<CM->stress(atom_index,step_position) << setprecision(8)  <<endl;
#endif


    if (iteration >= max_iterations) {
      //      cerr << "X";
      break;  // break condition on iteration count
    }

    gradient = CM->dStress(atom_index, step_position);

    if (iteration == 0) {
      hessian[0] = hessian[3] =  hessian[5] =  1;
      direction = (-1)*gradient;

    } // first iteration only
    else { // all other iters

      gwNumerics::bfgs_step(step_position, prev_position,
                            gradient, prev_gradient,
                            hessian, direction);
      // sets hessian and direction
    }

    derivative = direction*gradient;
#ifdef JOB_VERBOSE2
    cerr << "Value: \tgwRelaxAtom::run() gradient = " << gradient << endl;
    cerr << "Value: \tgwRelaxAtom::run() hessian0 = "
         << hessian[0] << "\t" << hessian[1]  << "\t" << hessian[2]  << "\t" << endl;
    cerr << "Value: \tgwRelaxAtom::run() hessian1 = "
         << hessian[1]  << "\t" << hessian[3]  << "\t" << hessian[4] << endl;
    cerr << "Value: \tgwRelaxAtom::run() hessian2 = "
         << hessian[2]  << "\t" << hessian[4]  << "\t" << hessian[5] << endl;
    cerr << "Value: \tgwRelaxAtom::run() direction =  " << direction << endl;
    cerr << "Value: \tgwRelaxAtom::run() derivative =  " << derivative << endl;
#endif

```

```
    if (fabs(derivative) < gwNumerics::ls_deriv_tolerance) {
#ifdef JOB_VERBOSE2
      if (iteration == 0) {prev_direction = direction;}
      cerr << "break on derivative too small" << endl;
    cerr << "Value: \tgwRelaxAtom::breaking step_position =  " << step_position <<endl;
    cerr << "Value: \tgwRelaxAtom::breaking prev_position =  " << prev_position <<endl;
#endif
      break; // break condition on derivative
    }

    ContinueFlag = gwNumerics::CubicLineSearch(step_position, direction, \
                                   derivative, *CM, \
                                   atom_index, new_position, new_step_length);
    // both new_position and step_length are reset here

#ifdef JOB_VERBOSE2
    cerr << "Line search done, ContinueFlag = "<< ContinueFlag << endl;
    cerr << "Value: \tgwRelaxAtom::run() step_length =  " << step_length << endl;
    cerr << "Value: \tgwRelaxAtom::run() new_step_length =  " << new_step_length << endl;
    cerr << "Value: \tgwRelaxAtom::run() new_position =  " << new_position <<endl;
#endif

    if (ContinueFlag) {  // successful line search
      prev_position = step_position;
      step_position = new_position;
      prev_direction = direction;
      prev_gradient = gradient;
      step_length = new_step_length;
//       cerr << "Value: \tgwRelaxAtom::breaking step_length =  " << step_length <<endl;
//       cerr << "Value: \tgwRelaxAtom::breaking new_step_length =  " << new_step_length <<endl;

      //     } else {
//     if (iteration == 0){
//      prev_direction = direction;
//      WARN ("debug from gnomad");
//      cerr << "prev_direction =  " << prev_direction << endl;
//      cerr << "direction =       " << direction << endl;
//     }
    }

#ifdef JOB_VERBOSE2
    cerr << "Value: \tgwRelaxAtom::run() step_position (end of iteration)=  " << step_position << endl;
    cerr << "Working\tgwRelaxAtom::run() done iteration number " << iteration << endl << endl;
#endif

    iteration++;
  } // end of inner iteration loop

// #ifdef GNOMAD_OUTPUT
//     printf("  In vdw_segs\n");
//     printf("    initial x, y, z = %lf  %lf  %lf\n", prev_position[0], prev_position[1], prev_position[2]);
//     printf("    initial direction vector = %lf  %lf  %lf\n", prev_direction[0], prev_direction[1], prev_direction[2]);
//   //  printf("    initial lbound, ubound = %lf %lf\n", *lbound, *ubound);
//   //  printf("    num_segments = %d\n\n", *num_segments);
// #endif


  //  if (step_length != 1) cerr  << step_length << endl;

  /* do hard constraint (VDW) check. */
  //jVector ddd = prev_position;



//     cerr << "Value: \tgwRelaxAtom::after iterator step_position =  " << step_position <<endl;
```

```
//     cerr << "Value: \tgwRelaxAtom::after iterator prev_position =  " << prev_position <<endl;
//     cerr << "Value: \tgwRelaxAtom::after iterator step_length =  " << step_length <<endl;

    new_position = gwNumerics::SetHCPosition2(start_position, prev_position,prev_direction,
                                              *CM, atom_index, step_length, gwNumerics::MaxAbsDistRes);


//   jVector new_position2 = gwNumerics::SetHCPosition(start_position, prev_position,prev_direction,
//                                                     *CM, atom_index, step_length);
//
//   if (euclid(new_position, new_position2) > 0.5) {
//     cout << "\nACK:  atom=" << atom_index << " grd=" << new_position << " trd=" << new_position2 << flush;
//   }

    CM->setPosition(atom_index, new_position);

//   if (EC){
//     //    direction = EC->getDirection(new_position);
//     //  cerr << "atom_index:" << atom_index << "pos: " << new_position << "Dir:" << direction << endl;
//     CM->setPosition(atom_index, new_position+direction);
//   }


  //  DLList<int> Moved;
  //  CM->setPosition(atom_index, prev_position+(prev_direction*step_length));
  //  gwNumerics::LocalPert(atom_index, *CM, CM->stress(atom_index), start_position);
  //cerr <<"PRE-SPREAD DIST MOVED"<< euclid(prev_position,start_position) << endl;
  //  gwNumerics::SpreadR(atom_index, CM->getRadius(atom_index), Moved,
  //          0, *CM, CM->getPosition(atom_index), prev_direction);



  // shows how much Hard constraint would have moved the atom
  // ddd = (ddd -  CM->getMolecule()(atom_index));
  // cerr << ddd.mag() << endl;



// #ifdef GNOMAD_OUTPUT
//   printf("  after checks for atom %d\n", atom_index);
//   printf("    x, y, z = %lf  %lf  %lf\n", CM->getMolecule()(atom_index)[0],  CM->getMolecule()(atom_index)[1],
//       CM->getMolecule()(atom_index)[2]);
//   printf("    fv = %lf\n",  CM->stress(atom_index));
//
// //   printf("    atomcount = %d\n", );
// //   maxresid = 0.0;
// //   for(i=0; i<ndis; i++) {
// //     if(dis[i].resid>maxresid) {
// //       maxresid = dis[i].resid;
// //       maxatom1 = dis[i].atom1;
// //       maxatom2 = dis[i].atom2;
// //     }
// //   }

```

4.31.4 Member Data Documentation

4.31.4.1 int gwRelaxAtom::atom_index [private]

The index of the atom in the molecule this process is associated with.

Definition at line 67 of file gw_relax_atom.h.

4.31.4.2 int gwRelaxAtom::max_iterations = 40 [static]

Maximum nuber of times to perform linesearch while updating atom.

Definition at line 32 of file globals.C.

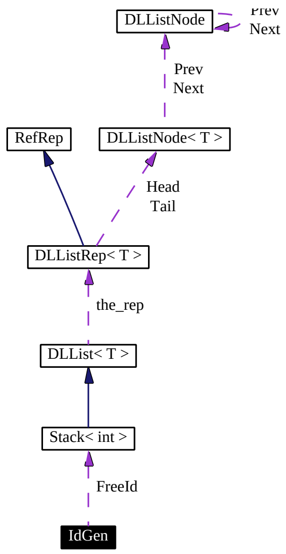
The documentation for this class was generated from the following files:

- **gw_relax_atom.h**
- **globals.C**
- **gw_relax_atom.C**

4.32 IdGen Class Reference

`#include <idgen.h>`

Collaboration diagram for IdGen:



Public Methods

- **IdGen** (int min=0)
- **~IdGen** ()
- int **Max** () const
- int **Peek** () const
- int **Get** ()
- void **Free** (int id)
- void **Reset** ()
- int **NumberIssued** () const
- int **NumberUsed** () const
- int **NumberFree** () const

Private Attributes

- **Stack**<int> **FreeId**
- int **MaxId**
- int **MinId**

Friends

- ostream& **operator**<< (ostream &os, const IdGen &idg)

4.32.1 Constructor & Destructor Documentation

4.32.1.1 IdGen::IdGen (int *min* = 0) [inline]

Definition at line 74 of file idgen.h.

```
{ MinId=min; MaxId = MinId; }
```

4.32.1.2 IdGen::~IdGen () [inline]

Definition at line 77 of file idgen.h.

```
{ /*empty*/ }
```

4.32.2 Member Function Documentation

4.32.2.1 void IdGen::Free (int *id*) [inline]

Definition at line 103 of file idgen.h.

Referenced by ddGraph::RemVertex().

```
{
  // Doesnt check for duplicates in the list...

  if(id>=MinId && id<MaxId)
      FreeId.Push(id);
```

4.32.2.2 int IdGen::Get () [inline]

Definition at line 90 of file idgen.h.

Referenced by ddGraph::AddVertex().

```
{
  int id;
  if(FreeId.IsEmpty())
    {
      id = MaxId;
      ++MaxId;
    }
```

4.32.2.3 int IdGen::Max () const [inline]

Definition at line 80 of file idgen.h.

```
{ return MaxId-1; }
```

4.32.2.4 int IdGen::NumberFree () const [inline]

Definition at line 120 of file idgen.h.

Referenced by NumberUsed().

```
00121   { return FreeId.Number(); }
00122
```

4.32.2.5 int IdGen::NumberIssued () const [inline]

Definition at line 117 of file idgen.h.

Referenced by NumberUsed().

```
00118   { return (MaxId-MinId); }
00119
```

4.32.2.6 int IdGen::NumberUsed () const [inline]

Definition at line 123 of file idgen.h.

```
00124   { return NumberIssued() - NumberFree(); }
00125
```

4.32.2.7 int IdGen::Peek () const [inline]

Definition at line 83 of file idgen.h.

Referenced by ddGraph::NextFreeId().

```
00084   {
00085     if(FreeId.IsEmpty())
00086         return MaxId;
00087     return FreeId.Peek();
00088
```

4.32.2.8 void IdGen::Reset () [inline]

Definition at line 111 of file idgen.h.

Referenced by ddGraph::Clear().

```
00112   {
00113     FreeId.Clear();
00114     MaxId = MinId;
00115
```

4.32.3 Friends And Related Function Documentation

4.32.3.1 ostream& operator<< (ostream & *os*, const IdGen & *idg*) [friend]

Definition at line 126 of file idgen.h.

```
00127   {
00128     os << '[' << idg.MaxId << "] {" << idg.FreeId << '}'
00129
```

4.32.4 Member Data Documentation

4.32.4.1 Stack< int > IdGen::FreeId [private]

Definition at line 63 of file idgen.h.

4.32.4.2 int IdGen::MaxId [private]

Definition at line 66 of file idgen.h.

4.32.4.3 int IdGen::MinId [private]

Definition at line 70 of file idgen.h.

The documentation for this class was generated from the following file:

- **idgen.h**

4.33 IntPriorityQueue Class Template Reference

`#include <ipqueue.h>`

Public Methods

- int **Copy** (const IntPriorityQueue< PT > &ipq)
- **IntPriorityQueue** (int maxsize=**IPQDEFLEN**, int rev=0)
- **IntPriorityQueue** (const IntPriorityQueue< PT > &ipq)
- ~**IntPriorityQueue** ()
- void **Build** ()
- int **IsEmpty** () const
- **operator int** () const
- int **IsFull** () const
- int **Size** () const
- int **Number** () const
- int **IsBad** () const
- int **BadId** (int id) const
- int **In** (int id) const
- void **Clear** ()
- int **Add** (int id, const PT &priority)
- PT **Peek** (int id) const
- int **ChangeId** (int oldid, int newid)
- void **Modify** (int id, const PT &newpriority)
- void **RelModify** (int id, const PT &del)
- int **Extreme** () const
- int **Remove** (int &id, PT &priority)
- const IntPriorityQueue<PT>& **operator=** (const IntPriorityQueue< PT > &ipq)
- const PT& **GetPriority** (int id) const
- const PT& **operator**[] (int id) const
- ostream& **PrintTop** (ostream &os, **uint** n) const

Private Methods

- **uint Parent** (**uint** i) const
- **uint Left** (**uint** i) const
- **uint Right** (**uint** i) const
- void **Swap** (**uint** i, **uint** j)
- int **FixDir** (**uint** i) const
- void **Heapify** (**uint** i, **uint** len)
- void **BubbleUp** (**uint** i)

Private Attributes

- int **Rev**
- **uint M**
- **uint N**
- int* **Index**
- int* **Id**
- PT* **Pri**
- int **built**

Friends

- ostream& **operator** (ostream &os, const IntPriorityQueue< PT > &Q)

template<class PT = double> class IntPriorityQueue

4.33.1 Constructor & Destructor Documentation

4.33.1.1 template<class PT> IntPriorityQueue< PT >::IntPriorityQueue<PT> (int *maxsize* = IPQDEFLEN, int *rev* = 0) `[inline]`

Definition at line 196 of file ipqueue.inc.

```
{
  M      = 0;
  N      = 0;
  Rev    = rev;
  built  = 0;

  Pri   = 0;
  Index = 0;
  Id    = 0;

  maxsize = (maxsize < 1) ? IPQDEFLEN : maxsize;

  if(!(Pri = new PT[maxsize+1]) ||
     !(Id  = new int[maxsize+1]) ||
     !(Index = new int[maxsize+1]))
    {
      delete[] Index; Index = 0;
      delete[] Id;    Id    = 0;
      delete[] Pri;     Pri     = 0;
      raise(SIGQUIT);
    }
```

4.33.1.2 template<class PT> IntPriorityQueue< PT >::IntPriorityQueue<PT> (const IntPriorityQueue< PT > & *ipq*) `[inline]`

Definition at line 183 of file ipqueue.inc.

```
{
  M     = 0;
  N     = 0;
  Rev   = 0;
  built = 0;
  Pri   = 0;
  Index = 0;
  Id    = 0;
  Copy(ipq);
```

4.33.1.3 template<class PT> IntPriorityQueue< PT >::~IntPriorityQueue<PT> () `[inline]`

Definition at line 223 of file ipqueue.inc.

```
{ delete[] Pri; Pri = 0; delete[] Id; Id=0; delete[] Index; Index = 0; }

```

4.33.2 Member Function Documentation

4.33.2.1 template<class PT> int IntPriorityQueue< PT >::Add (int *id*, const PT & *priority*) [inline]

Definition at line 252 of file ipqueue.inc.

Referenced by sdGraph::Dijkstra().

```
{
  int i;

  // See if the queue's full:
  // See if the id's ok:
  //
  if(IsBad() || BadId(id)) raise(SIGQUIT);
  if(IsFull() || In(id)) return 0;

  // If the heap has not been built yet, we can just stupidly copy the
  // new element into the last spot available in the array.
  if(!built)
    {
      Pri[N+1]      = priority;
      Index[id]   = N+1;
      Id[N+1]     = id;
      // ++N;
    }

  // Otherwise, we need to insert the new element and maintain the heap
  // structure of the array.
  else
    {
      i = N+1;
      if(!Rev)
        {
          while(i>1 && (Pri[Parent(i)] < priority))
            {
              if(BadId(i) || BadId(Parent(i))) raise(SIGQUIT);
              Pri[i] = Pri[Parent(i)];
              Id[i] = Id[Parent(i)];
              Index[Id[i]] = i;
              i = Parent(i);
            }
        }
      else
        {
          while(i>1 && (Pri[Parent(i)] > priority))
            {
              if(BadId(i) || BadId(Parent(i))) raise(SIGQUIT);
              Pri[i] = Pri[Parent(i)];
              Id[i] = Id[Parent(i)];
              Index[Id[i]] = i;
              i = Parent(i);
            }
        }
      Pri[i]      = priority;
      Id[i]     = id;
      Index[id] = i;
      //

```

```
      /*
      ++N;
      Pri[N]    = priority;
      Id[N]     = id;
      Index[id] = N;
      BubbleUp(N);
      */
    }

```

4.33.2.2 template<class PT> int IntPriorityQueue< PT >::BadId (int *id*) const [inline]

Definition at line 19 of file ipqueue.inc.

Referenced by Add(), BubbleUp(), ChangeId(), GetPriority(), In(), Modify(), Peek(), and RelModify().

```
{ return (id<0 || id > (int) M); }

```

4.33.2.3 template<class PT> void IntPriorityQueue< PT >::BubbleUp (uint *i*) [inline, private]

Definition at line 84 of file ipqueue.inc.

Referenced by Modify().

```
{
  // if(BadId(i)) raise(SIGQUIT);

  if(!Rev)
    {
      while(i>1 && (Pri[i] > Pri[Parent(i)]))
        {
          if(BadId(i) || BadId(Parent(i)))
              raise(SIGQUIT);
          Swap(i,Parent(i));
          i = Parent(i);
        }
    }
  else
    {
      while(i>1 && (Pri[i] < Pri[Parent(i)]))
        {
          if(BadId(i) || BadId(Parent(i)))
              raise(SIGQUIT);
          Swap(i,Parent(i));
          i = Parent(i);
        }
    }

```

4.33.2.4 template<class PT> void IntPriorityQueue< PT >::Build () [inline]

Definition at line 111 of file ipqueue.inc.

Referenced by Remove().

```
00112 {
00113   if(built) return;
00114   if(N>1)
00115       for(uint i = N>>1; i>=1; --i)
00116           Heapify(i,N);
00117   built=1;
00118
```

4.33.2.5 template<class PT> int IntPriorityQueue< PT >::ChangeId (int *oldid*, int *newid*) [inline]

Definition at line 352 of file ipqueue.inc.

```
00353 {
00354   if(IsBad() || BadId(oldid) || BadId(newid)) raise(SIGQUIT);
00355
00356   if(oldid == newid) return 1;
00357   if(!In(oldid) || In(newid))
00358       return 0;
00359
00360   int ndx = Index[oldid];
00361   Index[oldid]  = -1;
00362   Index[newid]  = ndx;
00363   Id[ndx] = newid;
00364   return 1;
00365
```

4.33.2.6 template<class PT> void IntPriorityQueue< PT >::Clear () [inline]

Definition at line 121 of file ipqueue.inc.

Referenced by IntPriorityQueue().

```
00122 {
00123   for(uint i=0; i<=M; ++i)
00124     {
00125       Index[i]  = -1;
00126       Id[i]     = -1;
00127       Pri[i]    = Rev ? INF : -INF;
00128     }
00129
```

4.33.2.7 template<class PT> int IntPriorityQueue< PT >::Copy (const IntPriorityQueue< PT > & *ipq*) [inline]

Definition at line 134 of file ipqueue.inc.

Referenced by IntPriorityQueue().

```
00135 {
00136   if(this == &ipq) return 1;
00137
00138   PT * tmpPri = Pri;
00139   int * tmpIndex  = Index;
00140   int * tmpId     = Id;
00141   if(ipq.M==0)
00142     {
00143       delete[] Pri;   Pri   = 0;
```

```
      delete[] Index; Index = 0;
      delete[] Id;    Id    = 0;
      N = M = 0;
      return 1;
    }
  if(M != ipq.M)
    {
      if(!(Pri   = new PT[ipq.M+1]) ||
         !(Index = new int[ipq.M+1])    ||
         !(Id    = new int[ipq.M+1]))
        {
          delete[] Pri;   Pri   = tmpPri;
          delete[] Index; Index = tmpIndex;
          delete[] Id;    Id    = tmpId;
          return 0;
        }
      delete[] tmpPri;
      delete[] tmpIndex;
      delete[] tmpId;
      M = ipq.M;
    }
  N = ipq.N;
  Rev = ipq.Rev;
  built = ipq.built;
  for(uint i=0; i<=M; ++i)
    {
      Pri[i]   = ipq.Pri[i];
      Index[i] = ipq.Index[i];
      Id[i]    = ipq.Id[i];
    }

```

4.33.2.8 template<class PT> int IntPriorityQueue< PT >::Extreme () const [inline]

Definition at line 400 of file ipqueue.inc.

```
{
  if(IsBad() || N<1)
      return -1;
  return Id[1];

```

4.33.2.9 template<class PT> int IntPriorityQueue< PT >::FixDir (uint *i*) const [inline, private]

Definition at line 317 of file ipqueue.inc.

Referenced by Modify().

```
{
  uint l=Left(i), r=Right(i);
  if(!Rev)
    {
      if(i>1 && Pri[i] > Pri[Parent(i)]) return 1;
      else
        {
          if((l<=N) && (Pri[i] < Pri[l]) ||
             (r<=N) && (Pri[i] < Pri[r]))
               return -1;
```

```
        }
    }
  else
    {
      if(i>1 && Pri[i] < Pri[Parent(i)]) return 1;
      else
        {
          if((l<=N) && (Pri[i] > Pri[l]) ||
             (r<=N) && (Pri[i] > Pri[r]))
               return -1;
        }
    }

```

4.33.2.10 template<class PT> const PT & IntPriorityQueue< PT >::GetPriority (int *id*) const `[inline]`

Definition at line 442 of file ipqueue.inc.

Referenced by operator[]().

```
{
  if(IsBad() || BadId(id)) raise(SIGQUIT);
  return Pri[Index[id]];

```

4.33.2.11 template<class PT> void IntPriorityQueue< PT >::Heapify (uint *i*, uint *len*) `[inline, private]`

Definition at line 42 of file ipqueue.inc.

Referenced by Build(), Modify(), and Remove().

```
{
  uint l = Left(i), r = Right(i), extreme;

  // if(BadId(i) || len>N)
  //  raise(SIGQUIT);

  if(!Rev)
    {
      if((l<=len) && (Pri[l] > Pri[i]))
        extreme = l;
      else
        extreme = i;

      if((r<=len) && (Pri[r]>Pri[extreme]))
        extreme = r;

      if(extreme != i)
        {
          Swap(i,extreme);
          Heapify(extreme, len);
        }
    }
  else
    {
      if((l<=len) && (Pri[l] < Pri[i]))
        extreme = l;
      else
```

```
        extreme = i;

      if((r<=len) && (Pri[r]<Pri[extreme]))
        extreme = r;

      if(extreme != i)
        {
          Swap(i,extreme);
          Heapify(extreme, len);
        }
    }

```

4.33.2.12 template<class PT> int IntPriorityQueue< PT >::In (int *id*) const [inline]

Definition at line 248 of file ipqueue.inc.

Referenced by Add(), ChangeId(), Modify(), Peek(), and RelModify().

```
{ return (BadId(id) ? 0 : (Index[id] != -1)); }

```

4.33.2.13 template<class PT> int IntPriorityQueue< PT >::IsBad () const [inline]

Definition at line 14 of file ipqueue.inc.

Referenced by Add(), ChangeId(), Extreme(), GetPriority(), Modify(), Peek(), PrintTop(), RelModify(), and Remove().

```
   { return 0; }

```

4.33.2.14 template<class PT> int IntPriorityQueue< PT >::IsEmpty () const [inline]

Definition at line 228 of file ipqueue.inc.

Referenced by sdGraph::Dijkstra().

```
{ return N==0; }

```

4.33.2.15 template<class PT> int IntPriorityQueue< PT >::IsFull () const [inline]

Definition at line 236 of file ipqueue.inc.

Referenced by Add().

```
{ return N==M; }

```

4.33.2.16 template< class PT > uint IntPriorityQueue< PT >::Left (uint *i*) const [inline, private]

Definition at line 6 of file ipqueue.inc.

Referenced by FixDir(), and Heapify().

```
{ return (i << 1); }
```

4.33.2.17 template< class PT > void IntPriorityQueue< PT >::Modify (int *id*, const PT & *newpriority*) [inline]

Definition at line 368 of file ipqueue.inc.

Referenced by sdGraph::Dijkstra(), and RelModify().

```
{
  if(IsBad() || BadId(id)) raise(SIGQUIT);
  if(!In(id)) return;

  uint ndx = Index[id];
  int dir;

  if(ndx>N) raise(SIGQUIT);

  Pri[ndx] = newpriority;

  if(built)
    {
      if((dir = FixDir(ndx)) == -1)   // Modify downward
          Heapify(ndx,N);
      else if(dir == 1)               // Modify upward
          BubbleUp(ndx);
    }
```

4.33.2.18 template< class PT > int IntPriorityQueue< PT >::Number () const [inline]

Definition at line 244 of file ipqueue.inc.

```
{ return (int) N; }
```

4.33.2.19 template< class PT > uint IntPriorityQueue< PT >::Parent (uint *i*) const [inline, private]

Definition at line 2 of file ipqueue.inc.

Referenced by Add(), BubbleUp(), and FixDir().

```
{ return (i >> 1); }
```

4.33.2.20 template<class PT> PT IntPriorityQueue< PT >::Peek (int *id*) const `[inline]`

Definition at line 344 of file ipqueue.inc.

```
{
  if(IsBad() || BadId(id)) raise(SIGQUIT);
  if(!In(id)) return -1.0;
  return Pri[ Index[id] ];
```

4.33.2.21 template<class PT> ostream & IntPriorityQueue< PT >::PrintTop (ostream & *os*, uint *n*) const `[inline]`

Definition at line 466 of file ipqueue.inc.

```
{
  if(IsBad())
    {
      os << "<<BADIPQUEUE>>";
      return os;
    }

  n = (n>N) ? N : n;
  for(uint i=1; i<=n; ++i)
    {
      os << Id[i];
      if(i!=n) os << ' ';
    }
```

4.33.2.22 template<class PT> void IntPriorityQueue< PT >::RelModify (int *id*, const PT & *del*) `[inline]`

Definition at line 390 of file ipqueue.inc.

```
{
  if(IsBad() || BadId(id)) raise(SIGQUIT);
  if(!In(id)) return;
  uint ndx = Index[id];
  if(ndx>N) raise(SIGQUIT);
  Modify(id, Pri[ndx]+del);
```

4.33.2.23 template<class PT> int IntPriorityQueue< PT >::Remove (int & *id*, PT & *priority*) `[inline]`

Definition at line 408 of file ipqueue.inc.

Referenced by sdGraph::Dijkstra().

```
{
  if(IsBad() || N==0)
    {
      priority = Rev ? INF : -INF;
      id = -1;
```

```
      return 0;
    }

  if(!built) Build();

  priority = Pri[1];
  id       = Id[1];
  if(N == 1)
    {
      Pri[1] = Rev ? INF : -INF;
      Id[1]  = -1;
      Index[id] = -1;
      N = 0;
      return 1;
    }

```

4.33.2.24 template<class PT> uint IntPriorityQueue< PT >::Right (uint *i*) const [inline, private]

Definition at line 10 of file ipqueue.inc.

Referenced by FixDir(), and Heapify().

```
{ return ((i << 1) | ((uint) 1)); }

```

4.33.2.25 template<class PT> int IntPriorityQueue< PT >::Size () const [inline]

Definition at line 240 of file ipqueue.inc.

```
{ return (int) M; }

```

4.33.2.26 template<class PT> void IntPriorityQueue< PT >::Swap (uint *i*, uint *j*) [private]

Definition at line 23 of file ipqueue.inc.

Referenced by BubbleUp(), and Heapify().

```
{
  // if(IsBad() || BadId(i) || BadId(j) || i>N || j>N) raise(SIGQUIT);
  if(i==j) return;

  PT d = Pri[i];
  Pri[i]   = Pri[j];
  Pri[j]   = d;

  int k        = Index[Id[i]];
  Index[Id[i]] = Index[Id[j]];
  Index[Id[j]] = k;

  k     = Id[i];
  Id[i] = Id[j];
  Id[j] = k;

```

4.33.2.27 template<class PT> IntPriorityQueue< PT >::operator int () const `[inline]`

Definition at line 232 of file ipqueue.inc.

```
{ return N>0; }

```

4.33.2.28 template<class PT> const IntPriorityQueue< PT > & IntPriorityQueue< PT >::operator= (const IntPriorityQueue< PT > & *ipq*) `[inline]`

Definition at line 179 of file ipqueue.inc.

```
{ Copy(ipq); return *this; }

```

4.33.2.29 template<class PT> const PT & IntPriorityQueue< PT >::operator[] (int *id*) const `[inline]`

Definition at line 449 of file ipqueue.inc.

```
{ return GetPriority(id); }

```

4.33.3 Friends And Related Function Documentation

4.33.3.1 template<class PT = double> ostream& operator (ostream & *os*, const IntPriorityQueue< PT > & *Q*) `[friend]`

4.33.4 Member Data Documentation

4.33.4.1 template<class PT = double> int * IntPriorityQueue<PT>::Id `[private]`

Definition at line 105 of file ipqueue.h.

4.33.4.2 template<class PT = double> int * IntPriorityQueue<PT>::Index `[private]`

Definition at line 102 of file ipqueue.h.

4.33.4.3 template<class PT = double> uint IntPriorityQueue<PT>::M `[private]`

Definition at line 96 of file ipqueue.h.

4.33.4.4 template<class PT = double> uint IntPriorityQueue<PT>::N `[private]`

Definition at line 99 of file ipqueue.h.

4.33.4.5 template<class PT = double> PT * IntPriorityQueue<PT>::Pri [private]

Definition at line 109 of file ipqueue.h.

4.33.4.6 template<class PT = double> int IntPriorityQueue<PT>::Rev [private]

Definition at line 91 of file ipqueue.h.

4.33.4.7 template<class PT = double> int IntPriorityQueue<PT>::built [private]

Definition at line 112 of file ipqueue.h.

The documentation for this class was generated from the following files:

- **ipqueue.h**
- **ipqueue.inc**

4.34 IntVec Class Reference

`#include <intvec.h>`

Public Methods

- int **AppendUnique** (int newval)
- int **Append** (int newval)
- int **Resize** (int newlen)
- int **Copy** (const IntVec &iv)
- **IntVec** (int len=**CreationLength**)
- **IntVec** (int len, int intital_value)
- **IntVec** (int len, int first_value, int delta_val)
- **IntVec** (const IntVec &iv)
- ~**IntVec** ()
- int **Length** () const
- **operator int** () const
- **operator int** * () const
- const IntVec& **operator=** (const IntVec &iv)
- bool **Contains** (int v) const
- const int& **Access** (int i) const
- int& **Access** (int i)
- const int& **operator**[] (int i) const
- int& **operator**[] (int i)
- const IntVec& **set** (int val)
- const IntVec& **setrange** (int val, int delta)
- const IntVec& **zero** ()
- int **dynread** (istream &is, const **DS** &compdelim="\t", const **DS** &vectdelim="\n")
- int **InputBinary** (istream &is)
- int **Input** (istream &is)
- int **OutputBinary** (ostream &os) const
- int **Output** (ostream &os) const
- **MAKE_IO_STREAM_OPERATORS** (IntVec)
- **MAKE_BIN_STREAM_OPERATORS** (IntVec)

Static Public Attributes

- const int **DefCreationLength** = 3
- int **CreationLength** = **DefCreationLength**
- char **IODelimiter** =

Private Attributes

- int* **I**
- int **L**

Friends

- int **operator==** (const IntVec &a, const IntVec &b)
- int **operator!=** (const IntVec &a, const IntVec &b)
- int **operator<=** (const IntVec &a, const IntVec &b)
- int **operator>=** (const IntVec &a, const IntVec &b)
- int **operator<** (const IntVec &a, const IntVec &b)
- int **operator>** (const IntVec &a, const IntVec &b)
- IntVec **cat** (const IntVec &a, const IntVec &b)

4.34.1 Constructor & Destructor Documentation

4.34.1.1 IntVec::IntVec (int *len* = CreationLength) `[inline]`

Definition at line 77 of file intvec.inc.

```
{ I = 0; L = 0; Resize(len); }

```

4.34.1.2 IntVec::IntVec (int *len*, int *initial_value*) `[inline]`

Definition at line 80 of file intvec.inc.

```
{ I = 0; L = 0; Resize(len); set(initial_value); }

```

4.34.1.3 IntVec::IntVec (int *len*, int *first_value*, int *delta*) `[inline]`

Definition at line 83 of file intvec.inc.

```
{ I = 0; L = 0; Resize(len); setrange(first_value, delta); }

```

4.34.1.4 IntVec::IntVec (const IntVec & *iv*) `[inline]`

Definition at line 86 of file intvec.inc.

```
{ I = 0; L = 0; Copy(iv); }

```

4.34.1.5 IntVec::~IntVec () `[inline]`

Definition at line 89 of file intvec.inc.

```
{ delete[] I; }

```

4.34.2 Member Function Documentation

4.34.2.1 int & IntVec::Access (int *i*) [inline]

Definition at line 107 of file intvec.inc.

```
{
  if(i<0 || i>=L) { cerr << "IntVec::Access: out of bounds\n"; raise(SIGQUIT); }
```

4.34.2.2 const int & IntVec::Access (int *i*) const [inline]

Definition at line 101 of file intvec.inc.

Referenced by operator[]().

```
{
  if(i<0 || i>=L) { cerr << "IntVec::Access const: out of bounds\n"; raise(SIGQUIT); }
```

4.34.2.3 int IntVec::Append (int *newval*) [inline]

Definition at line 27 of file intvec.inc.

Referenced by Residue::AddAtom().

```
{
  int * tmp = I;
  I = new int[L+1];
  memcpy(I,tmp, sizeof(int)*L);
  delete[] tmp;
  I[L]=newval;
  L++;
  return 1;
```

4.34.2.4 int IntVec::AppendUnique (int *newval*) [inline]

Definition at line 11 of file intvec.inc.

Referenced by Atom::AddBond(), Constraint::addAtomSupport(), and Constraint::addConstraintSupport().

```
{

  for (int i = 0; i < L; i++){
    if (I[i] == newval) return 1;
  }
```

4.34.2.5 bool IntVec::Contains (int *v*) const [inline]

Definition at line 52 of file intvec.inc.

Referenced by Atom::Bonded().

```
00052                                       {
00053   for (int i = 0; i < L; i++){
00054     if (I[i] == v) return true;
00055   }
00056
```

4.34.2.6 int IntVec::Copy (const IntVec & *iv*) [inline]

Definition at line 59 of file intvec.inc.

Referenced by IntVec(), and operator=().

```
00060   {
00061     if(&iv == this) return 1;
00062     if(!Resize(iv.L))
00063       return 0;
00064     memcpy(I,iv.I, sizeof(int)*L);
00065     return 1;
00066
```

4.34.2.7 int IntVec::Input (istream & *is*) [inline]

Definition at line 178 of file intvec.inc.

```
00179   {
00180     int w;
00181     for(int i=0; i<L; ++i)
00182     {
00183       if(!(is >> w)) return i;
00184       I[i] = w;
00185     }
00186
```

4.34.2.8 int IntVec::InputBinary (istream & *is*) [inline]

Definition at line 199 of file intvec.inc.

```
00200   {
00201     // Read "L":
00202     int e;
00203     is >= e;
00204     if(!Resize(e))
00205         { cerr << "IntVec::InputBinary\n"; raise(SIGQUIT); }
00206
```

4.34.2.9 int IntVec::Length () const [inline]

Definition at line 92 of file intvec.inc.

Referenced by Constraint::Input(), ConstraintManager::add(), Constraint::arity(), Constraint::assertValidity(), ConstraintManager::getAssocAtoms(), and Constraint::notifyRemove().

```
00093   { return L; }
00094
```

4.34.2.10 IntVec::MAKE_BIN_STREAM_OPERATORS (IntVec)

4.34.2.11 IntVec::MAKE_IO_STREAM_OPERATORS (IntVec)

4.34.2.12 int IntVec::Output (ostream & *os*) const `[inline]`

Definition at line 189 of file intvec.inc.

```
{
  for(int i=0; i<L; ++i)
  {
    os << I[i];
    if(i != L-1) os << IntVec::IODelimiter;
  }
```

4.34.2.13 int IntVec::OutputBinary (ostream & *os*) const `[inline]`

Definition at line 215 of file intvec.inc.

```
{
  // Write "L":
  //
  os <= L;

  // Write "I":
  //
  if(I && L>0)
    {
      os <= 1;
      os.write((char*)I,L*sizeof(int));
    }
```

4.34.2.14 int IntVec::Resize (int *newlen*) `[inline]`

Definition at line 39 of file intvec.inc.

Referenced by Atom::Atom(), Copy(), InputBinary(), IntVec(), Residue::Setup(), and dynread().

```
{
  if(L == newlen) return 1;
  if(newlen<1)
    { delete[] I; I=0; L=0; return 1; }
  int * tmp = I;
  if(!(I = new int[newlen]))
    { I=tmp; return 0; }
```

4.34.2.15 int IntVec::dynread (istream & *is*, const DS & *compdelim* = " \t", const DS & *vectdelim* = "\n")

Definition at line 25 of file intvec.C.

```
{
  if(is.eof() || is.bad()) return 0;
```

```
  DS old_delim = GetDSDelim();
  SetDSDelim(vectdelim);

  DLList<int> ComponentList;
  DS VecString;
  int component;

  // Read in the next string from the input stream:
  is >> VecString;
  DSIterator it(VecString,compdelim);

  // Extract the components from the string:
  for(; it; ++it)
    {
      component = ($\ast$it).ToInt();
      ComponentList.AddToBack(component);
    }
```

4.34.2.16 IntVec::operator int () const [inline]

Definition at line 95 of file intvec.inc.

```
{ return L; }
```

4.34.2.17 IntVec::operator int ∗ () const

Definition at line 17 of file intvec.C.

```
{
  int * i = 0;
  if(!I || L<1 || !(i = new int[L])) return 0;
  memcpy(i,I,sizeof(int)*L);
  return i;
```

4.34.2.18 const IntVec & IntVec::operator= (const IntVec & *iv*) [inline]

Definition at line 98 of file intvec.inc.

```
{ Copy(iv); return *this; }
```

4.34.2.19 int & IntVec::operator[] (int *i*) [inline]

Definition at line 116 of file intvec.inc.

```
{ return Access(i); }
```

4.34.2.20 const int & IntVec::operator[] (int *i*) const `[inline]`

Definition at line 113 of file intvec.inc.

```
{ return Access(i); }

```

4.34.2.21 const IntVec & IntVec::set (int *val*) `[inline]`

Definition at line 68 of file intvec.inc.

Referenced by IntVec(), and zero().

```
{ for(int i=0; i<L; ++i) I[i] = val; return *this; }

```

4.34.2.22 const IntVec & IntVec::setrange (int *val*, int *delta*) `[inline]`

Definition at line 71 of file intvec.inc.

Referenced by IntVec().

```
{ for(int i=0; i<L; ++i) I[i] = val+(i*delta); return *this; }

```

4.34.2.23 const IntVec & IntVec::zero () `[inline]`

Definition at line 74 of file intvec.inc.

```
{ return set(0); }

```

4.34.3 Friends And Related Function Documentation

4.34.3.1 IntVec cat (const IntVec & *a*, const IntVec & *b*) `[friend]`

Definition at line 233 of file intvec.inc.

```
{
  IntVec c(a.L+b.L);
  for(int i=0; i<a.L; ++i) { c.I[i] = a.I[i]; }
  for(int i=0; i<b.L; ++i) { c.I[a.L+i] = b.I[i]; }

```

4.34.3.2 int operator!= (const IntVec & *a*, const IntVec & *b*) `[friend]`

Definition at line 127 of file intvec.inc.

```
{ return !(a == b); }

```

4.34.3.3 int operator< (const IntVec & *a*, const IntVec & *b*) [friend]

Definition at line 154 of file intvec.inc.

```
{
  if(a.L > b.L) return 0;
  if(a.L < b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] > b.I[i]) return 0;
      if(a.I[i] < b.I[i]) return 1;
    }
```

4.34.3.4 int operator<= (const IntVec & *a*, const IntVec & *b*) [friend]

Definition at line 130 of file intvec.inc.

```
{
  if(a.L > b.L) return 0;
  if(a.L < b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] > b.I[i]) return 0;
      if(a.I[i] < b.I[i]) return 1;
    }
```

4.34.3.5 int operator== (const IntVec & *a*, const IntVec & *b*) [friend]

Definition at line 119 of file intvec.inc.

```
{
  if(a.L != b.L) return 0;
  for(int i=0; i<a.L; ++i)
      if(a.I[i] != b.I[i]) return 0;
  return 1;
```

4.34.3.6 int operator> (const IntVec & *a*, const IntVec & *b*) [friend]

Definition at line 166 of file intvec.inc.

```
{
  if(a.L < b.L) return 0;
  if(a.L > b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] < b.I[i]) return 0;
      if(a.I[i] > b.I[i]) return 1;
    }
```

4.34.3.7 int operator>= (const IntVec & *a*, const IntVec & *b*) `[friend]`

Definition at line 142 of file intvec.inc.

```
{
  if(a.L < b.L) return 0;
  if(a.L > b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] < b.I[i]) return 0;
      if(a.I[i] > b.I[i]) return 1;
    }
```

4.34.4 Member Data Documentation

4.34.4.1 int IntVec::CreationLength = DefCreationLength `[static]`

Definition at line 14 of file intvec.C.

4.34.4.2 const int IntVec::DefCreationLength = 3 `[static]`

Definition at line 13 of file intvec.C.

4.34.4.3 int ∗ IntVec::I `[private]`

Definition at line 90 of file intvec.h.

4.34.4.4 char IntVec::IODelimiter = `[static]`

Definition at line 15 of file intvec.C.

4.34.4.5 int IntVec::L `[private]`

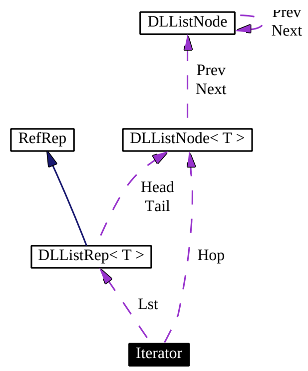
Definition at line 91 of file intvec.h.

The documentation for this class was generated from the following files:

- **intvec.h**
- **intvec.inc**
- **intvec.C**

4.35 Iterator Class Reference

Collaboration diagram for Iterator:



Public Methods

- void **Reset** ()
- T& **operator** ∗ ()
- **operator int** () const
- const Iterator& **operator++** ()
- void **Copy** (const Iterator &it)
- const Iterator& **operator=** (const Iterator &it)
- **Iterator** ()
- **Iterator** (const Iterator &it)
- **~Iterator** ()

Private Methods

- **Iterator** (**DLList**< T > &dll)

Private Attributes

- **DLListRep**<T>∗ **Lst**
- **DLListNode**<T>∗ **Hop**

Friends

- class **DLList**< **T** >

4.35.1 Constructor & Destructor Documentation

4.35.1.1 Iterator::Iterator () `[inline]`

Definition at line 31 of file dlliter.inc.

```
00031                          { Lst = 0; Hop = 0; }
00032
```

4.35.1.2 Iterator::Iterator (const Iterator & *it*) `[inline]`

Definition at line 32 of file dlliter.inc.

```
00033                 { Lst = 0; Hop = 0; Copy(it); }
00034
```

4.35.1.3 Iterator::∼Iterator () `[inline]`

Definition at line 34 of file dlliter.inc.

```
00034                           { if(Lst) Lst->Dereference(); Lst=0; }
00035
```

4.35.1.4 Iterator::Iterator (DLList< T > & *dll*) `[inline, private]`

Definition at line 36 of file dlliter.inc.

```
00037                 {
00038                   Lst = dll.the_rep;
00039                   if(Lst) Lst->Reference();
00040                   Reset();
00041
```

4.35.2 Member Function Documentation

4.35.2.1 void Iterator::Copy (const Iterator & *it*) `[inline]`

Definition at line 17 of file dlliter.inc.

Referenced by Iterator(), and operator=().

```
00018                 {
00019                   if(Lst != it.Lst)
00020                     {
00021                       if(Lst) Lst->Dereference();
00022                       Lst = it.Lst;
00023                       if(Lst) Lst->Reference();
00024                     }
00025
```

4.35.2.2 void Iterator::Reset () `[inline]`

Definition at line 12 of file dlliter.inc.

Referenced by Iterator().

```
00012                            { Hop = Lst->Head; }
00013
```

4.35.2.3 T & Iterator::operator ∗ () `[inline]`

Definition at line 13 of file dlliter.inc.

```
00013                               { return Hop->Data; }
00014
```

4.35.2.4 Iterator::operator int () const [inline]

Definition at line 14 of file dlliter.inc.

```
00014                                       { return Hop!=0; }
00015
```

4.35.2.5 const Iterator & Iterator::operator++ () [inline]

Definition at line 15 of file dlliter.inc.

```
00016                   { if(Hop) Hop = Hop->Next; return *this; }
00017
```

4.35.2.6 const Iterator & Iterator::operator= (const Iterator & *it*) [inline]

Definition at line 28 of file dlliter.inc.

```
00029                   { Copy(it); return *this; }
00030
```

4.35.3 Friends And Related Function Documentation

4.35.3.1 class DLList [friend]

Definition at line 10 of file dlliter.inc.

4.35.4 Member Data Documentation

4.35.4.1 DLListNode< T > ∗ Iterator::Hop [private]

Definition at line 43 of file dlliter.inc.

4.35.4.2 DLListRep< T > ∗ Iterator::Lst [private]

Definition at line 42 of file dlliter.inc.

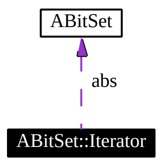
The documentation for this class was generated from the following file:

- **dlliter.inc**

4.36 ABitSet::Iterator Class Reference

`#include <abitset.h>`

Collaboration diagram for ABitSet::Iterator:



Public Methods

- **Iterator** (const **ABitSet** &iabs)
- **~Iterator** ()
- void **Reset** ()
- bool **operator++** ()
- bool **operator--** ()
- **uint operator** * () const
- **operator bool** () const

Protected Methods

- void **find_prev** ()
- void **find_next** ()

Private Attributes

- **uint i**
- const **ABitSet**& **abs**

4.36.1 Constructor & Destructor Documentation

4.36.1.1 ABitSet::Iterator::Iterator (const ABitSet & *iabs*)

Definition at line 376 of file abitset.C.

```
00377 : i(0)
00378 , abs(iabs)
00379 {
00380         find_next();
00381
```

4.36.1.2 ABitSet::Iterator::~Iterator ()

Definition at line 383 of file abitset.C.

```
00384 {
00385
```

4.36.2 Member Function Documentation

4.36.2.1 void ABitSet::Iterator::Reset ()

Definition at line 388 of file abitset.C.

```
{
        i = 0;
        find_next();
```

4.36.2.2 void ABitSet::Iterator::find_next () [protected]

Definition at line 422 of file abitset.C.

Referenced by Iterator(), Reset(), and operator++().

```
{
        while ( (i<abs.num) && (!abs[i]) )
                ++i;
```

4.36.2.3 void ABitSet::Iterator::find_prev () [protected]

Definition at line 429 of file abitset.C.

Referenced by operator–().

```
{
        while ( (i<abs.num) && (!abs[i]) )
                --i;
```

4.36.2.4 uint ABitSet::Iterator::operator * () const

Definition at line 400 of file abitset.C.

```
{
        return i;
```

4.36.2.5 ABitSet::Iterator::operator bool () const

Definition at line 394 of file abitset.C.

```
{
        return (i < abs.num);
```

4.36.2.6 bool ABitSet::Iterator::operator++ ()

Definition at line 414 of file abitset.C.

```
{
        ++i;
        find_next();
        return i < abs.num;
```

4.36.2.7 bool ABitSet::Iterator::operator– ()

Definition at line 406 of file abitset.C.

```
{
        --i;
        find_prev();
        return i < abs.num;
```

4.36.3 Member Data Documentation

4.36.3.1 const ABitSet & ABitSet::Iterator::abs `[private]`

Definition at line 33 of file abitset.h.

4.36.3.2 uint ABitSet::Iterator::i `[private]`

Definition at line 32 of file abitset.h.

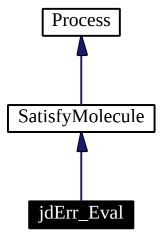
The documentation for this class was generated from the following files:

- **abitset.h**
- **abitset.C**

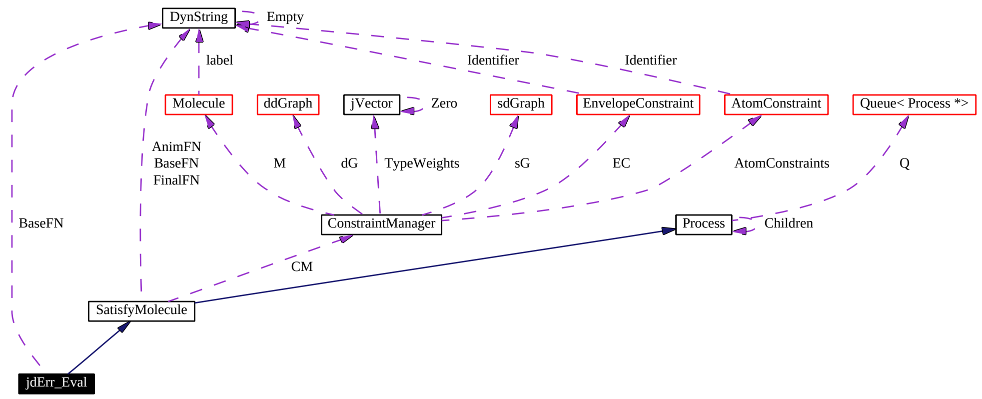
4.37 jdErr_Eval Class Reference

`#include <jd_err_eval.h>`

Inheritance diagram for jdErr_Eval



Collaboration diagram for jdErr_Eval:



Public Methods

- **jdErr_Eval** (**ConstraintManager** &**CM**, ifstream ∗inStream, int skip_in)
- **~jdErr_Eval** ()
- int **run** ()

Public Attributes

- int **skip**
- **DynString BaseFN**

Private Attributes

- ifstream∗ **mol_file**

4.37.1 Detailed Description

//

File: **jd_err_eval.h**

Created by: Jon Dugan

jdErr_Eval job.

//

Definition at line 25 of file jd_err_eval.h.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 jdErr_Eval::jdErr_Eval (ConstraintManager & *cm*, ifstream * *inStream*, int *skip_in* = 1)

Definition at line 3 of file jd_err_eval.C.

```
00004   : SatisfyMolecule(cm)
00005 {
00006   mol_file = inStream;
00007   skip =skip_in;
00008   if (skip < 1)
00009     DIE ("invalid slip number, use int > 0");
00010
```

4.37.2.2 jdErr_Eval::~jdErr_Eval ()

Definition at line 12 of file jd_err_eval.C.

```
00013   {
00014
```

4.37.3 Member Function Documentation

4.37.3.1 int jdErr_Eval::run () [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 16 of file jd_err_eval.C.

Referenced by main().

```
00016                   {
00017
00018   DynString FN = "err_out.xmol";
00019   AnimFrame::Enabled =true;
00020   AnimFrame *Out = new AnimFrame(*CM, FN);
00021   jVector E;
00022
00023 #ifdef JOB_VERBOSE
00024   cerr << "Start: \tErr_Eval::run()" << endl;
00025 #endif
00026
00027
00028   int a1 = 0;
00029   int a2 = 0;
00030   int num = CM->numAtoms();
00031   double d;
00032   double msd;
```

```
  int numerr = 0;

  cerr << "Checking msd: " ;

  for (a1 = 0;a1<num;a1++){
    for (a2 = a1+1;a2<num;a2++){
      d = euclid (CM->getPosition(a1),CM->getPosition(a2));
      msd = CM->getMolecule().MSD(a1, a2);
      if ( d < msd && ((d-msd)/msd > 1e-3)){
        numerr ++;
        cerr << endl << endl  << "atom1 = " << a1 << " "<< CM->getAtom(a1).Label() <<endl
             << "atom2 = " << a2 << " " << CM->getAtom(a2).Label() <<endl
             << "bonded = " << CM->getAtom(a1).Bonded(a2) << endl
             << "d = " << d <<endl
             << "msd = " << msd;
      }
    }
  }
  if (numerr)
    cerr << endl << "Found " << numerr << " error(s) with msd" << endl;
  else
    cerr << "ok" << endl << endl;

  if (CM->numConstraints() == 0){
    return 0;}

  int  i = 0;
  cerr << "Collecting constraint data..." << endl;
  E =  CM->getStressList();
  //  cerr << E << endl;
  cerr << i << " Total Error: " << E.sum() << endl;
  cerr << i << " Max   Error: " << E.max() << endl;

//   for (int i=0;i<CM->numAtoms();i++){
//     if (E[i] > 0.1){
//       cerr << CM->getAtom(i).Label() << " " << CM->getPosition(i) << " "
//         << E[i] << " " << CM->getEnvelope()->getGradient(CM->getPosition(i)) << endl;
//     }
//   }

  cerr << "======================================================================" << endl;

  mol_file->ipfx(); // gobbles whitespace & sets flags for good()

  cerr << "Writing xmol frame to " << FN << endl;
  Out->run();

  while (mol_file->good()){
    i++;

    //    M->writeXmol(cerr);
    //    CM->InvalidateAtoms();

    if (i%skip == 0){
      CM->readNewMoleculeXmol(*mol_file);
      CM->InvalidateEnvelope();
      E =  CM->getStressList();
      cerr << i << " Error: " << E.sum() << endl
           << "======================================================================" << endl;

      Out->run();
    } else {
      //CM->readNewMoleculeXmol(*mol_file);
      Molecule::skipXmol(*mol_file);
    }
```

```
      mol_file->ipfx();

  }

  delete Out;


  if (EnvelopeConstraint * E_ptr = CM->getEnvelope()){
    DynString F;
    char * s;
    ofstream * os;
    F = "err_out.iv";
    s = &(F[0]);  // ACK  not good, but it doesn't leak. -jd
    os = new ofstream (s);
    E_ptr->WriteVRML_OrigData(*os);
    delete os;
    cerr << "Wrote envelope data in inventor format to: " << F << endl;

    F = "err_out.csx";
    s = &(F[0]);  // ACK  not good, but it doesn't leak. -jd
    os = new ofstream (s);
    E_ptr->E[0]->WriteAvsTxt(*os);
    delete os;
    cerr << "Wrote envelope as constraint to: " << F << endl;
  }

```

4.37.4 Member Data Documentation

4.37.4.1 DynString jdErr_Eval::BaseFN

Reimplemented from **SatisfyMolecule** (p. 440).

Definition at line 34 of file jd_err_eval.h.

4.37.4.2 ifstream * jdErr_Eval::mol_file [private]

Definition at line 39 of file jd_err_eval.h.

4.37.4.3 int jdErr_Eval::skip

Definition at line 32 of file jd_err_eval.h.

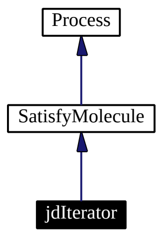
The documentation for this class was generated from the following files:

- **jd_err_eval.h**
- **jd_err_eval.C**

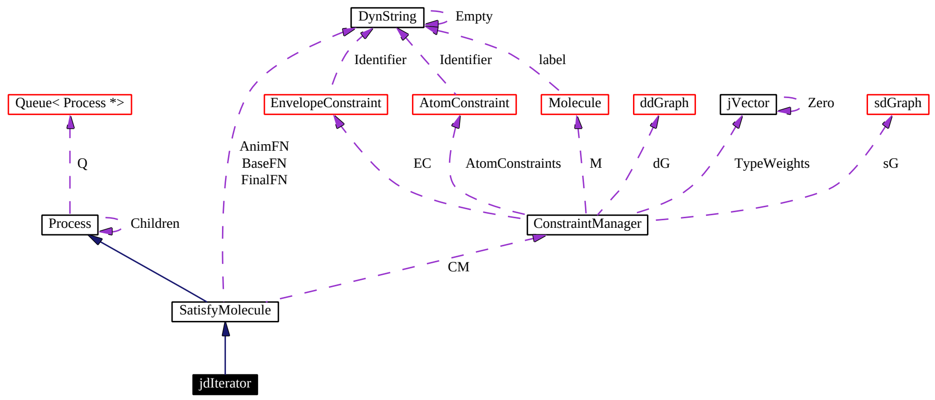
4.38 jdIterator Class Reference

`#include <jd_iterator.h>`

Inheritance diagram for jdIterator



Collaboration diagram for jdIterator:



Public Methods

- **jdIterator** (const jdIterator &ca)
- **jdIterator** (**ConstraintManager** &**CM**)
- **~jdIterator** ()
- int **run** ()

Static Public Attributes

- int **MaxIterations** = 2999

Private Attributes

- int **IterationNumber**

4.38.1 Detailed Description

//

File: **jd_iterator.h**

Created by: Jon Dugan

jdIterator job. iterates over all atoms based on error

//

Definition at line 28 of file jd_iterator.h.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 jdIterator::jdIterator (const jdIterator & *ca*)

Definition at line 3 of file jd_iterator.C.

```
  : SatisfyMolecule(*(ca.CM))
{
  DIE("jdIterator copy constructor called. ");

```

4.38.2.2 jdIterator::jdIterator (ConstraintManager & *cm*)

Definition at line 10 of file jd_iterator.C.

```
  : SatisfyMolecule(cm)
{


```

4.38.2.3 jdIterator::~jdIterator ()

Definition at line 16 of file jd_iterator.C.

```
  {
    // Don't delete my children (since Process takes care of it).
    // i.e. dont do for(int i=0; i<numChildren; ++i) delete Children[i].
    // or delete[] Children.

    // NB in THIS class I am not adding my children to the queue for
    // the parent class, Procss'  As a result, the children need to be
    // explicitly deleted.

    delete Children[0];
    delete Children[1];
    delete Children[2];
    delete Children[3];
    // Normally a job would NOT do this --
    // ~Process would do it, but I am not adding these to the queue

```

4.38.3 Member Function Documentation

4.38.3.1 int jdIterator::run () `[virtual]`

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 34 of file jd_iterator.C.

```
                    {

#ifdef JOB_VERBOSE
  cerr << "Start: \tIterator::run()" << endl;
#endif

  //CM->PrintGraph(cout);

  int StopFlag = 0;
  int iteration = 0;

  Children = new Process*[4];

  Children[0] = new CycleAll(*CM);
  Children[1] = new AnimFrame(*CM, AnimFN);
  Children[2] = new StopEval(*CM);
  Children[3] = new RelaxSubset(*CM, .4);


  while (!StopFlag && iteration < MaxIterations){

    // CYCLE and RELAX
    Children[0]->run();

    // RESET CYCLE ATOMS
    Children[0]->reset();


    //Children[3]->run();
    //Children[3]->run();


    // WRITE ANIMATION FRAME
    Children[1]->run();

    // EVALUATE STOP CONDITION
    StopFlag = Children[2]->run();  //editing this still -- performs evaluations and stop criteria

    iteration++;

  }

```

4.38.4 Member Data Documentation

4.38.4.1 int jdIterator::IterationNumber [private]

Definition at line 45 of file jd_iterator.h.

4.38.4.2 int jdIterator::MaxIterations = 2999 [static]

Definition at line 51 of file globals.C.

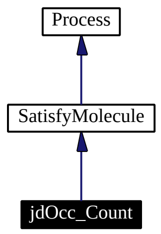
The documentation for this class was generated from the following files:

- **jd_iterator.h**
- **globals.C**
- **jd_iterator.C**

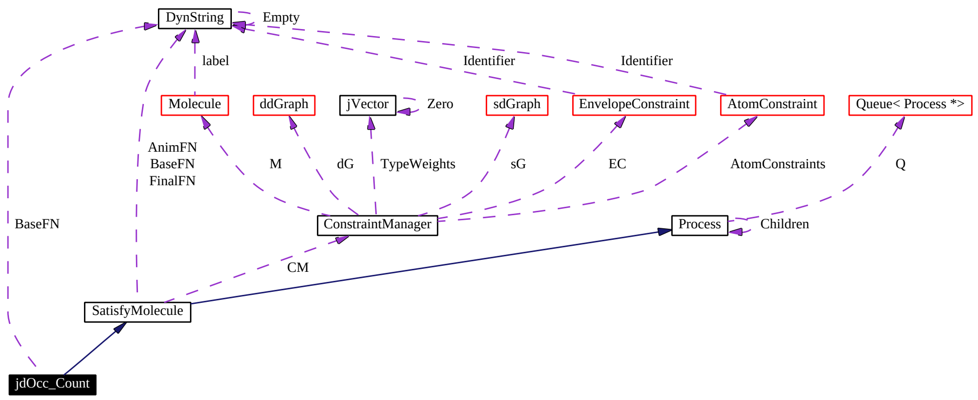
4.39 jdOcc_Count Class Reference

`#include <jd_occ_count.h>`

Inheritance diagram for jdOcc_Count



Collaboration diagram for jdOcc_Count:



Public Methods

- **jdOcc_Count** (**ConstraintManager** &**CM**)
- **~jdOcc_Count** ()
- int **run** ()

Public Attributes

- **DynString BaseFN**

Static Public Attributes

- double **cube_width** = 3.7

4.39.1 Detailed Description

//

File: **jd_occ_count.h**

Created by: Jon Dugan

jdOcc_Count job.

//

Definition at line 28 of file jd_occ_count.h.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 jdOcc_Count::jdOcc_Count (ConstraintManager & *cm*)

Definition at line 3 of file jd_occ_count.C.

```
00004   : SatisfyMolecule(cm)
00005 {
00006
```

4.39.2.2 jdOcc_Count::~jdOcc_Count ()

Definition at line 8 of file jd_occ_count.C.

```
00009   {
00010
```

4.39.3 Member Function Documentation

4.39.3.1 int jdOcc_Count::run () [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 12 of file jd_occ_count.C.

Referenced by main().

```
00012                      {
00013
00014 #ifdef JOB_VERBOSE
00015   cerr << "Start: \tOcc_Count::run() width =" << cube_width << endl;
00016 #endif
00017
00018   Envelope E;
00019   E.Build(CM->getMolecule(),cube_width);
00020
00021   DynString F;
00022   char * s;
00023   ofstream * os;
00024   F = BaseFN + ".csx";
00025   s = &(F[0]);  // ACK  not good, but it doesn't leak. -jd
00026   os = new ofstream (s);
00027   E.WriteAvsTxt(*os);
00028   delete os;
00029     cerr << "Wrote: " << F << endl;
00030
00031   F = BaseFN + ".iv";
00032   s = &(F[0]);  // ACK  not good, but it doesn't leak. -jd
00033   os = new ofstream (s);
```

```
  E.WriteVRML(*os);
  delete os;
  cerr << "Wrote: " << F << endl;


  F = BaseFN + ".density";
  s = &(F[0]);  // ACK  not good, but it doesn't leak. -jd
  os = new ofstream (s);
  E.WriteDensity(*os, CM->getMolecule());
  delete os;
  cerr << "Wrote: " << F << endl;


  //  E.Output(cout);



#ifdef JOB_VERBOSE
    cerr << "Ending:\tOcc_Count::run()" << endl;
#endif

    return 0; //Normal completion

```

4.39.4 Member Data Documentation

4.39.4.1 DynString jdOcc_Count::BaseFN

Reimplemented from **SatisfyMolecule** (p. 440).

Definition at line 37 of file jd_occ_count.h.

4.39.4.2 double jdOcc_Count::cube_width = 3.7 `[static]`

Definition at line 62 of file globals.C.

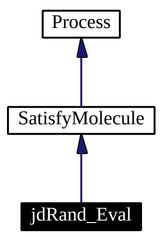
The documentation for this class was generated from the following files:

- **jd_occ_count.h**
- **globals.C**
- **jd_occ_count.C**

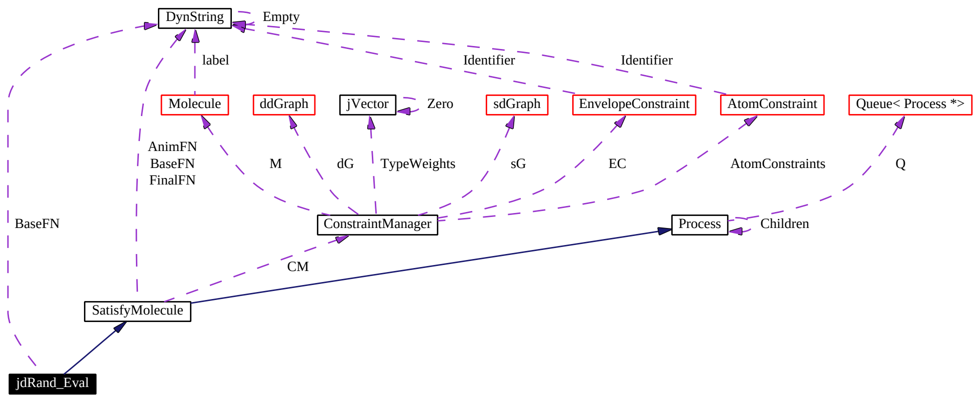
4.40 jdRand_Eval Class Reference

`#include <jd_rand_eval.h>`

Inheritance diagram for jdRand_Eval



Collaboration diagram for jdRand_Eval:



Public Methods

- **jdRand_Eval** (**ConstraintManager** &**CM**, ifstream ∗inStream, int cycle_in, double pert_in)
- **~jdRand_Eval** ()
- int **run** ()

Public Attributes

- int **cycles**
- double **pert**
- **DynString BaseFN**

Private Attributes

- ifstream∗ **mol_file**

4.40.1 Detailed Description

//

File: **jd_rand_eval.h**

Created by: Jon Dugan

jdRand_Eval job. perturb a molecule randomly and report on the satisfaction against a set of input constraints.

//

Definition at line 26 of file jd_rand_eval.h.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 jdRand_Eval::jdRand_Eval (ConstraintManager & *cm*, ifstream * *inStream*, int *c_in* = 10, double *p_in* = 0.05)

Definition at line 3 of file jd_rand_eval.C.

```
00005   : SatisfyMolecule(cm)
00006 {
00007   mol_file = inStream;
00008   cycles = c_in;
00009   pert = p_in;
00010
```

4.40.2.2 jdRand_Eval::~jdRand_Eval ()

Definition at line 12 of file jd_rand_eval.C.

```
00013   {
00014
```

4.40.3 Member Function Documentation

4.40.3.1 int jdRand_Eval::run () `[virtual]`

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 16 of file jd_rand_eval.C.

Referenced by main().

```
00016                    {
00017
00018   DynString FN = "rand_out.xmol";
00019   AnimFrame::Enabled =true;
00020   AnimFrame *Out = new AnimFrame(*CM, FN);
00021   jVector E;
00022
00023 #ifdef JOB_VERBOSE
00024   cerr << "Start: \tRand_Eval::run()" << endl;
00025 #endif
00026   int  i = 0;
```

```
E =  CM->getStressList();

// REPORT
cerr << "Frame: " << i << " Cycle: " << "original" << " Error: " << E.sum() << endl
     << "======================================================================" << endl;
Out->run();
for (int k = 1; k<=cycles;k++){
  CM->RandomPerturb(pert);
  CM->InvalidateEnvelope();
  E =  CM->getStressList();
  cerr << "Frame: " << i << " Cycle: " << k << " Error: " << E.sum() << endl
       << "======================================================================" << endl;
Out->run();
}


mol_file->ipfx(); // gobbles whitespace & sets flags for good()


while (mol_file->good()){
  i++;

  //    M->writeXmol(cerr);
  //    CM->InvalidateAtoms();


    CM->readNewMoleculeXmol(*mol_file);
    CM->InvalidateEnvelope();
    E =  CM->getStressList();

    // REPORT
    cerr << "Frame: " << i << " Cycle: " << "original" << " Error: " << E.sum() << endl
         << "======================================================================" << endl;
    Out->run();
    for (int k = 1; k<=cycles;k++){
      CM->RandomPerturb(pert);
      CM->InvalidateEnvelope();
      E =  CM->getStressList();
      cerr << "Frame: " << i << " Cycle: " << k << " Error: " << E.sum() << endl
           << "======================================================================" << endl;
    Out->run();
    }

    mol_file->ipfx(); //make sure input xmol file is still good

}

```

4.40.4 Member Data Documentation

4.40.4.1 DynString jdRand_Eval::BaseFN

Reimplemented from **SatisfyMolecule** (p. 440).

Definition at line 36 of file jd_rand_eval.h.

4.40.4.2 int jdRand_Eval::cycles

Definition at line 33 of file jd_rand_eval.h.

4.40.4.3 ifstream ∗ jdRand_Eval::mol_file [private]

Definition at line 40 of file jd_rand_eval.h.

4.40.4.4 double jdRand_Eval::pert

Definition at line 34 of file jd_rand_eval.h.

The documentation for this class was generated from the following files:

- **jd_rand_eval.h**
- **jd_rand_eval.C**

4.41 jVector Class Reference

`#include <jvector.h>`

Collaboration diagram for jVector:

Public Methods

- int **Resize** (int size)
- int **Copy** (const jVector &y)
- **operator double** * () const
- const double& **get** (int i) const
- void **set** (int i, const double &d)
- const jVector& **set** (const double &d)
- const jVector& **zero** ()
- **jVector** (int length=**CreationLength**)
- **jVector** (const jVector &v)
- **jVector** (int length, double d)
- **~jVector** ()
- const jVector& **transpose** ()
- int **Vertical** ()
- int **Length** () const
- double **sum** () const

  Returns the sum of all the components.

- double **sumsqr** () const

  Returns the sum of the squares.

- double **variance** () const

  Returns the variance.

- double **max** () const
- int **argmax** () const
- double **min** () const
- int **argmin** () const
- double **mean** () const
- double **median** () const
- double **nrm** () const

  Returns the norm (square of magnitude.

- jVector **norm_unit** () const

  Returns a jVector of unit length;.

- double **mag** () const

  returns magnitude.

- jVector **unit** () const

Returns a jVector of unit length;.

- const double& **operator**[ ] (int i) const

  indexing.

- double& **operator**[ ] (int i)
- jVector **operator-** ()

  negation.

- const jVector& **operator=** (const jVector &y)
- const jVector& **operator=** (const double &d)
- int **Compare** (const jVector &v) const
- const jVector& **operator-=** (const jVector &y)
- const jVector& **operator+=** (const jVector &y)
- const jVector& **operator%=** (const jVector &y)
- const jVector& **operator ∗=** (double d)
- const jVector& **operator/=** (double d)
- const jVector& **operator%=** (double d)
- int **Input** (istream &is)
- int **Output** (ostream &os) const
- int **dynread** (istream &is, const **DS** &compdelim="\t", const **DS** &vectdelim="\n")
- int **InputBinary** (istream &is)
- int **OutputBinary** (ostream &os) const
- **MAKE_IO_STREAM_OPERATORS** (jVector)
- **MAKE_BIN_STREAM_OPERATORS** (jVector)

Static Public Attributes

- int **CreationLength** = **DEFAULTLEN**
- char **IODelimiter** =
- jVector **Zero**

Private Methods

- void **mult** (double d)

Private Attributes

- double∗ **V**
- int **L**
- **usint vert**

Friends

- int **operator==** (const jVector &x, const jVector &y)

  Comparison operators.

- int **operator!=** (const jVector &x, const jVector &y)
- int **operator**< (const jVector &x, const jVector &y)

- int **operator>** (const jVector &x, const jVector &y)
- int **operator<=** (const jVector &x, const jVector &y)
- int **operator>=** (const jVector &x, const jVector &y)
- jVector **operator-** (jVector x, const jVector &y)
- jVector **operator+** (jVector x, const jVector &y)
- double **operator** ∗ (jVector x, const jVector &y)
- jVector **operator%** (jVector x, const jVector &y)
- jVector **operator** ∗ (jVector y, double d)
- jVector **operator** ∗ (double d, jVector y)
- jVector **operator/** (jVector y, double d)
- jVector **operator%** (jVector x, double d)
- double **JVAngle** (const jVector &x, const jVector &y)
- double **euclid** (const jVector &x, const jVector &y)
- ostream& **VBPrintLine** (ostream &os, const jVector &x, const jVector &y, const **DynString** &color)
- ostream& **VBPrintPoint** (ostream &os, const jVector &x, const **DynString** &color)
- ostream& **VBPrintCircle** (ostream &os, const jVector &x, const double &radius, const **DynString** &color)
- void **JVqsort** (jVector &v)
- int **binsearch** (const jVector &v, double d)
- double **Covariance** (const jVector &x, const jVector &y)

4.41.1 Constructor & Destructor Documentation

4.41.1.1 jVector::jVector (int *length* = CreationLength) `[inline]`

Definition at line 72 of file jvector.h.

```
00073   { L=0; V=0; vert=1; Resize(length); }
00074
```

4.41.1.2 jVector::jVector (const jVector & *v*) `[inline]`

Definition at line 75 of file jvector.h.

```
00076   { L=0; V=0; Copy(v); }
00077
```

4.41.1.3 jVector::jVector (int *length*, double *d*) `[inline]`

Definition at line 77 of file jvector.h.

```
00078   { L=0; V=0; Resize(length); set(d); }
00079
```

4.41.1.4 jVector::∼jVector () `[inline]`

Definition at line 80 of file jvector.h.

```
00080             { delete[] V; }
00081
```

4.41.2 Member Function Documentation

4.41.2.1 int jVector::Compare (const jVector & *v*) const [inline]

Definition at line 203 of file jvector.h.

```
{
  if(*this==v)
    return 0;
  if(*this>v)
    return 1;
  return -1;
```

4.41.2.2 int jVector::Copy (const jVector & *y*) [inline]

Definition at line 48 of file jvector.h.

Referenced by jVector(), and operator=().

```
{
  if(&y == this) return 1;
  if(!Resize(y.L)) return 0;
  memcpy(V, y.V, sizeof(double)*L);
  vert = y.vert;
  return 1;
```

4.41.2.3 int jVector::Input (istream & *is*)

Definition at line 165 of file jvector.C.

```
{
  double w;
  for(int i=0; i<L; ++i)
  {
    if(!(is >> w)) return i;
    V[i] = w;
  }
```

4.41.2.4 int jVector::InputBinary (istream & *is*)

Definition at line 281 of file jvector.C.

```
{
  // Read L:
  //
  int e;
  is >= e;
  if(!Resize(e))
      { cerr << "jVector::InputBinary\n"; raise(SIGQUIT); }
```

4.41.2.5 int jVector::Length () const [inline]

Definition at line 84 of file jvector.h.

Referenced by StopEval::check_stress_max_tot(), and AnimFrame::run().

```
00084                    { return L; }
00085
```

4.41.2.6 jVector::MAKE_BIN_STREAM_OPERATORS (jVector)

4.41.2.7 jVector::MAKE_IO_STREAM_OPERATORS (jVector)

4.41.2.8 int jVector::Output (ostream & *os*) const

Definition at line 176 of file jvector.C.

```
00177 {
00178   for(int i=0; i<L; ++i)
00179   {
00180     os << V[i];
00181     if(i != L-1) os << IODelimiter;
00182   }
00183
```

4.41.2.9 int jVector::OutputBinary (ostream & *os*) const

Definition at line 304 of file jvector.C.

```
00305   {
00306     // Write "L":
00307     //
00308     os <= L;
00309
00310     // Write "vert":
00311     //
00312     os <= vert;
00313
00314     // Write "V":
00315     //
00316     if(V && L>0)
00317       {
00318         os <= 1;
00319         os.write((char*)V,L*sizeof(double));
00320       }
00321
```

4.41.2.10 int jVector::Resize (int *size*) [inline]

Definition at line 34 of file jvector.h.

Referenced by Copy(), sdGraph::Dijkstra(), Grid::Grid(), InputBinary(), Grid::ReSize(), dynread(), and jVector().

```
00035   {
00036     if(L == size) return 1;
```

```
00037     if(size<1)
00038       { delete[] V; V = 0; L = 0; return 1; }
00039     double * tmp = V;
00040     if(!(V = new double[size]))
00041       { V = tmp; return 0; }
00042
```

4.41.2.11 int jVector::Vertical () [inline]

Definition at line 83 of file jvector.h.

```
00083                  { return (int) vert; }
00084
```

4.41.2.12 int jVector::argmax () const

Definition at line 64 of file jvector.C.

```
00065 {
00066   if(!L) return -1;
00067   double m = V[0];
00068   int mi = 0;
00069   for(int i=1; i<L; ++i) { if(m < V[i]) m = V[i]; mi = i; }
00070
```

4.41.2.13 int jVector::argmin () const

Definition at line 81 of file jvector.C.

```
00082 {
00083   if(!L) return -1;
00084   double m = V[0];
00085   int mi = 0;
00086   for(int i=1; i<L; ++i) { if(m > V[i]) m = V[i]; mi = i; }
00087
```

4.41.2.14 int jVector::dynread (istream & *is*, const DS & *compdelim* = " \t", const DS & *vectdelim* = "\n")

Definition at line 186 of file jvector.C.

Referenced by Molecule::DynRead().

```
00188   {
00189     if(is.eof() || is.bad()) return 0;
00190
00191     DS old_delim = GetDSDelim();
00192     SetDSDelim(vectdelim);
00193
00194     DLList<double> ComponentList;
00195     DS VecString;
00196     // DSIterator it;
00197     double component;
00198
00199     // Read in the next string from the input stream:
```

```
  is >> VecString;
  // cout << '[' << VecString << "]\n";
  // it.SetString(VecString);
  // it.SetDelim(compdelim);
  DSIterator it(VecString,compdelim);

  // Extract the components from the string:
  for(; it; ++it)
    {
      component = ($\ast$it).ToDbl();
      ComponentList.AddToBack(component);
    }

```

4.41.2.15 const double & jVector::get (int *i*) const `[inline]`

Definition at line 60 of file jvector.h.

```
{ return V[i]; }

```

4.41.2.16 double jVector::mag () const

returns magnitude.

Definition at line 110 of file jvector.C.

Referenced by EnvelopeConstraint::FindModelTranslationToCOM(), JVAngle(), EnvelopeConstraint::MoveModelbyGrad(), Envelope::WriteVRMLgr(), gwRelaxAtom::run(), and unit().

```
{ return sqrt(nrm()); }

```

4.41.2.17 double jVector::max () const

Definition at line 56 of file jvector.C.

Referenced by jdErr_Eval::run().

```
{
  if(!L) return INF;
  double m = V[0];
  for(int i=1; i<L; ++i) { if(m < V[i]) m = V[i]; }

```

4.41.2.18 double jVector::mean () const

Definition at line 100 of file jvector.C.

Referenced by Covariance().

```
{ return sum() / L; }

```

4.41.2.19 double jVector::median () const

Definition at line 91 of file jvector.C.

```
00092   {
00093     // Find the middle element.  Have to sort first:
00094     jVector sorted(*this);
00095     JVqsort(sorted);
00096     int i = (int) floor(L/2);
00097     return ( (L%2) ? sorted.V[i] : (sorted.V[i] + sorted.V[i+1]) );
00098
```

4.41.2.20 double jVector::min () const

Definition at line 73 of file jvector.C.

```
00074 {
00075   if(!L) return INF;
00076   double m = V[0];
00077   for(int i=1; i<L; ++i) { if(m > V[i]) m = V[i]; }
00078
```

4.41.2.21 void jVector::mult (double *d*) [private]

4.41.2.22 jVector jVector::norm_unit () const [inline]

Returns a jVector of unit length;.

Definition at line 106 of file jvector.h.

Referenced by Envelope::CalcGradient().

```
00106                            { return *this/nrm(); }
00107
```

4.41.2.23 double jVector::nrm () const

Returns the norm (square of magnitude.

Definition at line 103 of file jvector.C.

Referenced by EnvelopeConstraint::AlignModeltoCOM(), ConstraintManager::TranslateModel(), gwNumerics::addVDWSegments(), mag(), and norm_unit().

```
00104   {
00105     double sumsqr = 0.0;
00106     for(int i=0; i<L; ++i) sumsqr += V[i]*V[i];
00107     return sumsqr;
00108
```

4.41.2.24 const jVector & jVector::operator *= (double *d*)

Definition at line 138 of file jvector.C.

```
00139   { for(int i=0; i<L; ++i) V[i] *= d; return *this; }
00140
```

4.41.2.25 jVector::operator double * () const

Definition at line 20 of file jvector.C.

```
00021 {
00022   double * d = 0;
00023   if(!V || L<1 || !(d = new double[L]))
00024     { cerr << "jVector::operator double *\n"; raise(SIGQUIT); }
00025
```

4.41.2.26 const jVector & jVector::operator%= (double *d*)

Definition at line 147 of file jvector.C.

```
00148   { for(int i=0; i<L; ++i) V[i] = fmod(V[i],d); return *this; }
00149
```

4.41.2.27 const jVector & jVector::operator%= (const jVector & *y*)

Definition at line 132 of file jvector.C.

```
00133   { return y; }
00134
```

4.41.2.28 const jVector & jVector::operator+= (const jVector & *y*)

Definition at line 119 of file jvector.C.

```
00120   { if(L == y.L) for(int i=0; i<L; ++i) V[i] += y.V[i]; return *this; }
00121
```

4.41.2.29 jVector jVector::operator- () [inline]

negation.

Definition at line 129 of file jvector.h.

```
00130   {
00131     jVector x(L);
00132     for(int i=0; i<L; ++i)
00133       x.V[i] = -V[i];
00134     return x;
00135
```

4.41.2.30 const jVector & jVector::operator-= (const jVector & *y*)

Definition at line 115 of file jvector.C.

```
00116   { if(L == y.L) for(int i=0; i<L; ++i) V[i] -= y.V[i]; return *this; }
00117
```

4.41.2.31 const jVector & jVector::operator/= (double *d*)

Definition at line 143 of file jvector.C.

```
{ if(d==0.0) return *this; for(int i=0; i<L; ++i) V[i] /= d; return *this; }
```

4.41.2.32 const jVector & jVector::operator= (const double & *d*) `[inline]`

Definition at line 141 of file jvector.h.

```
{ return set(d); }
```

4.41.2.33 const jVector & jVector::operator= (const jVector & *y*) `[inline]`

Definition at line 138 of file jvector.h.

```
{ Copy(y); return *this; }
```

4.41.2.34 double & jVector::operator[] (int *i*) `[inline]`

Definition at line 121 of file jvector.h.

```
{
  if(i<0||i>=L)
    { cerr << "jVector::operator[]\n"; raise(SIGQUIT); }
```

4.41.2.35 const double & jVector::operator[] (int *i*) const `[inline]`

indexing.

Definition at line 115 of file jvector.h.

```
{
  if(i<0||i>=L)
    { cerr << "jVector::operator[]\n"; raise(SIGQUIT); }
```

4.41.2.36 const jVector & jVector::set (const double & *d*) `[inline]`

Definition at line 65 of file jvector.h.

```
{ for(int i=0; i<L; ++i) V[i]=d; return *this; }
```

4.41.2.37 void jVector::set (int *i*, const double & *d*) [inline]

Definition at line 62 of file jvector.h.

Referenced by jVector(), operator=(), and zero().

```
00063   { V[i]=d; }
00064
```

4.41.2.38 double jVector::sum () const

Returns the sum of all the components.

Definition at line 29 of file jvector.C.

Referenced by mean(), jdRand_Eval::run(), and jdErr_Eval::run().

```
00030   {
00031     if(!L) return INF;
00032     double s=0.0;
00033     for(int i=0; i<L; ++i)
00034         s += V[i];
00035     return s;
00036
```

4.41.2.39 double jVector::sumsqr () const

Returns the sum of the squares.

Definition at line 38 of file jvector.C.

```
00039   {
00040     if(!L) return INF;
00041     double s=0.0;
00042     for(int i=0; i<L; ++i)
00043         s += V[i]*V[i];
00044     return s;
00045
```

4.41.2.40 const jVector & jVector::transpose () [inline]

Definition at line 82 of file jvector.h.

```
00082                               { vert = vert ^ 1; return *this; }
00083
```

4.41.2.41 jVector jVector::unit () const [inline]

Returns a jVector of unit length;.

Definition at line 112 of file jvector.h.

Referenced by EnvelopeConstraint::MoveModelbyGrad().

```
00112                          { return *this/mag(); }
00113
```

4.41.2.42 double jVector::variance () const

Returns the variance.

Definition at line 47 of file jvector.C.

```
{
  if(!L) return INF;
  double s=0.0,ss=0.0;
  for(int i=0; i<L; ++i)
    { s += V[i]; ss += V[i]*V[i]; }

```

4.41.2.43 const jVector & jVector::zero () `[inline]`

Definition at line 68 of file jvector.h.

Referenced by Envelope::CalcGradient(), Grid::Grid(), EnvelopeConstraint::MoveModelbyGrad(), Grid::ReSize(), and Envelope::Upsize().

```
{ return set(0.0); }

```

4.41.3 Friends And Related Function Documentation

4.41.3.1 double Covariance (const jVector & *x*, const jVector & *y*) `[friend]`

Definition at line 326 of file jvector.C.

```
{
  if(x.L != y.L || x.L == 0)
      return INF;

  double x_mean = x.mean();
  double y_mean = y.mean();
  double cov = 0.0;

  for(int i=0; i<x.L; ++i)
      cov += (x[i] - x_mean)*(y[i] - y_mean);
  cov /= (x.L-1);

  return cov;

```

4.41.3.2 double JVAngle (const jVector & *x*, const jVector & *y*) `[friend]`

Definition at line 152 of file jvector.C.

```
{ return acos( (x*y) / x.mag() / y.mag() ); }

```

4.41.3.3 void JVqsort (jVector & *v*) `[friend]`

Definition at line 329 of file jvector.h.

Referenced by median().

```
                        { qsort((void *) v.V, v.L, sizeof(double), dcomp); }

```

4.41.3.4 ostream& VBPrintCircle (ostream & *os*, const jVector & *x*, const double & *radius*, const DynString & *color*) [friend]

Definition at line 264 of file jvector.C.

```
{

  if((x.L>3) || (x.L<2)) return os;
  char old_delim = jVector::IODelimiter;
  jVector::IODelimiter = ',';

  if(x.L==3)
    os << "circle (" << x << ") " << radius << " :" << color;
  else if(x.L==2)
    os << "circle (" << x << ",0) " << radius << " :" << color;
  os << '\n';
  jVector::IODelimiter = old_delim;
  return os;

```

4.41.3.5 ostream& VBPrintLine (ostream & *os*, const jVector & *x*, const jVector & *y*, const DynString & *color*) [friend]

Definition at line 226 of file jvector.C.

```
{
  if((x.L != y.L) || (x.L>3) || (y.L>3) || (x.L<1) || (y.L<1)) return os;

  char old_delim = jVector::IODelimiter;
  jVector::IODelimiter = ',';

  if(x.L==3)
    os << "line (" << x << ") (" << y << ") :" << color;
  else if(x.L==2)
    os << "line (" << x << ",0) (" << y << ",0) :" << color;
  else if(x.L==1)
    os << "line (" << x << "0,0) (" << y << "0,0) :" << color;
  os << '\n';
  jVector::IODelimiter = old_delim;
  return os;

```

4.41.3.6 ostream& VBPrintPoint (ostream & *os*, const jVector & *x*, const DynString & *color*) [friend]

Definition at line 245 of file jvector.C.

```
{

  if((x.L>3) || (x.L<1)) return os;
  char old_delim = jVector::IODelimiter;
  jVector::IODelimiter = ',';

  if(x.L==3)
```

```
    os << "point " << x << " :" << color;
  else if(x.L==2)
    os << "point " << x << ",0 :" << color;
  else if(x.L==1)
    os << "point " << x << ",0,0 :" << color;
  os << '\n';
  jVector::IODelimiter = old_delim;
  return os;

```

4.41.3.7 int binsearch (const jVector & *v*, double *d*) [friend]

Definition at line 299 of file jvector.h.

```
{
  // left index, right index, pivot, respectively.
  //
  int i=0;
  int j=v.L-1;
  int p;

  while((i<=j) && (j>=0))
    {
      p = (j+i)/2;
      if(d < v.V[p]) j = p-1;
      else if(d > v.V[p]) i = p+1;
      else return p;
    }

```

4.41.3.8 double euclid (const jVector & *x*, const jVector & *y*) [friend]

Definition at line 155 of file jvector.C.

```
  {
    if(x.L != y.L) return -1.0;
    double sumproducts = 0.0;
    for(int i=0; i<x.L; ++i) sumproducts += pow(x.V[i]-y.V[i], 2.0);
    return sqrt(sumproducts);

```

4.41.3.9 jVector operator * (double *d*, jVector *y*) [friend]

Definition at line 141 of file jvector.C.

```
                                          { return y*=d; }

```

4.41.3.10 jVector operator * (jVector *y*, double *d*) [friend]

Definition at line 140 of file jvector.C.

```
                                          { return y*=d; }

```

4.41.3.11 double operator * (jVector x, const jVector & y) [friend]

Definition at line 124 of file jvector.C.

```
00125 {
00126   double product_sum=0.0;
00127   if(x.L == y.L) for(int i=0; i<x.L; ++i) product_sum += x.V[i]*y.V[i];
00128   return product_sum;
00129
```

4.41.3.12 int operator!= (const jVector & x, const jVector & y) [friend]

Definition at line 152 of file jvector.h.

```
00153   { return !(x == y); }
00154
```

4.41.3.13 jVector operator% (jVector x, double d) [friend]

Definition at line 149 of file jvector.C.

```
00150   { return x%=d; }
00151
```

4.41.3.14 jVector operator% (jVector x, const jVector & y) [friend]

Definition at line 134 of file jvector.C.

```
00134                                                   { return x %= y; }
00135
```

4.41.3.15 jVector operator+ (jVector x, const jVector & y) [friend]

Definition at line 121 of file jvector.C.

```
00121                                                  { return (x += y); }
00122
```

4.41.3.16 jVector operator- (jVector x, const jVector & y) [friend]

Definition at line 117 of file jvector.C.

```
00117                                                  { return (x -= y); }
00118
```

4.41.3.17 jVector operator/ (jVector y, double d) [friend]

Definition at line 145 of file jvector.C.

```
00145                                          { return y/=d; }
00146
```

4.41.3.18 int operator< (const jVector & *x*, const jVector & *y*) [friend]

Definition at line 154 of file jvector.h.

```
{
  if(x.L < y.L) return 1;
  else if(x.L > y.L) return 0;
  // If the vectors are the same size:
  for(int i=x.L-1; i>=0; --i)
    {
      if(x.V[i] > y.V[i]) return 0;
      else if(x.V[i] < y.V[i]) return 1;
    }
```

4.41.3.19 int operator<= (const jVector & *x*, const jVector & *y*) [friend]

Definition at line 178 of file jvector.h.

```
{
  if(x.L < y.L) return 1;
  else if(x.L > y.L) return 0;
  // If the vectors are the same size:
  for(int i=x.L-1; i>=0; --i)
    {
      if(x.V[i] > y.V[i]) return 0;
      else if(x.V[i] < y.V[i]) return 1;
    }
```

4.41.3.20 int operator== (const jVector & *x*, const jVector & *y*) [friend]

Comparison operators.

Definition at line 145 of file jvector.h.

```
{
  if((x.L != y.L) || (x.vert != y.vert)) return 0;
  for(int i=x.L-1; i>=0; --i)
    if(x.V[i] != y.V[i]) return 0;
  return 1;
```

4.41.3.21 int operator> (const jVector & *x*, const jVector & *y*) [friend]

Definition at line 166 of file jvector.h.

```
{
  if(x.L > y.L) return 1;
  else if(x.L < y.L) return 0;
  // If the vectors are the same size:
  for(int i=x.L-1; i>=0; --i)
    {
      if(x.V[i] < y.V[i]) return 0;
      else if(x.V[i] > y.V[i]) return 1;
    }
```

4.41.3.22 int operator>= (const jVector & x, const jVector & y) [friend]

Definition at line 190 of file jvector.h.

```
{
  if(x.L > y.L) return 1;
  else if(x.L < y.L) return 0;
  // If the vectors are the same size:
  for(int i=x.L-1; i>=0; --i)
    {
      if(x.V[i] < y.V[i]) return 0;
      else if(x.V[i] > y.V[i]) return 1;
    }

```

4.41.4 Member Data Documentation

4.41.4.1 int jVector::CreationLength = DEFAULTLEN [static]

Definition at line 15 of file jvector.C.

4.41.4.2 char jVector::IODelimiter = [static]

Definition at line 16 of file jvector.C.

4.41.4.3 int jVector::L [private]

Definition at line 291 of file jvector.h.

4.41.4.4 double * jVector::V [private]

Definition at line 290 of file jvector.h.

4.41.4.5 jVector jVector::Zero [static]

Definition at line 18 of file jvector.C.

4.41.4.6 usint jVector::vert [private]

Definition at line 293 of file jvector.h.

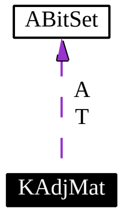
The documentation for this class was generated from the following files:

- **jvector.h**
- **jvector.C**

4.42 KAdjMat Class Reference

`#include <kadjmat.h>`

Collaboration diagram for KAdjMat:



Public Methods

- **KAdjMat** (int numvertices=**default_size**)
- **KAdjMat** (const KAdjMat &adj)
- **~KAdjMat** ()
- int **Copy** (const KAdjMat &adj)
- const KAdjMat& **operator=** (const KAdjMat &adj)
- int **Connect** (int i, int j)
- int **Disconnect** (int i, int j)
- const KAdjMat& **operator** ∗**=** (const KAdjMat &adj)
- const **ABitSet**& **operator**[ ] (int i) const
- const **ABitSet**& **operator()** (int i) const
- int **SetRow** (int i, const **ABitSet** &rowbits)
- int **SetCol** (int i, const **ABitSet** &colbits)

Static Public Methods

- int **DefaultSize** (int s)
- int **DefaultSize** ()

Static Public Attributes

- int **default_size**

Private Attributes

- int **N**
- **ABitSet**∗ **A**
- **ABitSet**∗ **T**

Friends

- KAdjMat **operator** ∗ (KAdjMat x, const KAdjMat &y)
- KAdjMat **pow** (KAdjMat adj, int n)
- ostream& **operator**<< (ostream &os, const KAdjMat &adj)

4.42.1 Constructor & Destructor Documentation

4.42.1.1 KAdjMat::KAdjMat (int *numvertices* = default_size)

4.42.1.2 KAdjMat::KAdjMat (const KAdjMat & *adj*)

4.42.1.3 KAdjMat::~KAdjMat ()

4.42.2 Member Function Documentation

4.42.2.1 int KAdjMat::Connect (int *i*, int *j*) [inline]

Definition at line 38 of file kadjmat.h.

Referenced by sdGraph::AdjMatrix().

```
{
  if(N<1 || i<0 || j<0 || i>=N || j>=N) return 0;
  A[i].Set((uint) j);
  T[j].Set((uint) i);
  return 1;

```

4.42.2.2 int KAdjMat::Copy (const KAdjMat & *adj*)

4.42.2.3 int KAdjMat::DefaultSize () [static]

Referenced by sdGraph::TravMatrices().

4.42.2.4 int KAdjMat::DefaultSize (int *s*) [static]

4.42.2.5 int KAdjMat::Disconnect (int *i*, int *j*) [inline]

Definition at line 47 of file kadjmat.h.

```
{
  if(N<1 || i<0 || j<0 || i>=N || j>=N) return 0;
  A[i].Clear((uint) j);
  T[j].Clear((uint) i);
  return 1;

```

4.42.2.6 int KAdjMat::SetCol (int *i*, const ABitSet & *colbits*)

4.42.2.7 int KAdjMat::SetRow (int *i*, const ABitSet & *rowbits*)

Referenced by sdGraph::TravMatrices().

4.42.2.8 const KAdjMat& KAdjMat::operator *= (const KAdjMat & *adj*)

4.42.2.9 const ABitSet & KAdjMat::operator() (int *i*) const [inline]

Definition at line 67 of file kadjmat.h.

```
00067                                       { return T[i]; }
00068
```

4.42.2.10 const KAdjMat& KAdjMat::operator= (const KAdjMat & *adj*)

4.42.2.11 const ABitSet & KAdjMat::operator[] (int *i*) const `[inline]`

Definition at line 65 of file kadjmat.h.

```
00065                                       { return A[i]; }
00066
```

4.42.3 Friends And Related Function Documentation

4.42.3.1 KAdjMat operator * (KAdjMat *x*, const KAdjMat & *y*) `[friend]`

4.42.3.2 ostream& operator<< (ostream & *os*, const KAdjMat & *adj*) `[friend]`

4.42.3.3 KAdjMat pow (KAdjMat *adj*, int *n*) `[friend]`

4.42.4 Member Data Documentation

4.42.4.1 ABitSet * KAdjMat::A `[private]`

Definition at line 79 of file kadjmat.h.

4.42.4.2 int KAdjMat::N `[private]`

Definition at line 78 of file kadjmat.h.

4.42.4.3 ABitSet * KAdjMat::T `[private]`

Definition at line 80 of file kadjmat.h.

4.42.4.4 int KAdjMat::default_size `[static]`

Definition at line 24 of file kadjmat.h.

The documentation for this class was generated from the following file:

- **kadjmat.h**

4.43 Matrix Class Reference

`#include <matrix.h>`

Public Methods

- **Matrix** ()
- **Matrix** (int in_Rows, int in_Cols)
- **Matrix** (const Matrix &m)
- **~Matrix** ()
- Matrix& **operator=** (const Matrix &m)
- double& **operator()** (int row, int col)
- const double& **operator()** (int row, int col) const
- double& **operator()** (int row)
- const double& **operator()** (int row) const
- void **setSize** (int Rows, int Cols)
- int **getRows** () const
- int **getCols** () const
- void **zero** ()
- void **identity** ()
- void **swapRows** (int row1, int row2)
- void **swapCols** (int col1, int col2)
- Matrix **subMatrix** (int row1, int row2, int col1, int col2) const
- Matrix **subMatrix** (int row, int col) const

  excludes row and col only – the rest is transferred.

- Matrix **transpose** () const
- Matrix **t** () const
- Matrix **operator+** (const Matrix &m) const
- Matrix **operator-** (const Matrix &m) const
- Matrix **operator** * (double d) const
- Matrix **operator** * (const Matrix &m) const
- double **det** () const

  NOTE: this is an inefficient implementation (Gauss-Jordan would probably be faster).

- double **tr** () const
- double **Sum** () const
- double **SumAbsoluteValue** () const
- void **findEigenVectors** (Matrix &eigenValues, Matrix &eigenVectors) const
- ostream& **PrettyPrint** (ostream &os)

  pretty print normal numbers.

Private Attributes

- int **rows**
- int **cols**
- double* **data**

4.43.1 Constructor & Destructor Documentation

4.43.1.1 Matrix::Matrix ()

Definition at line 9 of file matrix.C.

```
00009              {
00010   rows = cols = 0;
00011   data = 0;
00012
```

4.43.1.2 Matrix::Matrix (int *in_Rows*, int *in_Cols*)

Definition at line 14 of file matrix.C.

```
00014                                          {
00015   rows = in_Rows;
00016   cols = in_Cols;
00017   data = new double[rows*cols];
00018
```

4.43.1.3 Matrix::Matrix (const Matrix & *m*)

Definition at line 20 of file matrix.C.

```
00020                                 {
00021   rows = m.rows;
00022   cols = m.cols;
00023   data = new double[rows*cols];
00024   for (int i = 0; i < rows*cols; i++)
00025     data[i] = m.data[i];
00026
```

4.43.1.4 Matrix::~Matrix ()

Definition at line 28 of file matrix.C.

```
00028               {
00029   delete[] data;
00030   data = 0;
00031
```

4.43.2 Member Function Documentation

4.43.2.1 ostream & Matrix::PrettyPrint (ostream & *os*)

pretty print normal numbers.

Definition at line 388 of file matrix.C.

Referenced by EnvelopeConstraint::FindModelRotation().

```
00388                                            {
00389   for (int row = 1; row <= getRows(); row++) {
00390     os << setprecision(4) << "[ ";
```

```
00391     for (int col = 1; col <= getCols(); col++){
00392       if (fabs((*this)(row,col)> 1e-10))
00393         os << setw(8) << ($\ast$this)(row,col) << " ";
00394       else
00395         os << "0.000000 ";
00396     }
00397     os << "]" << endl;
00398   }
00399
```

4.43.2.2 double Matrix::Sum () const

Definition at line 159 of file matrix.C.

```
00159                         {
00160
00161   double rv = 0;
00162   for (int row = 1; row <= rows; row++)
00163     for (int col = 1; col <= cols; col++)
00164       rv += ($\ast$this)(row,col);
00165
00166   return rv;
00167
00168
```

4.43.2.3 double Matrix::SumAbsoluteValue () const

Definition at line 170 of file matrix.C.

```
00170                                       {
00171
00172   double rv = 0;
00173   for (int row = 1; row <= rows; row++)
00174     for (int col = 1; col <= cols; col++)
00175       rv += fabs((*this)(row,col));
00176
00177   return rv;
00178
00179
```

4.43.2.4 double Matrix::det () const

NOTE: this is an inefficient implementation (Gauss-Jordan would probably be faster).

Definition at line 332 of file matrix.C.

```
00332                         {
00333   if (rows != cols) {
00334     DIE("Matrix det() on non-square matrix");
00335   }
00336
00337   // Anton p.71
00338   if (rows == 1)
00339     return ($\ast$this)(1,1);
00340
00341   if (rows == 2)
00342     return ($\ast$this)(1,1)*($\ast$this)(2,2)-($\ast$this)(1,2)*($\ast$this)(2,1);
00343
```

```
if (rows == 3) {
  return \
    ($\ast$this)(1,1)*($\ast$this)(2,2)*($\ast$this)(3,3) \
    + ($\ast$this)(1,2)*($\ast$this)(2,3)*($\ast$this)(3,1) \
    + ($\ast$this)(1,3)*($\ast$this)(2,1)*($\ast$this)(3,2) \
    - ($\ast$this)(1,3)*($\ast$this)(2,2)*($\ast$this)(3,1) \
    - ($\ast$this)(1,2)*($\ast$this)(2,1)*($\ast$this)(3,3) \
    - ($\ast$this)(1,1)*($\ast$this)(2,3)*($\ast$this)(3,2) ;
}

DIE("Det is not working > 3x3 -- FIX");
double sum = 0;
for (int i = 1; i < cols; i++) {

  if (odd(i))
    sum += ($\ast$this)(1,i) * subMatrix(1,i).det();
  else
    sum -= ($\ast$this)(1,i) * subMatrix(1,i).det();

}

```

4.43.2.5 void Matrix::findEigenVectors (Matrix & *eigenValues*, Matrix & *eigenVectors*) const

Definition at line 230 of file matrix.C.

Referenced by Envelope::Calc_Eigenvectors(), and EnvelopeConstraint::FindModelRotation().

```
                                                                              {
if (rows != cols) {
  DIE("Matrix::findEigenVectors matrix must be square");
}
int i, j, row, col, n = rows, nrot = 0;
Matrix a=*this,b(n,1),z(n,1);
double tresh,theta,tau,t,s,h,g,c;

eigenValues.setSize(n,1);
eigenVectors.setSize(n,n);
eigenVectors.identity();
// initialize b and d to diagonal
for (i = 1; i <= n; i++) {
  b(i,1) = eigenValues(i,1) = a(i,i);
  z(i,1) = 0;
}
for (i = 1; i <= 50; i++) {
  double sum = 0;
  // sum off diagonal elements
  for (row = 1; row <= n-1; row++)
    for (col = row+1; col <= n; col++)
      sum += fabs(a(row,col));
  if (sum == 0) {  // normal return (relies on quadratic convergence to machine underflow)
    // sort eigenvalues and eigenvectors in descending order
    bool keepGoing=true;
    while (keepGoing) {
      keepGoing = false;
      for (row = 1; row <= n-1; row++)
        if (eigenValues(row,1) < eigenValues(row+1,1)) {
          eigenValues.swapRows(row,row+1);
          eigenVectors.swapCols(row,row+1);
          keepGoing = true;
        }
    }
```

```
      return;
    }
    if (i < 4)
      tresh = 0.2*sum/sq(n);  // first three steps
    else
      tresh = 0;              // thereafter
    for (row = 1; row <= n-1; row++)
      for (col = row+1; col <= n; col++) {
        g = 100*fabs(a(row,col));
        // after 4 sweeps, skip rotation if off-diagonal element is small
        if ((i > 4) && (fabs(eigenValues(row,1)+g) == fabs(eigenValues(row,1))) &&
                       (fabs(eigenValues(col,1)+g) == fabs(eigenValues(col,1))))
          a(row,col) = 0;
        else if (fabs(a(row,col)) > tresh) {
          h = eigenValues(col,1) - eigenValues(row,1);
          if (fabs(h)+g == fabs(h))
            t = a(row,col)/h;
          else {
            theta = 0.5*h/a(row,col);
            t = 1.0/(fabs(theta)+sqrt(1.0+sq(theta)));
            if (theta < 0.0)  t = -t;
          }
          c = 1.0/sqrt(1+sq(t));
          s = t*c;
          tau = s/(1.0+c);
          h = t*a(row,col);
          z(row,1) -= h;
          z(col,1) += h;
          eigenValues(row,1) -= h;
          eigenValues(col,1) += h;
          a(row,col) = 0;
          for (j = 1; j <= row-1; j++) {
            g = a(j,row);
            h = a(j,col);
            a(j,row) = g-s*(h+g*tau);
            a(j,col) = h+s*(g-h*tau);
          }
          for (j = row+1; j <= col-1; j++) {
            g = a(row,j);
            h = a(j,col);
            a(row,j) = g-s*(h+g*tau);
            a(j,col) = h+s*(g-h*tau);
          }
          for (j = col+1; j <= n; j++) {
            g = a(row,j);
            h = a(col,j);
            a(row,j) = g-s*(h+g*tau);
            a(col,j) = h+s*(g-h*tau);
          }
          for (j = 1; j <= n; j++) {
            g = eigenVectors(j,row);
            h = eigenVectors(j,col);
            eigenVectors(j,row) = g-s*(h+g*tau);
            eigenVectors(j,col) = h+s*(g-h*tau);
          }
          nrot++;
        }
      }
    for (int row = 1; row <= n; row++) {
      b(row,1) += z(row,1);
      eigenValues(row,1) = b(row,1);
      z(row,1) = 0;
    }
  }
```

4.43.2.6 int Matrix::getCols () const [inline]

Definition at line 42 of file matrix.h.

Referenced by PrettyPrint(), operator ∗(), and operator<<().

```
00042                     { return cols; }
00043
```

4.43.2.7 int Matrix::getRows () const [inline]

Definition at line 41 of file matrix.h.

Referenced by PrettyPrint(), operator ∗(), and operator<<().

```
00041                     { return rows; }
00042
```

4.43.2.8 void Matrix::identity ()

Definition at line 104 of file matrix.C.

Referenced by EnvelopeConstraint::AlignPrincComp(), and findEigenVectors().

```
00104                     {
00105   if (rows != cols) {
00106     DIE("Matrix must be square to make identity");
00107   }
00108
```

4.43.2.9 Matrix Matrix::operator ∗ (const Matrix & *m*) const

Definition at line 212 of file matrix.C.

```
00212                                             {
00213   if (cols != m.rows) {
00214     DIE("Matrix multiplication left side cols != right side rows");
00215   }
00216   Matrix result(rows,m.cols);
00217   for (int row = 1; row <= rows; row++)
00218     for (int col = 1; col <= m.cols; col++) {
00219       double sum = 0;
00220       for (int i = 1; i <= cols; i++)
00221         sum += ($\ast$this)(row,i) * m(i,col);
00222       result(row,col) = sum;
00223     }
00224
```

4.43.2.10 Matrix Matrix::operator ∗ (double *d*) const

Definition at line 182 of file matrix.C.

```
00182                                      {
00183   Matrix result(rows,cols);
00184   for (int row = 1; row <= rows; row++)
```

```
00185     for (int col = 1; col <= cols; col++)
00186       result(row,col) = d * ($\ast$this)(row,col);
00187   return result;
00188
```

4.43.2.11 const double & Matrix::operator() (int *row*) const

Definition at line 75 of file matrix.C.

```
00075                                              {
00076   if (rows == 1 || cols == 1) {
00077     if ((row < 1) || (row > rows)) {
00078       DIE("Matrix access out of bounds");
00079     }
00080     return data[row-1];
00081
00082   } else {
00083     DIE("Matrix singular access on 2D Matrix");
00084     return data[row-1];
00085   }
00086
```

4.43.2.12 double & Matrix::operator() (int *row*)

Definition at line 60 of file matrix.C.

```
00060                                  {
00061   if (rows == 1 || cols == 1) {
00062     if ((row < 1) || (row > rows)) {
00063       DIE("Matrix access out of bounds");
00064     }
00065     return data[row-1];
00066
00067   } else {
00068     DIE("Matrix singular access on 2D Matrix");
00069     return data[row-1];
00070   }
00071
```

4.43.2.13 const double & Matrix::operator() (int *row*, int *col*) const

Definition at line 53 of file matrix.C.

```
00053                                                       {
00054   if ((row < 1) || (row > rows) || (col < 1) || (col > cols)) {
00055     DIE("Matrix access out of bounds");
00056   }
00057
```

4.43.2.14 double & Matrix::operator() (int *row*, int *col*)

Definition at line 46 of file matrix.C.

```
00046                                            {
00047   if ((row < 1) || (row > rows) || (col < 1) || (col > cols)) {
00048     DIE("Matrix access out of bounds");
00049   }
00050
```

4.43.2.15 Matrix Matrix::operator+ (const Matrix & *m*) const

Definition at line 190 of file matrix.C.

```
00190                                       {
00191   if ((cols != m.cols) || (rows != m.rows)) {
00192     DIE("Matrix addition sizes are different");
00193   }
00194
```

4.43.2.16 Matrix Matrix::operator- (const Matrix & *m*) const

Definition at line 201 of file matrix.C.

```
00201                                       {
00202   if ((cols != m.cols) || (rows != m.rows)) {
00203     DIE("Matrix subtraction sizes are different");
00204   }
00205
```

4.43.2.17 Matrix & Matrix::operator= (const Matrix & *m*)

Definition at line 33 of file matrix.C.

```
00033                                  {
00034   if (this == &m)  return *this;  // handle self-assignment
00035   if (data)
00036     delete[] data;
00037
00038   rows = m.rows;
00039   cols = m.cols;
00040   data = new double[rows*cols];
00041   for (int i = 0; i < rows*cols; i++)
00042     data[i] = m.data[i];
00043   return *this;
00044
```

4.43.2.18 void Matrix::setSize (int *Rows*, int *Cols*)

Definition at line 89 of file matrix.C.

Referenced by findEigenVectors().

```
00089                                {
00090   if (data)
00091     delete[] data;
00092   rows = Rows;
00093   cols = Cols;
00094   data = new double[rows*cols];
00095
```

4.43.2.19 Matrix Matrix::subMatrix (int *row*, int *col*) const

excludes row and col only – the rest is transferred.

Definition at line 135 of file matrix.C.

```
00135                                             {
00136   Matrix result(rows-1,cols-1);
00137   int row2 = 1, col2 = 1;
00138   int row1 = 1, col1 = 1;
00139   for (row1 = 1; row1 <= rows; row1++) {
00140     col2 = 1;
00141     for (col1 = 1; col1 <= cols; col1++) {
00142       if ((row1 != row) && (col1 != col))
00143         result(row2,col2) = ($\ast$this)(row1,col1);
00144       if (col1 != col)  col2++;
00145     }
00146     if (row1 != row)  row2++;
00147   }
00148
```

4.43.2.20 Matrix Matrix::subMatrix (int *row1*, int *row2*, int *col1*, int *col2*) const

Referenced by det().

4.43.2.21 void Matrix::swapCols (int *col1*, int *col2*)

Definition at line 121 of file matrix.C.

Referenced by findEigenVectors().

```
00121                                       {
00122   for (int row = 1; row <= rows; row++)
00123     util_swap((*this)(row,col1),($\ast$this)(row,col2));
00124
```

4.43.2.22 void Matrix::swapRows (int *row1*, int *row2*)

Definition at line 116 of file matrix.C.

Referenced by findEigenVectors().

```
00116                                       {
00117   for (int col = 1; col <= cols; col++)
00118     util_swap((*this)(row1,col),($\ast$this)(row2,col));
00119
```

4.43.2.23 Matrix Matrix::t () const `[inline]`

Definition at line 81 of file matrix.h.

Referenced by EnvelopeConstraint::FindModelRotation().

```
00081                            {return transpose();}
00082
```

4.43.2.24 double Matrix::tr () const

Definition at line 368 of file matrix.C.

Referenced by EnvelopeConstraint::FindModelRotation().

```
                      {

if (rows == 1)
  return ($\ast$this)(1,1);

if (rows == 2)
  return ($\ast$this)(1,1)+($\ast$this)(2,2);

if (rows == 3) {
  return ($\ast$this)(1,1)+($\ast$this)(2,2)+($\ast$this)(3,3);
}

```

4.43.2.25 Matrix Matrix::transpose () const

Definition at line 151 of file matrix.C.

Referenced by t().

```
                             {
Matrix result(cols,rows);
for (int row = 1; row <= rows; row++)
  for (int col = 1; col <= cols; col++)
    result(col,row) = ($\ast$this)(row,col);
return result;

```

4.43.2.26 void Matrix::zero ()

Definition at line 97 of file matrix.C.

Referenced by Envelope::Calc_Eigenvectors(), and EnvelopeConstraint::FindModelRotation().

```
                {
for (int row = 1; row <= rows; row++)
  for (int col = 1; col <= cols; col++)
    //      (*this)(row,col) = 0;
    data[(col-1)+(row-1)*cols] = 0;

```

4.43.3 Member Data Documentation

4.43.3.1 int Matrix::cols [private]

Definition at line 76 of file matrix.h.

4.43.3.2 double * Matrix::data [private]

Definition at line 77 of file matrix.h.

4.43.3.3 int Matrix::rows [private]

Definition at line 75 of file matrix.h.

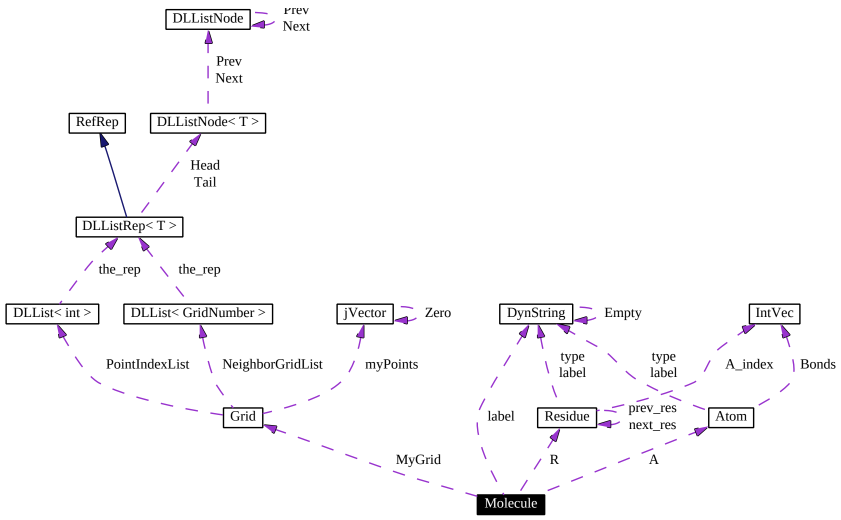
The documentation for this class was generated from the following files:

- **matrix.h**
- **matrix.C**

4.44 Molecule Class Reference

`#include <molecule.h>`

Collaboration diagram for Molecule:



Public Methods

- **Molecule** ()
- **Molecule** (int num_atoms)

  Constructs an atomic structure.

- **Molecule** (const **Atom** ∗atoms, int num_atoms)
- **Molecule** (const **jVector** ∗points_3d, int num_pos)
- **Molecule** (const Molecule &mol)
- **∼Molecule** ()
- int **Number** () const

  Reports the number of atoms in the molecule.

- int **NumRes** () const

  Reports the number of residues in the molecule.

- const Molecule& **operator=** (const Molecule &mol)

  Assignment (calls the **Copy***()* (p. 361) *function).*

- const **Atom**& **getAtom** (int i) const

  access to an atom.

- void **setAtom** (int i, const **Atom** &a)

  make atom a the i'th atom.

- const **Grid**& **getGrid** () const

  talk to the grid (note const).

- const **Atom**& **operator**[ ] (int i) const

  Access an atom in the molecule by the atom id. Note, the id and index of an atom are the same.

- **Atom**& **operator**[ ] (int i)

  Set an atom in the molecule by the atom id. Note, the id and index of an atom are the same.

- **DLList**<int> **approx_neighbor_i** (const **jVector** &p, double distance) const

  grid interface to neighbor atoms (Approx, fast).

- **DLList**<int> **neighbor_i** (const **jVector** &p, double distance) const

  grid interface to neighbor atoms (Exact, same as above, but checked).

- const **jVector**& **operator()** (int i) const

  returns position of atom i.

- const **jVector**& **getPosition** (int i) const

  returns position of atom i.

- void **setPosition** (int i, const **jVector** &newPos)

  sets position of atom i.

- **jVector getCOM** () const

  return center of mass.

- double **VDW** (int i) const

  VDW for atom i'.

- const **IntVec**& **GetBonds** (int atom) const

  returns list of bonds in an atom.

- double **MSD** (int a1, int a2) const

  return physical Minimum Separation Distance between two atoms.

- int **Output** (ostream &os) const
- int **Input** (istream &is)
- int **OutputBinary** (ostream &os) const
- int **InputBinary** (istream &is)
- istream& **DynRead** (istream &is)
- int **setRandom** (void)

  Generates random positions for Num' atoms.

- int **readXmol** (istream &is)

  Reads in a molecule from an xyz formatted file. Returns the number of atoms read in.

- int **readFasta** (istream &is)

  Reads in a molecule from an xyz formatted file. Returns the number of atoms read in.

- void **writeXmol** (ostream &os) const

Writes a file in xmol xyz format. This version puts the atom int atom number as the first Xmol field.

- void **writeXmolData** (ostream &os, int ∗data, **DynString** comment) const

  Writes a client specified integer for each atom in the output Xmol formatted file.

- void **writeXmolData** (ostream &os, double ∗data, **DynString** comment) const

  Writes a client specified double for each atom in the output Xmol formatted file.

- void **Bounds** (double &minx, double &miny, double &minz,\double &maxx, double &maxy, double &maxz) const

  writes the upper and lower bounds in 3 dimensions by reference.

- **MAKECOMPARISONS_NO_COMPARE** (Molecule)
- **MAKE_IO_STREAM_OPERATORS** (Molecule)
- **MAKE_BIN_STREAM_OPERATORS** (Molecule)

Static Public Methods

- void **skipXmol** (istream &is)

  skip a frame in a xmol format file stream.

Static Public Attributes

- double **grid_delta** = 3.7

  expected inter-atomic width.

- double **grid_max_radius** = 250

  expected max molecule size.

Protected Methods

- void **Allocate** (int n)

  Allocates space for n atoms.

- void **Deallocate** ()

  Deallocates the space.

- void **Reallocate** (int n)

  Destroys the currently allocated space if n!=N and creates space for n atoms.

- void **Copy** (const Molecule &mol)

  Copies a molecule...

Protected Attributes

- **DynString label**

  label for the molecule.

- int **N**

  The number of atoms in the structure.

- int **P**

  The number of residues in the structure.

- **Atom* A**

  A is a vector of atoms in the molecule (0-> N-1).

- **Residue* R**

  R is a vector of residues in the molecule (0-> P-1).

- **Grid* MyGrid**

  My own grid to keep track of atom positions.

Friends

- class **Residue**

4.44.1 Detailed Description

///

File: Molecule.h

This class manages molecules. A separate class called atom is used to represent the data for particular atoms.

///

Definition at line 32 of file molecule.h.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 Molecule::Molecule ()

Definition at line 3 of file molecule.C.

```
00004   { N = 0; A = 0; R = 0;MyGrid =0; }
00005
```

4.44.2.2 Molecule::Molecule (int *num_atoms*)

Constructs an atomic structure.

4.44.2.3 Molecule::Molecule (const Atom * *atoms*, int *num_pos*)

Definition at line 18 of file molecule.C.

```
00019   {
00020     N=0; A=0;
00021     Allocate(num_pos);
00022     for(int i=0; i<N; ++i)
00023         A[i] = atoms[i];
00024     R = 0;
00025
```

4.44.2.4 Molecule::Molecule (const jVector * *points_3d*, int *num_pos*)

Definition at line 6 of file molecule.C.

```
00006                                                         {
00007   N=0; A=0;
00008   Allocate(num_pos);
00009   for(int i=0; i<N; ++i)
00010     {
00011       //     A[i].position() = points_3d[i];
00012       MyGrid->setPosition(i, points_3d[i]);
00013       A[i].ID() = i;
00014     }
00015
```

4.44.2.5 Molecule::Molecule (const Molecule & *mol*)

Definition at line 27 of file molecule.C.

```
00028   {
00029     N=0; A=0;
00030     Copy(mol);
00031
```

4.44.2.6 Molecule::~Molecule ()

Definition at line 33 of file molecule.C.

```
00034   { Deallocate(); }
00035
```

4.44.3 Member Function Documentation

4.44.3.1 void Molecule::Allocate (int *n*) `[protected]`

Allocates space for n atoms.

Definition at line 36 of file molecule.C.

Referenced by Molecule(), and Reallocate().

```
00036                              {
00037   N = (n < 1) ? 0 : n;
```

```
A = 0;
if(N)
  {
    if(!(A = new Atom[N]))
      N = 0;
    else
      for(int i=0; i<N; ++i)
        A[i].ID() = i;
  }

```

4.44.3.2 void Molecule::Bounds (double & *minx*, double & *miny*, double & *minz*, \double & *maxx*, double & *maxy*, double & *maxz*) const

writes the upper and lower bounds in 3 dimensions by reference.

Definition at line 92 of file molecule.C.

Referenced by Envelope::Build().

```
                                                                        {

//  minx = miny  = minz = INF;
//  maxx = maxy  = maxz = (-1) * INF;
jVector temp;

temp = MyGrid->getPosition(0);
minx = maxx = temp[0];
miny = maxy = temp[1];
minz = maxz = temp[2];


for (int i = 1; i < N ; i++){
  // assumes 2 atoms
  temp = MyGrid->getPosition(i);

  if (temp[0] < minx)
    minx = temp[0];
  else
    if (temp[0] > maxx)
      maxx = temp[0];

  if (temp[1] < miny)
    miny = temp[1];
  else
    if (temp[1] > maxy)
      maxy = temp[1];

  if (temp[2] < minz)
    minz = temp[2];
  else
    if (temp[2] > maxz)
      maxz = temp[2];

}

```

4.44.3.3 void Molecule::Copy (const Molecule & *mol*) [protected]

Copies a molecule...

Definition at line 77 of file molecule.C.

Referenced by Molecule(), and operator=().

```
00077                                       {
00078   Reallocate(mol.N);
00079   for(int i=0; i<N; ++i){
00080     A[i] = mol.A[i];
00081     MyGrid->setPosition(i, mol.MyGrid->getPosition(i));
00082   }
00083   if (mol.R){ // if mol is using residues
00084     P = mol.P;
00085     R = new Residue[P];
00086     for(int i=0; i<P; ++i){
00087       R[i] = mol.R[i];
00088     }
00089   }
00090
```

4.44.3.4 void Molecule::Deallocate () [protected]

Deallocates the space.

Definition at line 51 of file molecule.C.

Referenced by Reallocate(), and ~Molecule().

```
00051                          {
00052   if (A) delete[] A;
00053   if (R) delete[] R;
00054
00055
00056   if (MyGrid) delete MyGrid;
00057   N = 0;
00058   A = 0;
00059   R = 0;
00060
```

4.44.3.5 istream & Molecule::DynRead (istream & *is*)

Definition at line 259 of file molecule.C.

```
00260   {
00261     DLList<jVector> L;
00262     jVector input_vec(3);
00263
00264     while(input_vec.dynread(is))
00265         L.AddToBack(input_vec);
00266
00267     Reallocate(L.Number());
00268
00269     for(int i=0; L; ++i)
00270         MyGrid->setPosition(i, L.RemoveFromFront());
00271
00272     return is;
00273
```

4.44.3.6 const IntVec & Molecule::GetBonds (int *atom*) const [inline]

returns list of bonds in an atom.

Definition at line 235 of file molecule.h.

```
00236 { return A[atom].Bonds;}
00237
```

4.44.3.7 int Molecule::Input (istream & *is*)

Definition at line 218 of file molecule.C.

```
00219   {
00220     int n=0;
00221     is >> n;
00222
00223     if(n<1)
00224       return 0;
00225
00226     Reallocate(n);
00227
00228     jVector tempPos;
00229     for(int i=0; i<n; ++i){
00230       is >> tempPos ;
00231       MyGrid->setPosition(i, tempPos);
00232     }
00233
```

4.44.3.8 int Molecule::InputBinary (istream & *is*)

Definition at line 246 of file molecule.C.

```
00247   {
00248     // Read in the number of atoms:
00249     is >= N;
00250     Reallocate(N);
00251     jVector tempPos;
00252     for(int i=0; i<N; ++i){
00253       is >= tempPos;
00254       MyGrid->setPosition(i,tempPos);
00255     }
00256
```

4.44.3.9 Molecule::MAKECOMPARISONS_NO_COMPARE (Molecule)

4.44.3.10 Molecule::MAKE_BIN_STREAM_OPERATORS (Molecule)

4.44.3.11 Molecule::MAKE_IO_STREAM_OPERATORS (Molecule)

4.44.3.12 double Molecule::MSD (int *a1*, int *a2*) const

return physical Minimum Separation Distance between two atoms.
Definition at line 154 of file molecule.C.

```
00154                                         {
00155   double d = 1;
00156
00157
```

```
  //  cerr << "msd: a1 " << a1 << "msd: a2 " << a2 << endl;
//   if (A[a1].Bonded(a2)){
//     d=  A[a1].cov_dist()+A[a2].cov_dist();
//     cerr << "BONDED: a1=" ;
//   } else {
//        d = A[a1].VDW()+A[a2].VDW();
//     //    cerr << "NONBONDED: a1=" ;
//   }
// //   cerr << A[a1].Label() << ", a2=" << A[a2].Label() << ", msd=" << d << endl;
// //   cerr << "atom " << a1 <<  " bonds: " << GetBonds(a1) << endl;
// //   cerr << "atom " << a2 <<  " bonds: " << GetBonds(a2) << endl;
// //   cerr << endl;


  int r1 = A[a1].residue_number;
  int r2 = A[a2].residue_number;


  if ((R[r1].N_chain_atom == a1) &&
      (R[r2].N_chain_atom == a2))
    {
          d=  2.6;
          //d= 2.7;
    }
  else if ((R[r1].CA_chain_atom == a1) &&
           (R[r2].CA_chain_atom == a2))
    {
          d = 3.6;
          //d= 3.7;
    }
  else if ((R[r1].O_chain_atom == a1) &&
           (R[r2].O_chain_atom == a2))
    {
      d = 2.8;
    }
  else if ((R[r1].C_chain_atom == a1) &&
           (R[r2].C_chain_atom == a2))
    {
          d = 2.8;
          //d= 2.9;
    }
  else if ((R[r1].CB_chain_atom == a1) &&
           (R[r2].CB_chain_atom == a2))
    {
      d = 3.5;
    }

```

4.44.3.13 int Molecule::NumRes () const [inline]

Reports the number of residues in the molecule.

Definition at line 199 of file molecule.h.

```
{ return P; }

```

4.44.3.14 int Molecule::Number () const [inline]

Reports the number of atoms in the molecule.

Definition at line 196 of file molecule.h.

Referenced by Envelope::Build(), ConstraintManager::ConstraintManager(), ConstraintManager::RandomPerturb(), ConstraintManager::TurnAtomOn(), ConstraintManager::TurnAtomsOff(), Envelope::WriteDensity(), ConstraintManager::clearConstraints(), ConstraintManager::getAssocAtoms(), ConstraintManager::getStressList(), ConstraintManager::numAtoms(), ConstraintManager::resetMoved(), ConstraintManager::setMolecule(), ConstraintManager::setPosition(), and ConstraintManager::stress().

```
00197 { return N; }
00198
```

4.44.3.15 int Molecule::Output (ostream & *os*) const

Definition at line 211 of file molecule.C.

```
00212   {
00213     os << N << '\n';
00214     for(int i=0; i<N; ++i)
00215         os << MyGrid->getPosition(i) << '\n';
00216     return N;
00217
```

4.44.3.16 int Molecule::OutputBinary (ostream & *os*) const

Definition at line 236 of file molecule.C.

```
00236                                           {
00237     // Write the number of atoms in the molecule:
00238     os <= N;
00239
00240     // Write out all the positions:
00241     for(int i=0; i<N; ++i)
00242         os <= MyGrid->getPosition(i);
00243     return N;
00244
```

4.44.3.17 void Molecule::Reallocate (int *n*) `[protected]`

Destroys the currently allocated space if n!=N and creates space for n atoms.

Definition at line 62 of file molecule.C.

Referenced by Copy(), DynRead(), Input(), InputBinary(), readFasta(), and readXmol().

```
00062                               {
00063   if(N == n)
00064     return;
00065   if (R) {
00066     WARN ("ACK! LOSING RESIDUE INFORMATION");
00067     cerr << "Reallocating from " << N << " atoms to " << n << " atoms." << endl;
00068   }
00069
```

4.44.3.18 double Molecule::VDW (int *i*) const [inline]

VDW for atom i'.

Definition at line 223 of file molecule.h.

Referenced by ConstraintManager::getRadius().

```
{ return A[i].VDW();}
```

4.44.3.19 DLList< int > Molecule::approx_neighbor_i (const jVector & *p*, double *distance*) const [inline]

grid interface to neighbor atoms (Approx, fast).

Definition at line 229 of file molecule.h.

Referenced by ConstraintManager::approx_neighbor_i().

```
{ return MyGrid->approx_neighbor_i(p,distance); }
```

4.44.3.20 const Atom & Molecule::getAtom (int *i*) const [inline]

access to an atom.

Definition at line 202 of file molecule.h.

Referenced by Envelope::Build().

```
{ return A[i]; }
```

4.44.3.21 jVector Molecule::getCOM () const

return center of mass.

Definition at line 141 of file molecule.C.

```
{
  jVector rv(3,0);

  for(int i=0; i<N; ++i){
    rv += MyGrid->getPosition(i);
  }

```

4.44.3.22 const Grid & Molecule::getGrid () const [inline]

talk to the grid (note const).

Definition at line 205 of file molecule.h.

```
{ return *MyGrid; }
```

4.44.3.23 const jVector & Molecule::getPosition (int *i*) const [inline]

returns position of atom i.

Definition at line 220 of file molecule.h.

Referenced by Output(), and OutputBinary().

```
00221 { return MyGrid->getPosition(i); }
00222
```

4.44.3.24 DLList< int > Molecule::neighbor_i (const jVector & *p*, double *distance*) const [inline]

grid interface to neighbor atoms (Exact, same as above, but checked).

Definition at line 232 of file molecule.h.

Referenced by ConstraintManager::neighbor_i().

```
00233 { return MyGrid->neighbor_i(p,distance); }
00234
```

4.44.3.25 const jVector & Molecule::operator() (int *i*) const [inline]

returns position of atom i.

Definition at line 217 of file molecule.h.

```
00218 { return MyGrid->getPosition(i); }
00219
```

4.44.3.26 const Molecule & Molecule::operator= (const Molecule & *mol*)

Assignment (calls the **Copy**() (p. 361) function).

Definition at line 138 of file molecule.C.

```
00139   { Copy(mol); return *this; }
00140
```

4.44.3.27 Atom & Molecule::operator[] (int *i*) [inline]

Set an atom in the molecule by the atom id. Note, the id and index of an atom are the same.

Definition at line 214 of file molecule.h.

```
00215 { return A[i]; }
00216
```

4.44.3.28 const Atom & Molecule::operator[] (int *i*) const [inline]

Access an atom in the molecule by the atom id. Note, the id and index of an atom are the same.

Definition at line 211 of file molecule.h.

```
00212 { return A[i]; }
00213
```

4.44.3.29 int Molecule::readFasta (istream & *is*)

Reads in a molecule from an xyz formatted file. Returns the number of atoms read in.

Definition at line 302 of file molecule.C.

Referenced by main().

```
00303 {
00304   int res_count = 0;
00305   int tot_residues = 0;
00306   int atom_count = 0;
00307   int res_type_temp = UNINITIALIZED_RES_ID;
00308   DLList<DynString *> ResList;
00309   DLList<DynString *>::ConstIterator it;
00310   DynString * DynPtr;
00311
00312   DynString str,previous; // strings
00313   // ASSUMES CODE_1 SERIES
00314   previous.ReadLine(is); // read comment line
00315   label = previous;
00316
00317
00318   while(is.good()){
00319     str.Clear();
00320     str.GetChar(is);
00321     str.ToUpper();
00322     res_type_temp = ResidueRegistry::CheckType_code_1(str);
00323     if (res_type_temp > -1) {// its a real residue code
00324       res_count++;
00325       atom_count += ResidueRegistry::numAtoms[res_type_temp];
00326       DynPtr = new DynString(str);
00327       ResList.Add(DynPtr);
00328     }
00329   }
00330
00331   // set up molecule
00332   Reallocate(atom_count);
00333   P = res_count;
00334   R = new Residue[P];
00335
00336   tot_residues = res_count;
00337   res_count = 0;
00338   atom_count = 0;
00339
00340   // iterate over list, create molecule
00341   for(it = ResList.GetConstIterator(); it; ++it){
00342     DynPtr = ($\ast$it);
00343     R[res_count].ID() = res_count;
00344     ResidueRegistry::SetType_code_1(R[res_count], *DynPtr);
00345     if (tot_residues == 1)
00346       R[res_count].Setup(*this, atom_count,0,0); // this configures atoms
00347     else if (res_count == 0)
00348       atom_count = R[res_count].Setup(*this, atom_count,0,&(R[1]));
00349     else if (res_count == tot_residues-1)
00350       atom_count = R[res_count].Setup(*this, atom_count, \
00351                                        &(R[tot_residues-2]),0);
00352     else
00353       // this configures atoms
00354       atom_count = R[res_count].Setup \
```

```
      (*this, atom_count,&(R[res_count-1]),&(R[res_count+1]));

    res_count++;
  }

  // delete the DynStrings
  while (ResList.Number()){
    DynPtr = ResList.Remove();
    delete DynPtr;
  }

```

4.44.3.30 int Molecule::readXmol (istream & *is*)

Reads in a molecule from an xyz formatted file. Returns the number of atoms read in.

Definition at line 373 of file molecule.C.

Referenced by main(), and ConstraintManager::readNewMoleculeXmol().

```
{

  DynString str,previous; // strings
  jVector tempPos(3); // 3 element vector
  char cJunk[256];


  previous.ReadLine(is); // read num atoms
  Reallocate(previous.ToInt()); // make me a molecule with this many atoms available

  str.ReadLine(is); // read comment
  label = str;

    //    cerr << "Read: " << label << endl;

    // Read in the atoms (the number of atoms is specified on
    // the line before the -- line:
    //

  for(int i=0; i<N; i++)
    {
      if(!is.good())
        {
          cerr << "Warning: read in " << i << ", not "
               << N << " atoms.\n";
          DIE("this is a fatal error; check file");
        }

      //        is >> A[i].Type() >> tempPos ;
      // label removed from front -- now at the end
      is >> str >> tempPos ;
      is.getline(cJunk,256); //scrap rest -o- line!
      MyGrid->setPosition(i, tempPos);
      //      cerr << "setting type on atom " << i << " to " << str << endl;
      AtomRegistry::SetType(A[i],str, str);
    }

```

4.44.3.31 void Molecule::setAtom (int *i*, const Atom & *a*) [inline]

make atom a the i'th atom.

Definition at line 208 of file molecule.h.

```
00209 { A[i] = a; }
00210
```

4.44.3.32 void Molecule::setPosition (int *i*, const jVector & *newPos*) [inline]

sets position of atom i.

Definition at line 226 of file molecule.h.

Referenced by ConstraintManager::setPosition().

```
00227 { MyGrid->setPosition(i, newPos); }
00228
```

4.44.3.33 int Molecule::setRandom (void)

Generates random positions for Num' atoms.

Definition at line 275 of file molecule.C.

Referenced by main().

```
00275                         {
00276   // Generates random positions for Num' atoms
00277   //  RanInt randomPosition((-1)*Num/3, Num/3);
00278
00279   RanInt randomPosition(-50, 50);
00280   // actually want double, but int is reproducible across systems
00281   jVector tempPos(3); // 3 element vector
00282
00283   ofstream out("random_start.xmol");
00284   out << N << endl;
00285   out << "Random Start: "<< label << endl;
00286
00287   for(int i=0; i<N; i++)
00288     {
00289       tempPos[0] = randomPosition.Get();
00290       tempPos[1] = randomPosition.Get();
00291       tempPos[2] = randomPosition.Get();
00292
00293       MyGrid->setPosition(i, tempPos);
00294
00295       out << A[i].type << tempPos << "\t  9" << endl;
00296     }
00297
```

4.44.3.34 void Molecule::skipXmol (istream & *is*) [static]

skip a frame in a xmol format file stream.

Definition at line 444 of file molecule.C.

Referenced by jdErr_Eval::run().

```
00445 {
00446
```

```
  DynString previous; // strings
  char cJunk[256];


  previous.ReadLine(is); // read num atoms
  int lines = previous.ToInt();
  is.getline(cJunk,256); // pull \n

  is.getline(cJunk,256); //scrap comment

  for(int i=0; i<lines; i++)
    {
      if(!is.good())
        {
          cerr << "Warning: skipXmol: read in " << i << ", not "
               << lines << " atoms.\n";
          DIE("this is a fatal error; check file");
        }

      is.getline(cJunk,256); //scrap each line

    }

```

4.44.3.35 void Molecule::writeXmol (ostream & *os*) const

Writes a file in xmol xyz format. This version puts the atom int atom number as the first Xmol field.

Definition at line 413 of file molecule.C.

Referenced by main().

```
00413                                              {
00414   os << N << '\n';
00415   os << label << "\n";
00416   for(int i=0; i<N; ++i)
00417     os  << A[i].type << " " << MyGrid->getPosition(i) << " 9"  << endl;
00418
```

4.44.3.36 void Molecule::writeXmolData (ostream & *os*, double ∗ *data*, DynString *new_comment*) const

Writes a client specified double for each atom in the output Xmol formatted file.

Definition at line 432 of file molecule.C.

```
00433 {
00434   // Warning. using wrong data array can cause seg fault, cf above.
00435
00436   os << N << '\n';
00437   os << new_comment << "\n";
00438   for(int i=0; i<N; ++i)
00439     os << A[i].type  <<" "<< MyGrid->getPosition(i) << ' ' << data[i] << '\n';
00440
00441
```

4.44.3.37 void Molecule::writeXmolData (ostream & *os*, int ∗ *data*, DynString *new_comment*) const

Writes a client specified integer for each atom in the output Xmol formatted file.

Definition at line 420 of file molecule.C.

```
00420                                                                          {
00421   // Warning.  This assumes that data points to at least N integers
00422   // I'm not sure a good way to make this safer. -jd
00423
00424   os << N << '\n';
00425   os << new_comment << "\n";
00426   for(int i=0; i<N; ++i)
00427     os << A[i].type <<" "<< MyGrid->getPosition(i) << ' ' << data[i] << endl;
00428
00429
```

4.44.4 Friends And Related Function Documentation

4.44.4.1 class Residue [friend]

Definition at line 35 of file molecule.h.

4.44.5 Member Data Documentation

4.44.5.1 Atom ∗ Molecule::A [protected]

A is a vector of atoms in the molecule (0-> N-1).

Definition at line 170 of file molecule.h.

4.44.5.2 Grid ∗ Molecule::MyGrid [protected]

My own grid to keep track of atom positions.

Definition at line 176 of file molecule.h.

4.44.5.3 int Molecule::N [protected]

The number of atoms in the structure.

Definition at line 164 of file molecule.h.

4.44.5.4 int Molecule::P [protected]

The number of residues in the structure.

Definition at line 167 of file molecule.h.

4.44.5.5 Residue ∗ Molecule::R [protected]

R is a vector of residues in the molecule (0-> P-1).

Definition at line 173 of file molecule.h.

4.44.5.6 double Molecule::grid_delta = 3.7 [static]

expected inter-atomic width.

Definition at line 78 of file globals.C.

4.44.5.7 double Molecule::grid_max_radius = 250 [static]

expected max molecule size.

Definition at line 79 of file globals.C.

4.44.5.8 DynString Molecule::label [protected]

label for the molecule.

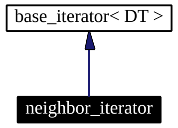
Definition at line 161 of file molecule.h.

The documentation for this class was generated from the following files:

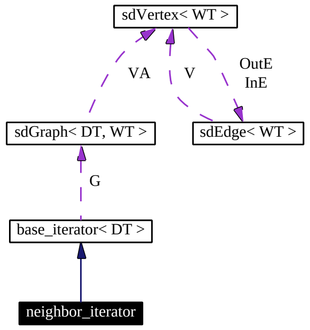
- **molecule.h**
- **globals.C**
- **molecule.C**

4.45 neighbor_iterator Class Reference

Inheritance diagram for neighbor_iterator



Collaboration diagram for neighbor_iterator:



Public Methods

- **neighbor_iterator** (int reverse=0)
- **neighbor_iterator** (const neighbor_iterator &ni)
- ~**neighbor_iterator** ()
- void **reset** ()
- **operator int** () const
- const **base_iterator**& **operator++** ()
- int **operator** ∗ () const

Protected Methods

- **neighbor_iterator** (**sdGraph**< DT, WT > ∗g, int vndx, int reverse=0)

Friends

- class **sdGraph< DT, WT >**

4.45.1 Constructor & Destructor Documentation

4.45.1.1 neighbor_iterator::neighbor_iterator (sdGraph< DT, WT > ∗ *g*, int *vndx*, int *reverse* = 0) `[inline, protected]`

Definition at line 58 of file sdgiter.inc.

```
00059                  : base_iterator(g,vndx,reverse) { reset(); }
00060
```

4.45.1.2 neighbor_iterator::neighbor_iterator (int *reverse* = 0) [inline]

Definition at line 61 of file sdgiter.inc.

```
00062                                   : base_iterator<DT>(reverse) { reset(); }
00063
```

4.45.1.3 neighbor_iterator::neighbor_iterator (const neighbor_iterator & *ni*) [inline]

Definition at line 65 of file sdgiter.inc.

```
00066                                   : base_iterator<DT>(ni) {}
00067
```

4.45.1.4 neighbor_iterator::∼neighbor_iterator () [inline]

Definition at line 70 of file sdgiter.inc.

```
00070                                      { /* empty */ }
00071
```

4.45.2 Member Function Documentation

4.45.2.1 int neighbor_iterator::operator * () const [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 95 of file sdgiter.inc.

```
00096                   {
00097                     if((vid==-1) || (vstart==-1)) return -1;
00098                     if(!rev) { if(vid>=G->VA[vstart]->NumO) return -1; }
00099
```

4.45.2.2 neighbor_iterator::operator int () const [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 82 of file sdgiter.inc.

```
00083                   {
00084                     if(vid == -1) return 0;
00085                     if(!rev) return (vid < G->VA[vstart]->NumO);
00086                     return (vid < G->VA[vstart]->NumI);
00087
```

4.45.2.3 const base_iterator & neighbor_iterator::operator++ () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 89 of file sdgiter.inc.

```
{
  if(vid != -1) ++vid;
  return *this;

```

4.45.2.4 void neighbor_iterator::reset () [inline, virtual]

Reimplemented from **base_iterator** (p. 65).

Definition at line 72 of file sdgiter.inc.

```
{
  if(bad()) vid = -1;
  else
    {
      if(!rev) vid = (G->VA[vstart]->NumO > 0) ? 0 : -1;
      else vid = (G->VA[vstart]->NumI > 0) ? 0 : -1;
    }

```

4.45.3 Friends And Related Function Documentation

4.45.3.1 class sdGraph [friend]

Reimplemented from **base_iterator** (p. 66).

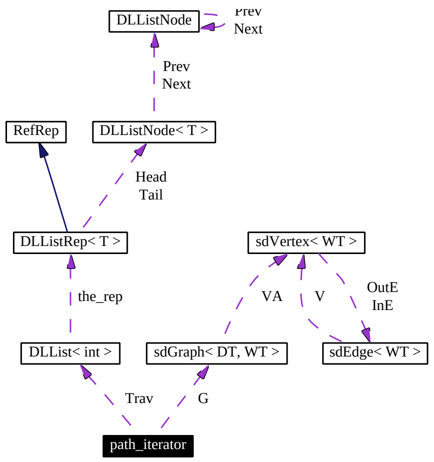
Definition at line 55 of file sdgiter.inc.

The documentation for this class was generated from the following file:

- **sdgiter.inc**

4.46 path_iterator Class Reference

Collaboration diagram for path_iterator:



Public Methods

- int **bad** () const
- int **source** () const
- int **target** () const
- int **length** () const
- void **copy** (const path_iterator &**pi**)
- const path_iterator& **operator=** (const path_iterator &**pi**)
- **path_iterator** (path_iterator &**pi**)
- **path_iterator** ()
- ∼**path_iterator** ()
- const path_iterator& **operator++** ()
- **operator int** () const
- int∗ **operator** ∗ () const
- void **reset** ()

Private Methods

- **path_iterator** (const **sdGraph**< DT, WT > ∗sdg, int s, int t, int k, **DLList**< int > ∗const ∗tl, int(∗filter)(const **sdGraph**< DT, WT > &graph, const int ∗path, int ndx, int pathlen, void ∗arbitrary_criteria), void ∗criteria)

Private Attributes

- const **sdGraph**<DT,WT>∗ **G**
- int **K**
- int∗ **Path**
- **DLList**<int>::**Iterator**∗∗ **Trav**
- int **D**
- int **Done**

- int **AlmostDone**
- int (∗ **Filter**)(const **sdGraph**< DT, WT > &,const int ∗,int, int, void ∗)
- void∗ **Criteria**

Friends

- class **sdGraph< DT, WT >**

4.46.1 Constructor & Destructor Documentation

4.46.1.1 path_iterator::path_iterator (path_iterator & *pi*) [inline]

Definition at line 405 of file sdgiter.inc.

```
00406                 {
00407                   G = 0; K = -1; D = -1;
00408                   Done = 1; AlmostDone = 1;
00409                   Path = 0; Trav = 0;
00410                   Filter = 0;
00411                   Criteria = 0;
00412                   copy(pi);
00413
```

4.46.1.2 path_iterator::path_iterator () [inline]

Definition at line 415 of file sdgiter.inc.

```
00416                 {
00417                   G = 0; K = -1; D = -1;
00418                   Done = 1; AlmostDone = 1;
00419                   Path = 0; Trav = 0; Path = 0;
00420                   Filter = 0;
00421                   Criteria = 0;
00422
```

4.46.1.3 path_iterator::∼path_iterator () [inline]

Definition at line 424 of file sdgiter.inc.

```
00425                 {
00426                   delete[] Path;
00427                   for(int i=0; i<K-1; ++i) delete[] Trav[i];
00428                   delete[] Trav;
00429
```

4.46.1.4 path_iterator::path_iterator (const sdGraph< DT, WT > ∗ *sdg*, int *s*, int *t*, int *k*, DLList< int > ∗const ∗ *tl*, int(∗ *filter*)(const sdGraph< DT, WT > &graph, const int ∗path, int ndx, int pathlen, void ∗arbitrary_criteria), void ∗ *criteria*) [inline, private]

Definition at line 562 of file sdgiter.inc.

```
        {
          int n = sdg->NumV;
          G = 0; K = -1; D = -1; AlmostDone = Done = 1;
          Path = 0; Trav = 0;

          if(k<0 || n<1 || s<0 || s>=n || t<0 || t>=n) return;

          if(!tl && k>1) return;

          if(!(Path = new int[k+1])) { return; }

          if(k>1)
            {
              if(!(Trav = new DLList<int>::Iterator*[k-1]))
                { delete[] Path; return; }

              for(int i=0; i<k-1; ++i)
                {
                  if(!(Trav[i] = new DLList<int>::Iterator[n]))
                    {
                      for(int j=0; j<=i; ++j)
                        { delete[] Trav[j]; }
                      delete[] Path; delete[] Trav;
                      return;
                    }
                  for(int j=0; j<n; ++j)
                      Trav[i][j] = tl[i][j].GetIterator();
                }
            }

```

4.46.2 Member Function Documentation

4.46.2.1 int path_iterator::bad () const [inline]

Definition at line 328 of file sdgiter.inc.

Referenced by copy().

```
        { return (!G || !Path || (K>1 && !Trav)); }

```

4.46.2.2 void path_iterator::copy (const path_iterator & *pi*) [inline]

Definition at line 335 of file sdgiter.inc.

Referenced by operator=(), and path_iterator().

```
        {
          if(this == &pi) return;

          if(pi.bad())
            // If "pi" is bad then we make this iterator bad (copy
            // a bad iterator to a good iterator makes the good
            // iterator bad!).
            {
              for(int i=0; i<K-1; ++i) delete[] Trav[i];
              delete[] Trav; delete[] Path;
              G=0; K=0; D=-1; Done=1;
              return;
            }
```

```

          DLList<int>::Iterator ** tmpT = 0;
          int * tmpP = 0;

          if(G->NumV != pi.G->NumV || K!=pi.K)
            // Have to resize our lists.
            {
              tmpT = Trav; tmpP = Path;

              if(!(Path = new int[pi.K+1]))
                  { Path = tmpP; return; }

              if(pi.K>1)
                {
                  if(!(Trav = new DLList<int>::Iterator*[pi.K-1]))
                    {
                      delete[] Path; delete[] Trav;
                      Path = tmpP; Trav = tmpT;
                      return;
                    }
                  for(int i=0; i<pi.K-1; ++i)
                    {
             if(!(Trav[i] = new DLList<int>::Iterator[pi.G->NumV]))
                      {
                        for(int j=0; j<i; ++j) delete[] Trav[j];
                        delete[] Path; delete[] Trav;
                        Path = tmpP; Trav = tmpT;
                        return;
                      }
                    }
                }
              else Trav = 0;

              if(K>1) for(int i=0; i<K-1; ++i) delete[] tmpT[i];
              delete[] tmpP; delete[] tmpT;
              G = pi.G; K = pi.K;
            }

          for(int i=0; i<K-1; ++i)
            {
              for(int j=0; j<G->NumV; ++j)
                  Trav[i][j] = pi.Trav[i][j];
              Path[i+1] = pi.Path[i+1];
            }

```

4.46.2.3 int path_iterator::length () const [inline]

Definition at line 333 of file sdgiter.inc.

```
00333                                  { return K; }
00334
```

4.46.2.4 int * path_iterator::operator * () const [inline]

Definition at line 536 of file sdgiter.inc.

```
        {
          int * current_path;
          if(!(current_path = new int[K+1])) return 0;
```

```
00540                     for(int i=0; i<K+1; ++i) current_path[i] = Path[i];
00541                     return current_path;
00542
```

4.46.2.5 path_iterator::operator int () const [inline]

Definition at line 534 of file sdgiter.inc.

```
00534                                       { return !Done; }
00535
```

4.46.2.6 const path_iterator & path_iterator::operator++ () [inline]

Definition at line 431 of file sdgiter.inc.

```
00432                   {
00433
00434                     int found, pre;
00435
00436                     if(bad()) AlmostDone = 1;
00437                     if(AlmostDone) Done = 1;
00438                     if(Done) return *this;
00439
00440                     if(K==0)
00441                       {
00442 #ifdef DEBUG
00443   ++PathsVisited;
00444 #endif
00445                         AlmostDone = 1;
00446                         if(D != Path[0]) Done = 1;
00447                         return *this;
00448                       }
00449
00450                     if(K==1)
00451                       {
00452 #ifdef DEBUG
00453   ++PathsVisited;
00454 #endif
00455                         if(G->HasEdge(Path[0],Path[1]) &&
00456                           ((!Filter || !Filter(*G,Path,0,1,Criteria))))
00457                             AlmostDone=1;
00458                         else { AlmostDone = Done = 1; }
00459                         return *this;
00460                       }
00461
00462                     // Backward stage of the iteration: build up a path
00463                     // from the end to the beginning.
00464                     //
00465                     for(; D>=0; --D)
00466                       {
00467                         pre = Path[D+2];
00468
00469                         if(!Trav[D][pre])
00470                           // If this happens it means no path exists.
00471                           { AlmostDone = Done = 1; return *this; }
00472
00473                         for(found = 0; Trav[D][pre]; ++(Trav[D][pre]))
00474                           {
00475                             if(found) break;
00476
```

```
                            Path[D+1] = *(Trav[D][pre]);

#ifdef DEBUG
  ++PathsVisited;
#endif

                            // Apply the filter to this path.  If the
                            // path is to be filtered, we assume all
                            // paths containing the current path as a
                            // subset should not be expanded.  If this
                            // path should be filtered, we simply skip
                            // this and look for a different path.
                            //
                            if(!Filter || !Filter(*G,Path,D+1,K,Criteria))
                                found = 1;
                          }

                        if(!found)
                          // All the sub-paths got filtered so we need to back
                          // up to find another candidate sub-path.
                          {
                            for(; D<K-1; ++D)
                              {
                                pre = Path[D+2];
                                if(!Trav[D][pre]) Trav[D][pre].reset();
                                else break;
                              }

                            // If we're at the beginning, then we must be
                            // done.
                            if(D==K-1)
                              { AlmostDone = Done = 1; return *this; }
                            else ++D;
                          }
                      }

                    // Forward stage of the iteration: prepare the matrices
                    // for the next iteration.  We reset all the matrices/sets
                    // that are *completely* empty.  As soon as we reach a set
                    // that contains elements in it we stop.  This is the point
                    // where the next search will pick up.

                    for(D=0; D<K-1; ++D)
                      {
                        pre = Path[D+2];
                        if(!Trav[D][pre]) Trav[D][pre].reset();
                        else break;
                      }

```

4.46.2.7 const path_iterator & path_iterator::operator= (const path_iterator & *pi*) [inline]

Definition at line 402 of file sdgiter.inc.

```
                  { copy(pi); return *this; }

```

4.46.2.8 void path_iterator::reset () [inline]

Definition at line 544 of file sdgiter.inc.

```
00545                 {
00546                   if(K>1)
00547                     {
00548                       for(; D<K-1; ++D)
00549                           for(int i=0; i<G->NumV; ++i)
00550                               Trav[D][i].reset();
00551                       D = K-2;
00552                     }
00553
```

4.46.2.9 int path_iterator::source () const [inline]

Definition at line 331 of file sdgiter.inc.

```
00331                                   { return Path ? Path[0] : -1; }
00332
```

4.46.2.10 int path_iterator::target () const [inline]

Definition at line 332 of file sdgiter.inc.

```
00332                                   { return Path ? Path[K] : -1; }
00333
```

4.46.3 Friends And Related Function Documentation

4.46.3.1 class sdGraph [friend]

Definition at line 326 of file sdgiter.inc.

4.46.4 Member Data Documentation

4.46.4.1 int path_iterator::AlmostDone [private]

Definition at line 627 of file sdgiter.inc.

4.46.4.2 void * path_iterator::Criteria [private]

Definition at line 633 of file sdgiter.inc.

4.46.4.3 int path_iterator::D [private]

Definition at line 624 of file sdgiter.inc.

4.46.4.4 int path_iterator::Done [private]

Definition at line 625 of file sdgiter.inc.

4.46.4.5 int (* path_iterator::Filter)(const sdGraph<DT,WT>&,const int*,int,int,void*) [private]

Referenced by operator++().

4.46.4.6 const sdGraph< DT, WT > * path_iterator::G [private]

Definition at line 618 of file sdgiter.inc.

4.46.4.7 int path_iterator::K [private]

Definition at line 619 of file sdgiter.inc.

4.46.4.8 int * path_iterator::Path [private]

Definition at line 620 of file sdgiter.inc.

4.46.4.9 DLList< int >::Iterator ** path_iterator::Trav [private]

Definition at line 621 of file sdgiter.inc.

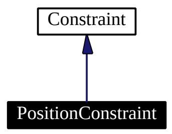
The documentation for this class was generated from the following file:

- **sdgiter.inc**

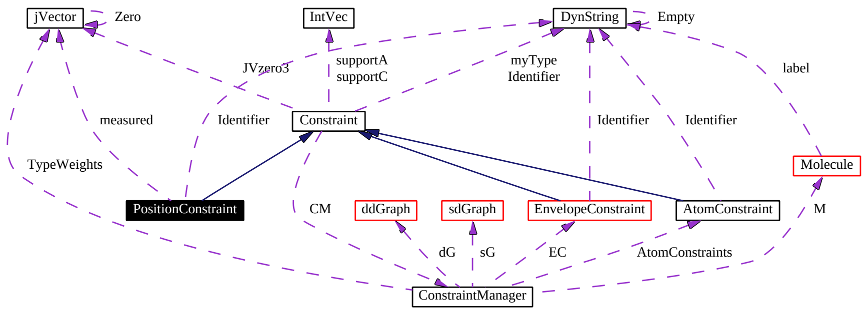
4.47 PositionConstraint Class Reference

`#include <position_constraint.h>`

Inheritance diagram for PositionConstraint



Collaboration diagram for PositionConstraint:



Public Methods

- **PositionConstraint** (**ConstraintManager** &cm)
- **PositionConstraint** (**ConstraintManager** &cm, int atom_i, double x, double y, double **z**, double variance_in)
- **~PositionConstraint** ()
- **jVector dStress** (int k, const **jVector** &pos) const
- int **Compare** (const PositionConstraint &dc) const
- **MAKECOMPARISONS_WITH_COMPARE** (PositionConstraint)
- int **Output** (ostream &os) const
- int **OutputBinary** (ostream &os) const

Static Public Attributes

- **DynString Identifier** = "atompos:"

Protected Methods

- double **computeStress** () const
- double **computeStress** (int k, const **jVector** &pos) const

  Mimics moving the atom to a new position.

- int **Read** (istream &is)
- int **ReadBinary** (istream &is)

  Same scenerio as **Read**() (p. 388) *only binary version.*

- void **PreSetup** ()

Protected Attributes

- **jVector measured**

  input, measured position (the correct position).

- double **variance**
- double **stdev**

4.47.1 Detailed Description

//

position_constraint.h

This is a constraint type for GRAPE. It constrains the position of a single atom to set coordinates, each with a specified variance.

//

Definition at line 20 of file position_constraint.h.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 PositionConstraint::PositionConstraint (ConstraintManager & *cm*)

Definition at line 3 of file position_constraint.C.

```
  : Constraint(cm,IntVec(1,-1))
  {
    // this is an unititialized constraint.  The atom position
    // and the variance values must be added.
    measured[0]=0;
    measured[1]=0;
    measured[2]=0;
    //    variance[0] = INF;
    //    variance[1] = INF;
    //    variance[2] = INF;
    variance = INF;

```

4.47.2.2 PositionConstraint::PositionConstraint (ConstraintManager & *cm*, int *atom_i*, double *x*, double *y*, double *z*, double *variance_in*)

Definition at line 20 of file position_constraint.C.

```
  : Constraint(cm, IntVec(1,atom_i))

{
  measured[0]=x;
  measured[1]=y;
  measured[2]=z;
  //  variance[0] = x_var;
```

```
00030   //variance[1] = y_var;
00031   //variance[2] = z_var;
00032   variance = variance_in;
00033   //  stdev[0] = sqrt(x_var);
00034   //  stdev[1] = sqrt(y_var);
00035   //  stdev[2] = sqrt(z_var);
00036   stdev = sqrt(variance);
00037
```

4.47.2.3 PositionConstraint::~PositionConstraint () [inline]

Definition at line 34 of file position_constraint.h.

```
00034                         {}
00035
```

4.47.3 Member Function Documentation

4.47.3.1 int PositionConstraint::Compare (const PositionConstraint & *pc*) const [inline]

Definition at line 72 of file position_constraint.h.

```
00073 {
00074   if(supportA[0] == pc.supportA[0])
00075     return 0;
00076   if(supportA[0]<pc.supportA[0])
00077     return -1;
00078   return 1;
00079
```

4.47.3.2 PositionConstraint::MAKECOMPARISONS_WITH_COMPARE (PositionConstraint)

4.47.3.3 int PositionConstraint::Output (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 137 of file position_constraint.C.

```
00138 {
00139   os << supportA[0] << ' ' << measured[0] << ' '<< measured[1] << ' '
00140      << measured[2] << ' ' << variance;
00141   return 2;
00142
```

4.47.3.4 int PositionConstraint::OutputBinary (ostream & *os*) const [virtual]

Reimplemented from **Constraint** (p. 84).

Definition at line 151 of file position_constraint.C.

```
00152 {
00153   os <= supportA <= measured <= variance;
00154   return 2;
00155
```

4.47.3.5 void PositionConstraint::PreSetup (void) `[protected, virtual]`

The setup function is called if the uninialized constraint method is used to create the constraint to allow different data representations within the constraint. Without this, only data read in from the input file will be available to the member functions. For simple constraints, this is really not very useful. However for multi-atom constraints that need to be a little 'smarter' about data representations, the Setup function will be helpful.

Reimplemented from **Constraint** (p. 85).

Definition at line 39 of file position_constraint.C.

```
00040 {
00041   //  stdev[0] = sqrt(variance[0]);
00042   //  stdev[1] = sqrt(variance[1]);
00043   //  stdev[2] = sqrt(variance[2]);
00044   stdev = sqrt(variance);
00045
```

4.47.3.6 int PositionConstraint::Read (istream & *is*) `[protected, virtual]`

Derived classes need to define here how to read themselves in. Notice that the base class defines **Input**() (p. 83) as a wrapper around this virtual function. Subclasses should typically overwrite just the **Read**() (p. 388) function (and not the **Input**() (p. 83) function).

In the **Read**() (p. 388) function, the derived class should *not* log itself with the constraint manager since this is taken care of in the base classes **Input**() (p. 83) function.

Reimplemented from **Constraint** (p. 85).

Definition at line 144 of file position_constraint.C.

```
00145 {
00146   is >> supportA[0] >> measured[0] >>  measured[1]
00147      >> measured[2] >> variance;
00148   return 2;
00149
```

4.47.3.7 int PositionConstraint::ReadBinary (istream & *is*) `[protected, virtual]`

Same scenerio as **Read**() (p. 388) only binary version.

Reimplemented from **Constraint** (p. 85).

Definition at line 157 of file position_constraint.C.

```
00158 {
00159   is >= supportA[0] >= measured >= variance;
00160   return 2;
00161
```

4.47.3.8 double PositionConstraint::computeStress (int *k*, const jVector & *pos_k*) const `[protected, virtual]`

Mimics moving the atom to a new position.

Reimplemented from **Constraint** (p. 89).

Definition at line 70 of file position_constraint.C.

```
00071 {
00072
00073   if(k != supportA[0])
00074     {
00075       cerr << "Fatal error: position_constraint.C, computeStress("<< k \
00076            <<", pos) not legal on position constraint for atom " \
00077            << supportA[0] << endl;
00078       DIE("");
00079     }
00080
```

4.47.3.9 double PositionConstraint::computeStress () const [protected, virtual]

Returns how similar the distance between atoms i and j in molecule, mol, are compared to the measured distance represented in this constraint.

Reimplemented from **Constraint** (p. 89).

Definition at line 52 of file position_constraint.C.

```
00053 {
00054
00055
00056   // Find the current position in the predicted molecule:
00057   jVector diff = getMolecule()(supportA[0])-measured;
00058
00059   // Return the squared difference between this and our model position:
00060   // double calc = (diff[0]*diff[0]/variance[0])+
00061   // (diff[1]*diff[1]/variance[1])+
00062   // (diff[2]*diff[2]/variance[2]);
00063   double calc = (diff[0]*diff[0]+ \
00064                 (diff[1]*diff[1])+\
00065                 (diff[2]*diff[2]));
00066
00067   return calc/(2.0*variance);
00068
```

4.47.3.10 jVector PositionConstraint::dStress (int *k*, const jVector & *pos_k*) const [virtual]

asking constraint number k for its 12 element dStress vector we pass in pos_a vector – corresponding to the atom position if the constraint is an atom, and corresponding to the position of the atom k in question if the call is to a non-atom constraint. EG for a distance constraint atom 12 needs the dStress from the d-constraint, so it will call dStress on the d-constraint(12, atom_pos) so the d_constr can answer the question if atom 12 were in atom_pos, I would say my dStress is blah.

Reimplemented from **Constraint** (p. 89).

Definition at line 95 of file position_constraint.C.

```
00096 {
00097   // A HACK and a TEMPORARY FIX to an inherent logic problems in this
00098   // set of functions.  come see me w/ questions -jd
00099
00100
00101   if(k != supportA[0])
00102     {
00103       cerr << "Fatal error: position_constraint.C, dStress("<< k \
00104            <<", pos) not legal on position constraint for atom " <<
00105         supportA[0] << endl;
```

```
      DIE("");
    }

```

4.47.4 Member Data Documentation

4.47.4.1 DynString PositionConstraint::Identifier = "atompos:" [static]

Reimplemented from **Constraint** (p. 95).

Definition at line 118 of file globals.C.

4.47.4.2 jVector PositionConstraint::measured [protected]

input, measured position (the correct position).

Definition at line 59 of file position_constraint.h.

4.47.4.3 double PositionConstraint::stdev [protected]

Definition at line 62 of file position_constraint.h.

4.47.4.4 double PositionConstraint::variance [protected]

Definition at line 61 of file position_constraint.h.

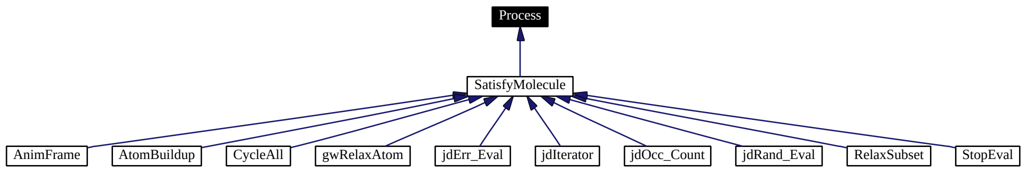
The documentation for this class was generated from the following files:

- **position_constraint.h**
- **globals.C**
- **position_constraint.C**

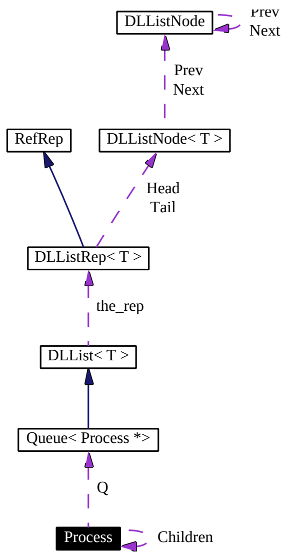
4.48 Process Class Reference

`#include <process.h>`

Inheritance diagram for Process



Collaboration diagram for Process:



Public Methods

- virtual **~Process** ()
- virtual int **run** ()=0
- virtual void **reset** ()

  Run any required operations to reset the process.

- void **runChildren** ()
- void **runChildrenRepeatFirst** ()
- int **runChild** (int i)
- int **runNextChild** ()
- void **addChild** (Process ∗c=NULL)
- int **numChildren** () const
- int **numQueued** () const

  Returns the number of children awaiting execution.

- void **cleanQueue** ()

Removes all queued jobs without run' ing them.

- void **setPriority** (double pri)

  Sets the priority of the process (low priorities are executed first.

- double **getPriority** () const

  Returns the priority of this process.

- int **Compare** (const Process &p) const

  Compares processes based on their priority.

- **MAKECOMPARISONS_WITH_COMPARE** (Process)

  Give me all the comparison operators.

Protected Methods

- **Process** (double pri=-INF, int numChildren=0)

Protected Attributes

- double **Priority**

  High priorities are executed first. These can range from -INF to +INF.

- Process** **Children**

  An array of pointers to sub-processes.

- int **numC**

  The number of total children (including zombies).

- bool **active**

  defaults to t, if f, the process is never run.

Private Methods

- void **assertOrder** ()

Private Attributes

- int **sorted**
- **Queue**<Process *> **Q**

4.48.1 Detailed Description

//

File: Process.h

Abstract class to manage and provide functionality for a set of 'jobs.' Each job inherits from class Process and can create children, add them to a queue and execute chilren on that queue. All jobs must have a member called **run**(void) (p. 396).

//

Definition at line 24 of file process.h.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 Process::~Process () `[virtual]`

Definition at line 18 of file process.C.

```
{
  if(Children)
    {
      for(int i=0; i<numC; ++i)
        delete Children[i];
      delete[] Children;
      Children=0;
    }

```

4.48.2.2 Process::Process (double *pri* = -INF, int *numChildren* = 0) `[protected]`

Derived classes can construct a process with a specified priority and a specified number of children. These Children are ∗not∗ added to the queue.

Definition at line 4 of file process.C.

```
{
  Children=0;
  numC = 0;
  sorted = 0;

  if(numChildren>0 && (Children = new Process*[numChildren]))
    {
      for(int i=0; i<numChildren; i++)
        Children[i] = 0;
      numC = numChildren;
    }

```

4.48.3 Member Function Documentation

4.48.3.1 int Process::Compare (const Process & *p*) const `[inline]`

Compares processes based on their priority.

Definition at line 99 of file process.h.

```
00101   {
00102     if(Priority<p.Priority)
00103       return -1;
00104     else if(Priority>p.Priority)
00105       return 1;
00106     return 0;
00107
```

4.48.3.2 Process::MAKECOMPARISONS_WITH_COMPARE (Process)

Give me all the comparison operators.

4.48.3.3 void Process::addChild (Process * *p* = NULL)

Adds a child to the execution queue. Note that processes do not need to queue children from the Children array. For example, a process may add itself to the execution queue for recursive purposes (via addChild(this)).

Definition at line 160 of file process.C.

Referenced by RelaxSubset::run(), and CycleAll::run().

```
00161 {
00162   // If this guy's priority is not extremal, we'll have to resort
00163   // at some point in the future.  If it *is* extremal, we just tack
00164   // this process is on one of the queue's ends.
00165   if(Q && p->Priority <= Q.PeekFront()->Priority)
00166     Q.AddToFront(p);
00167   else if(Q && p->Priority >= Q.PeekBack()->Priority)
00168     Q.AddToBack(p);
00169   else
00170     {
00171       Q.AddToBack(p);
00172       sorted = 0;
00173     }
00174
```

4.48.3.4 void Process::assertOrder () `[private]`

Definition at line 29 of file process.C.

Referenced by runChild(), runChildren(), runChildrenRepeatFirst(), and runNextChild().

```
00030 {
00031   if (sorted) return;
00032   Process ** child;
00033   int numQ = Q.Number();
00034   if (numQ>0)
00035     {
00036       // Empty out the queue and then sort all the children
00037       // that were in there.
00038       if(!(child = new Process*[numQ]))
00039         DIE("Process::assertOrder memory failure");
00040
00041       for(int i=0; Q; ++i)
00042         child[i] = Q.RemoveFromBack();
00043
00044       QuickSortPtrs(child, numQ);
00045
```

```
      // ACK!  REMOVE ME
//        for(i=0; i<(nnonhelatoms-1); i++) {
//        for(j=(nnonhelatoms-1); j>i; j--) {
//          atom1 = atomorder[j-1];
//          atom2 = atomorder[j];
// //     debug_i ("SORT:atom1", atom1);
// //     debug_i ("SORT:atom2", atom2);
// //     debug_d ("SORT:a1err", atom_info[atom1].lfnc);
// //     debug_d ("SORT:a2err", atom_info[atom2].lfnc);
//       if(
//          (fabs(atom_info[atom1].lfnc-atom_info[atom2].lfnc) < 1e-12) &&
//          (atom1 > atom2)
//          ){
//         itemp = atomorder[j-1];
//         atomorder[j-1] = atomorder[j];
//         atomorder[j] = itemp;
//       }
//
//     }
//      }
//

      // for equal priority, sort by atom index
//       Process * temp;
//       int atom1, atom2;
//       for(int i=0; i<numQ-1; i++) {
//      for (int j = numQ-1; j > i;j--){
//        atom1 = ((gwRelaxAtom *) child[j-1])->atom_index; // ALSO FIX THIS IS NOW PUBLIC
//        atom2 = ((gwRelaxAtom *) child[j])->atom_index;
//        if(
//           (fabs(child[j-1]->Priority - child[j]->Priority) < 1e-12) &&
//           (atom1 < atom2)
//           ){
//          temp = child[j-1];
//          child[j-1] = child[j];
//          child[j] = temp;
//        }
//
//        }
//      }

      // Put the sorted list back into the queue.
      for(int i=0; i<numQ; ++i)
        Q.Add(child[i]);

      // Get rid of the extra array of pointers.
      delete[] child;

      sorted = 1;
    }

```

4.48.3.5 void Process::cleanQueue ()

Removes all queued jobs without run' ing them.

Definition at line 136 of file process.C.

Referenced by RelaxSubset::run().

```
{
  while(Q)
    Q.RemoveFromBack();
```

```
00140   // NB not deleted as in runChildren()
00141
```

4.48.3.6 double Process::getPriority () const [inline]

Returns the priority of this process.

Definition at line 156 of file process.h.

Referenced by gwRelaxAtom::run().

```
00157 { return Priority; }
00158
```

4.48.3.7 int Process::numChildren () const [inline]

Returns the *total* number of children created by this process. These children are not necessarily queued to execute. Call **numQueued**() (p. 396) if would like to find the number of children awaiting execution.

Definition at line 150 of file process.h.

```
00151 { return numC; }
00152
```

4.48.3.8 int Process::numQueued () const [inline]

Returns the number of children awaiting execution.

Definition at line 153 of file process.h.

```
00154 { return Q.Number(); }
00155
```

4.48.3.9 void Process::reset () [inline, virtual]

Run any required operations to reset the process.

Definition at line 36 of file process.h.

```
00036                     {}
00037
```

4.48.3.10 int Process::run (void) [pure virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented in **AtomBuildup** (p. 51), **CycleAll** (p. 120), **gwRelaxAtom** (p. 274), **AnimFrame** (p. 41), **jdErr_Eval** (p. 312), **jdIterator** (p. 316), **jdOcc_Count** (p. 319), **jdRand_Eval** (p. 322), **RelaxSubset** (p. 420), and **StopEval** (p. 493).

Referenced by main(), and runChildrenRepeatFirst().

4.48.3.11 int Process::runChild (int *i*)

Run the child with the i^th smallest priority. This child is ∗not∗ removed from the queue! Returns the exit status of the child.

Definition at line 143 of file process.C.

```
{
  if(i<0 || i>=Q.Number())
    return -1;

  // Make sure the queue is ordered.
  assertOrder();

  return Q.Get(i)->run();

```

4.48.3.12 void Process::runChildren ()

Execute all the children that have been added to the queue by order of their priority. After the call, the queue of children is empty.

Definition at line 117 of file process.C.

```
{
  // Make sure the queue is ordered.
  assertOrder();

//   Process * P;
//   while(Q){
//     P = Q.RemoveFromBack();
//     if (P->getPriority() >= 0)
//       P->run();
//   }

```

4.48.3.13 void Process::runChildrenRepeatFirst ()

Execute all the children that have been added to the queue by order of their priority. Repeat the first process at the end. After the call, the queue of children is empty.

Definition at line 98 of file process.C.

Referenced by CycleAll::run().

```
{
  // Make sure the queue is ordered.
  assertOrder();

   Process * P;
   if (Q)
     P = Q.RemoveFromBack();
   else
     return;

   P->run();
   while (Q)
     Q.RemoveFromBack()->run();
   P->run();

```

4.48.3.14 int Process::runNextChild ()

Runs the child with the next smallest priority. This child is removed from the queue and lost. Returns the exit status of the child.

Definition at line 154 of file process.C.

Referenced by RelaxSubset::run().

```
{
  assertOrder();
  return Q.RemoveFromBack()->run();

```

4.48.3.15 void Process::setPriority (double *pri*) [inline]

Sets the priority of the process (low priorities are executed first.

Definition at line 159 of file process.h.

```
{ Priority = pri; }

```

4.48.4 Member Data Documentation

4.48.4.1 Process ∗∗ Process::Children [protected]

An array of pointers to sub-processes.

Definition at line 126 of file process.h.

4.48.4.2 double Process::Priority [protected]

High priorities are executed first. These can range from -INF to +INF.

Definition at line 122 of file process.h.

4.48.4.3 Queue< Process ∗> Process::Q [private]

Definition at line 142 of file process.h.

4.48.4.4 bool Process::active [protected]

defaults to t, if f, the process is never run.

Definition at line 132 of file process.h.

4.48.4.5 int Process::numC [protected]

The number of total children (including zombies).

Definition at line 129 of file process.h.

4.48.4.6 int Process::sorted [private]

Definition at line 139 of file process.h.

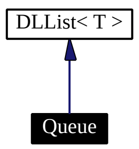
The documentation for this class was generated from the following files:

- **process.h**
- **process.C**

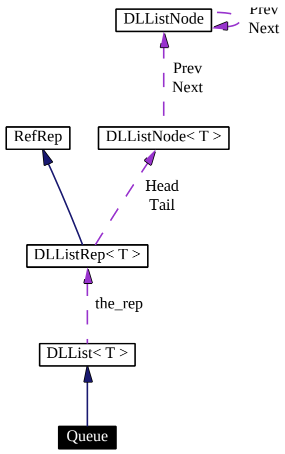
4.49 Queue Class Template Reference

`#include <queue.h>`

Inheritance diagram for Queue



Collaboration diagram for Queue:



Public Methods

- void **Add** (const T &x)
- void **Add** (const Queue< T > &q)
- T **Remove** ()
- const T& **Peek** () const
- T& **Peek** ()
- const Queue<T>& **operator+=** (const Queue< T > &q)

Friends

- Queue<T> **operator+** (Queue< T > x, const Queue< T > &y)

template<class T> class Queue

4.49.1 Member Function Documentation

4.49.1.1 template<class T> void Queue<T>::Add (const Queue< T > & *q*) `[inline]`

Definition at line 21 of file queue.h.

```
00022                 { AddToBack(q.List); }
00023
```

4.49.1.2 template<class T> void Queue<T>::Add (const T & *x*) [inline]

Reimplemented from **DLList** (p. 163).

Definition at line 19 of file queue.h.

Referenced by Process::assertOrder(), and bf_iterator::operator++().

```
00020                 { AddToBack(x); }
00021
```

4.49.1.3 template<class T> T & Queue<T>::Peek () [inline]

Definition at line 26 of file queue.h.

```
00026                         { return PeekFront(); }
00027
```

4.49.1.4 template<class T> const T & Queue<T>::Peek () const [inline]

Definition at line 25 of file queue.h.

```
00025                                     { return PeekFront(); }
00026
```

4.49.1.5 template<class T> T Queue<T>::Remove () [inline]

Reimplemented from **DLList** (p. 169).

Definition at line 23 of file queue.h.

Referenced by bf_iterator::operator++().

```
00024                 { return RemoveFromFront(); }
00025
```

4.49.1.6 template<class T> const Queue< T > & Queue<T>::operator+=<T> (const Queue< T > & *q*) [inline]

Definition at line 27 of file queue.h.

```
00028                 { Add(q); return *this; }
00029
```

4.49.2 Friends And Related Function Documentation

4.49.2.1 template<class T> Queue< T > operator+<T> (Queue< T > *x*, const Queue< T > & *y*) [friend]

Definition at line 29 of file queue.h.

```
            { return x+=y; }
```

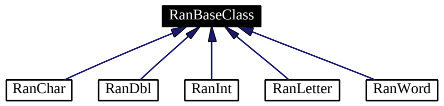
The documentation for this class was generated from the following file:

- **queue.h**

4.50 RanBaseClass Class Reference

`#include <randomgenerator.h>`

Inheritance diagram for RanBaseClass



Public Methods

- virtual **~RanBaseClass** ()
- void **Seed** (**uint** seed_val)

Protected Methods

- **RanBaseClass** ()
- double **GetReal** () const

4.50.1 Constructor & Destructor Documentation

4.50.1.1 RanBaseClass::~RanBaseClass () `[virtual]`

Definition at line 113 of file randomgenerator.C.

```
{}
```

4.50.1.2 RanBaseClass::RanBaseClass () `[protected]`

Definition at line 107 of file randomgenerator.C.

```
{
  if(!z.seeded)
      Seed();
```

4.50.2 Member Function Documentation

4.50.2.1 double RanBaseClass::GetReal () const `[protected]`

Definition at line 116 of file randomgenerator.C.

Referenced by RanLetter::Get(), RanChar::Get(), RanInt::Get(), and RanDbl::Get().

```
{
  double uni = z.u[z.i97] - z.u[z.j97];
  if(uni < 0.0) uni += 1.0;
  z.u[z.i97] = uni;
  z.i97 = (z.i97 == 1) ? 97 : z.i97-1;
```

```
    z.j97 = (z.j97 == 1) ? 97 : z.j97-1;
    z.c -= z.cd;
    if(z.c<0.0) z.c += z.cm;
    uni -= z.c;
    if(uni<0.0) uni += 1.0;
    return uni;

```

4.50.2.2 void RanBaseClass::Seed (uint *seed_val* = 0)

Definition at line 11 of file randomgenerator.C.

Referenced by RanBaseClass().

```
  {
    uint ij,kl;
    int i,j,k,l,ii,jj,m;
    double s,t;


    if (seed_val == 0){
      // This is a self-seed procedure
      // Added 4/11/2000 -JD
      // This effectively makes 0 an invalid seed.

#ifdef WIN32
      while (seed_val == 0) {

        // This sucks big-time: returns time in seconds since Jan 1, 1970.
        seed_val = (unsigned) time(0);
      }
#else
      // NOT WIN32
      while (seed_val == 0) { // 0 is an invalid seed value

        // This, of course, still sucks... but if we get interrupted in our read
        // from /dev/random below and don't get a full unsigned worth of random
        // bytes, we want to have *something* non-zero filling up seed_val.
        seed_val = (unsigned)getpid() ^ (unsigned)time(0);

#ifdef HAVE_LIBPTHREAD
        // If we are using threads on a system w/o /dev/random, and if two threads
        // both try to generate one of these "magic seeds" at exactly the same
        // time, we don't want them both getting identical rngs.
        seed_val ^= (unsigned)pthread_self();
#endif

        //#ifdef linux

        // I love /dev/random... but if we are on a system that doesn't support
        // it, this open will fail and we'll have to make due with the crappy
        // "randomness" from the above.
        int fd = open("/dev/random", O_RDONLY);
        if (fd != -1) {
          // Can we be interrupted in this read?  I guess so, so we'll settle for
          // some randomness, if not a full sizeof(s)-bytes worth...
          while (read(fd, &seed_val, sizeof(seed_val)) == 0);
          close(fd);
        }
        // #endif

      }

```

```
#endif

    } // end self-seed procedure


    // ij is the first seed to the random generator and kl is the
    // second seed.  These seeds are derived from the bits contained
    // in the number seed.  Specifically, interleaved bits are
    // assigned to each seed and then the bits are reversed (making the
    // lower order bits more significant.
    //
    ij = reverse_bits( 0xAAAAAAAA & seed_val );
    kl = reverse_bits( 0x77777777 & seed_val );
    ij = (uint) (31328 * ((double) ij / (double) UINT_MAX));
    kl = (uint) (30081 * ((double) kl / (double) UINT_MAX));

    i = (ij/177)%177+2;
    j = ij%177 + 2;
    k = (kl/169)%178 + 1;
    l = kl%169;

    for(ii=1; ii<=97; ii++)
      {
        s = 0.0;
        t = 0.5;
        for(jj=1; jj<=24; jj++)
          {
            m = (((i*j)%179)*k) % 179;
            i = j;
            j = k;
            k = m;
            l = (53*l + 1) % 169;
            if((l*m)%64 >= 32) s+= t;
            t *= 0.5;
          }
        z.u[ii] = s;
      }

```

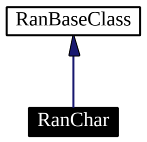
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

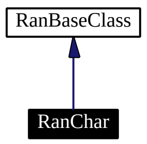
4.51 RanChar Class Reference

`#include <randomgenerator.h>`

Inheritance diagram for RanChar



Collaboration diagram for RanChar:



Public Methods

- **RanChar** ()
- **~RanChar** ()
- char **Get** () const

4.51.1 Constructor & Destructor Documentation

4.51.1.1 RanChar::RanChar () `[inline]`

Definition at line 122 of file randomgenerator.h.

```
00122                           {}
00123
```

4.51.1.2 RanChar::~RanChar () `[inline]`

Definition at line 123 of file randomgenerator.h.

```
00123                            {}
00124
```

4.51.2 Member Function Documentation

4.51.2.1 char RanChar::Get () const

Definition at line 153 of file randomgenerator.C.

```
00154   {
00155     int index = (int) (95.0 * GetReal());
00156     return (' ' + (char) index);
00157
```

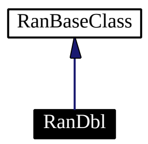
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

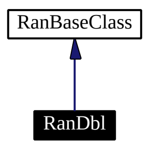
4.52 RanDbl Class Reference

`#include <randomgenerator.h>`

Inheritance diagram for RanDbl



Collaboration diagram for RanDbl:



Public Methods

- **RanDbl** (double minimum=0.0, double maximum=1.0)
- **~RanDbl** ()
- double **Get** () const
- double **Get** (double minimum, double maximum) const

Private Attributes

- double **min**
- double **max**

4.52.1 Constructor & Destructor Documentation

4.52.1.1 RanDbl::RanDbl (double *minimum* = 0.0, double *maximum* = 1.0)

Definition at line 130 of file randomgenerator.C.

```
00131   { min = minimum; max = maximum; }
00132
```

4.52.1.2 RanDbl::~RanDbl () `[inline]`

Definition at line 93 of file randomgenerator.h.

```
00093                           {}
00094
```

4.52.2 Member Function Documentation

4.52.2.1 double RanDbl::Get (double *minimum*, double *maximum*) const

Definition at line 136 of file randomgenerator.C.

```
{ return (minimum + (maximum-minimum)*GetReal()); }
```

4.52.2.2 double RanDbl::Get () const

Definition at line 133 of file randomgenerator.C.

Referenced by Envelope::Build().

```
{ return (min + (max-min)*GetReal()); }
```

4.52.3 Member Data Documentation

4.52.3.1 double RanDbl::max `[private]`

Definition at line 97 of file randomgenerator.h.

4.52.3.2 double RanDbl::min `[private]`

Definition at line 97 of file randomgenerator.h.

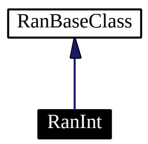
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

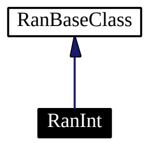
4.53 RanInt Class Reference

`#include <randomgenerator.h>`

Inheritance diagram for RanInt



Collaboration diagram for RanInt:



Public Methods

- **RanInt** (int minimum=0, int maximum=1)
- ~**RanInt** ()
- int **Get** () const
- int **Get** (int minimum, int maximum) const

Private Attributes

- int **min**
- int **max**

4.53.1 Constructor & Destructor Documentation

4.53.1.1 RanInt::RanInt (int *minimum* = 0, int *maximum* = 1)

Definition at line 139 of file randomgenerator.C.

```
00140 {
00141   if(minimum <= maximum)
00142     { min = minimum; max = maximum; }
00143   else
00144     { min = maximum; max = minimum; }
00145
```

4.53.1.2 RanInt::~RanInt () `[inline]`

Definition at line 109 of file randomgenerator.h.

```
00109                          {}
00110
```

4.53.2 Member Function Documentation

4.53.2.1 int RanInt::Get (int *minimum*, int *maximum*) const

Definition at line 150 of file randomgenerator.C.

```
{ return( minimum + (int) (((double) (maximum-minimum+1)) * GetReal()) ); }
```

4.53.2.2 int RanInt::Get () const

Definition at line 147 of file randomgenerator.C.

Referenced by RanWord::Get(), and Molecule::setRandom().

```
{ return( min + (int) (((double) (max-min+1)) * GetReal()) ); }
```

4.53.3 Member Data Documentation

4.53.3.1 int RanInt::max `[private]`

Definition at line 116 of file randomgenerator.h.

4.53.3.2 int RanInt::min `[private]`

Definition at line 116 of file randomgenerator.h.

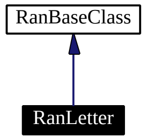
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

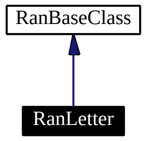
4.54 RanLetter Class Reference

#include <randomgenerator.h>

Inheritance diagram for RanLetter



Collaboration diagram for RanLetter:



Public Methods

- **RanLetter** ()
- ~**RanLetter** ()
- char **Get** () const

4.54.1 Constructor & Destructor Documentation

4.54.1.1 RanLetter::RanLetter () [inline]

Definition at line 132 of file randomgenerator.h.

```
00132                                 {}
00133
```

4.54.1.2 RanLetter::~RanLetter () [inline]

Definition at line 133 of file randomgenerator.h.

```
00133                                  {}
00134
```

4.54.2 Member Function Documentation

4.54.2.1 char RanLetter::Get () const

Definition at line 159 of file randomgenerator.C.

Referenced by RanWord::Get().

```
{
  int index = (int) (52.0 * GetReal());
  if(index < 26)
      return ('A' + (char) index);
  else
      return ('a' + (char) (index-26));

```

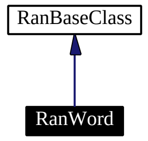
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

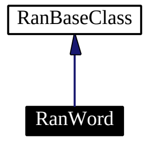
4.55 RanWord Class Reference

`#include <randomgenerator.h>`

Inheritance diagram for RanWord



Collaboration diagram for RanWord:



Public Methods

- **RanWord** ()
- **~RanWord** ()
- **DynString Get** (int exact_length) const
- **DynString Get** (int min_length=1, int max_length=100) const

4.55.1 Constructor & Destructor Documentation

4.55.1.1 RanWord::RanWord () [inline]

Definition at line 141 of file randomgenerator.h.

```
00141                                 {}
00142
```

4.55.1.2 RanWord::~RanWord () [inline]

Definition at line 142 of file randomgenerator.h.

```
00142                                  {}
00143
```

4.55.2 Member Function Documentation

4.55.2.1 DynString RanWord::Get (int *min_length* = 1, int *max_length* = 100) const

Definition at line 168 of file randomgenerator.C.

```
{
  int length;
  RanInt poof_a_number;
  RanLetter poof_a_letter;

  if(min_length<1)
      min_length = 1;
  if(max_length<min_length)
      max_length = min_length;

  if(min_length!=max_length)
      length = poof_a_number.Get(min_length, max_length);
  else
      length = min_length;

  DynString word;
  word.DestResizeToFit(length);
  for(int i=0; i<length; ++i)
      word[i] = poof_a_letter.Get();
  return word;

```

4.55.2.2 DynString RanWord::Get (int *exact_length*) const

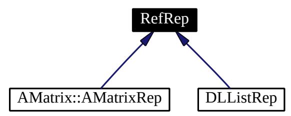
The documentation for this class was generated from the following files:

- **randomgenerator.h**
- **randomgenerator.C**

4.56 RefRep Class Reference

`#include <refrep.h>`

Inheritance diagram for RefRep



Public Methods

- **RefRep** ()
- **RefRep** (const RefRep &)
- virtual ~**RefRep** ()=0
- long **Reference** ()
- long **Dereference** ()
- long **References** () const

Private Attributes

- long **references**

4.56.1 Constructor & Destructor Documentation

4.56.1.1 RefRep::RefRep () `[inline]`

Definition at line 95 of file refrep.h.

```
00096 : references(1L)
00097
```

4.56.1.2 RefRep::RefRep (const RefRep &) `[inline]`

Definition at line 111 of file refrep.h.

```
00112 : references(1L)
00113
```

4.56.1.3 RefRep::~RefRep () `[inline, pure virtual]`

Definition at line 156 of file refrep.h.

```
00157 { /* empty */ }
00158
```

4.56.2 Member Function Documentation

4.56.2.1 long RefRep::Dereference () [inline]

Definition at line 134 of file refrep.h.

Referenced by ConstIterator::Copy(), Iterator::Copy(), DLList::operator=(), AMatrix::operator=(), AMatrix::~AMatrix(), ConstIterator::~ConstIterator(), DLList::~DLList(), and Iterator::~Iterator().

```
{
        if (--references<=0)
        {
                delete this;
                return 1;
        }
```

4.56.2.2 long RefRep::Reference () [inline]

Definition at line 120 of file refrep.h.

Referenced by AMatrix::AMatrix(), ConstIterator::ConstIterator(), ConstIterator::Copy(), Iterator::Copy(), DLList::DLList(), Iterator::Iterator(), DLList::operator=(), and AMatrix::operator=().

```
{ return ++references; }
```

4.56.2.3 long RefRep::References () const [inline]

Definition at line 148 of file refrep.h.

```
{ return references; }
```

4.56.3 Member Data Documentation

4.56.3.1 long RefRep::references [private]

Definition at line 78 of file refrep.h.

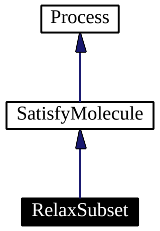
The documentation for this class was generated from the following file:

- **refrep.h**

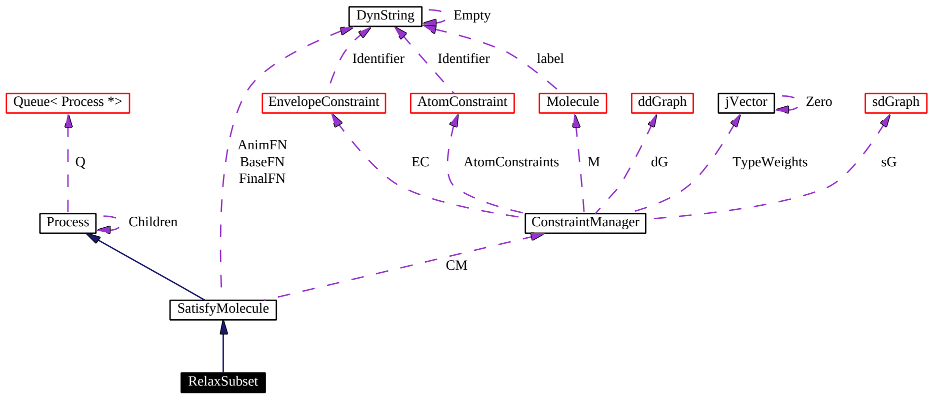
4.57 RelaxSubset Class Reference

#include <jd_relax_subset.h>

Inheritance diagram for RelaxSubset



Collaboration diagram for RelaxSubset:



Public Methods

- **RelaxSubset** (const RelaxSubset &ca)
- **RelaxSubset** (**ConstraintManager** &**CM**, double in_fraction=0.5)
- ~**RelaxSubset** ()
- int **run** ()
- void **setFraction** (double F)

    set the fraction of worst atoms {0..1} to be affected.

Private Attributes

- int **NumToRelax**

    == (int) (number of atoms)(fraction to relax).*

- double* **PriorityList**
- double **Fraction**

    Fraction' defines the number of worst atoms {0..1} to be affected.

4.57.1 Detailed Description

//

File: **jd_relax_subset.h**

This is a 'job' that works in the GRAPE system. It relaxes a subset of the molecule's atoms.

//

Definition at line 21 of file jd_relax_subset.h.

4.57.2 Constructor & Destructor Documentation

4.57.2.1 RelaxSubset::RelaxSubset (const RelaxSubset & *ca*)

Definition at line 3 of file jd_relax_subset.C.

```
00004   : SatisfyMolecule(*(ca.CM))
00005 {
00006   // FIX THIS...  I don't know what goes here.  -jd
00007   DIE("relax subset copt constructor called ");
00008
```

4.57.2.2 RelaxSubset::RelaxSubset (ConstraintManager & *cm_in*, double *Fraction_in* = 0.5)

It is assumed that fraction is between 0 and 1. The behavior of RelaxSubset when passed a double >= 1 will be to call a RelaxAtom job on all of the atoms in the molecule. If the constructor recieves a negative double, nothing is done to the molecule, and a error message is written to Standard Error.

Definition at line 11 of file jd_relax_subset.C.

```
00012   : SatisfyMolecule(cm_in)
00013   {
00014
00015     Fraction = Fraction_in;
00016
00017     // Create one child process for each atom in the list. Later only
00018     // the 'fraction' of these atoms will be queued to run.
00019     numC = CM->numAtoms();
00020
00021     Children = new Process*[numC];
00022
00023     PriorityList = new double [numC];
00024
00025     // Create a new relaxation process for every atom in the molecule:
00026     for(int i=0; i<numC; ++i)
00027       Children[i] = new gwRelaxAtom(*CM,i);
00028
```

4.57.2.3 RelaxSubset::∼RelaxSubset ()

Definition at line 30 of file jd_relax_subset.C.

```
00031   {
```

```
  // Don't delete my children (since Process takes care of it).
  // i.e. dont do for(int i=0; i<numChildren; ++i) delete Children[i].
  // or delete[] Children.

  //    CM_ptr=0;
  delete [] PriorityList;
```

4.57.3 Member Function Documentation

4.57.3.1 int RelaxSubset::run (void) [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 40 of file jd_relax_subset.C.

```
00041 {
00042 #ifdef JOB_VERBOSE
00043   cerr << "Start: \tRelaxSubset::run()"<<endl;
00044 #endif
00045 
00046   // DEBUG
00047   if (Fraction < 0 || Fraction > 1) {
00048     cerr << "tried to call RelaxSubset on a fraction = " << Fraction << endl;
00049     return (-1);
00050   }
00051 
00052   NumToRelax = (int) (Fraction * numC );
00053 
00054   // Compute the priorities of the children, based on their
00055   // residuals
00056   for(int i=0; i<numC; ++i)
00057     {
00058       double tempPriority = CM->stress(i);
00059 
00060       Children[i]->setPriority(tempPriority);
00061       PriorityList[i] = tempPriority;
00062 
00063 #ifdef JOB_VERBOSE3
00064       cerr << "Set:   \t" << "Priority for atom " <<i<<"="<< tempPriority << endl;
00065 #endif
00066     }
00067 
00068   //double Cutoff = LinearSelect(PriorityList, NumToRelax, numC);
00069 
00070   // Queue children to run queue
00071   for(int i=0; i<numC; ++i)
00072     {
00073       //if (Children[i]->getPriority() > Cutoff) addChild(Children[i]);
00074       addChild(Children[i]); // add em all
00075     }
00076 
00077   // now do the queue!
00078   for(int i=0; i<NumToRelax; ++i){
00079     runNextChild();
00080   }
00081 
```

4.57.3.2 void RelaxSubset::setFraction (double *F*) [inline]

set the fraction of worst atoms {0..1} to be affected.

Definition at line 40 of file jd_relax_subset.h.

```
00040                    {Fraction = F;}
00041
```

4.57.4 Member Data Documentation

4.57.4.1 double RelaxSubset::Fraction [private]

Fraction' defines the number of worst atoms {0..1} to be affected.

Definition at line 52 of file jd_relax_subset.h.

4.57.4.2 int RelaxSubset::NumToRelax [private]

== (int) (number of atoms)∗(fraction to relax).

Definition at line 48 of file jd_relax_subset.h.

4.57.4.3 double ∗ RelaxSubset::PriorityList [private]

Definition at line 49 of file jd_relax_subset.h.

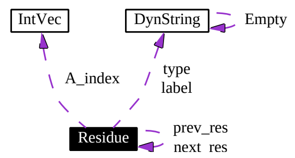
The documentation for this class was generated from the following files:

- **jd_relax_subset.h**
- **jd_relax_subset.C**

4.58 Residue Class Reference

`#include <residue.h>`

Collaboration diagram for Residue:



Public Methods

- const int& **ID** () const

  need residue_registry need to point to atoms need to have atoms point to residue.

- int& **ID** ()
- const **DynString**& **Type** () const
- **DynString**& **Type** ()
- const **DynString**& **Label** () const
- **DynString**& **Label** ()
- const **jVector**& **operator** ∗ () const
- **jVector**& **operator** ∗ ()
- int **Setup** (**Molecule** &Mol, int start_atom, Residue ∗prev_res_in, Residue ∗next_res_in)
- int **Compare** (const Residue &a) const
- const **IntVec**& **GetAtomList** () const
- const Residue& **operator=** (const Residue &res)
- int **Output** (ostream &os) const
- int **Input** (istream &is)
- int **OutputBinary** (ostream &os) const
- int **InputBinary** (istream &is)
- void **Copy** (const Residue &res)
- **MAKECOMPARISONS_WITH_COMPARE** (Residue)
- **MAKE_IO_STREAM_OPERATORS** (Residue)
- **MAKE_BIN_STREAM_OPERATORS** (Residue)

Private Methods

- **Residue** (int identifier=UNINITIALIZED_RES_ID)
- void **AddAtom** (**Molecule** &Mol, int atom_id, const **DynString** &atom_type, const char ∗label_end)
- void **AddBond** (**Molecule** &Mol, int atom_id1, int atom_id2)

Private Attributes

- int **id**
- int **Type_index**
- **DynString label**

- **DynString type**
- int **numAtoms**
- int **numBonds**
- **IntVec A_index**
- int **N_chain_atom**
- int **CA_chain_atom**
- int **O_chain_atom**
- int **C_chain_atom**
- int **CB_chain_atom**
- Residue∗ **next_res**
- Residue∗ **prev_res**

Friends

- class **Molecule**
- class **ResidueRegistry**

4.58.1 Constructor & Destructor Documentation

4.58.1.1 Residue::Residue (int *identifier* = UNINITIALIZED_RES_ID) `[inline, private]`

Definition at line 119 of file residue.h.

```
00119                                      {
00120   id=identifier;
00121   type.Clear();
00122   Type_index = UNINITIALIZED_RES_ID;
00123   label = "uninitialized residue";
00124   numAtoms = -1;
00125   numBonds = -1;
00126   N_chain_atom = UNINITIALIZED_ATOM_ID;
00127   CA_chain_atom = UNINITIALIZED_ATOM_ID;
00128   O_chain_atom = UNINITIALIZED_ATOM_ID;
00129   C_chain_atom = UNINITIALIZED_ATOM_ID;
00130   CB_chain_atom = UNINITIALIZED_ATOM_ID;
00131   next_res = 0;
00132   prev_res = 0;
00133
00134
```

4.58.2 Member Function Documentation

4.58.2.1 void Residue::AddAtom (Molecule & *Mol*, int *atom_id*, const DynString & *atom_type*, const char ∗ *label_end*) `[private]`

Definition at line 13 of file residue.C.

Referenced by Setup().

```
00013                                                                                                 {
00014
00015
00016   DynString label = "A:";
00017   label += atom_id;
```

```
00018   label += " ";
00019   label += ResidueRegistry::code_3[Type_index];
00020   label += id;
00021   label += " ";
00022   label += label_end;
00023
00024   //cerr << "residue.C:: Adding atom:"<< atom_id << " as " <<  label << endl;
00025
00026   A_index.Append(atom_id);
00027   AtomRegistry::SetType (Mol.A[atom_id],atom_type, label, id);
00028
00029
```

4.58.2.2 void Residue::AddBond (Molecule & *Mol*, int *atom_id1*, int *atom_id2*) [private]

Definition at line 5 of file residue.C.

Referenced by Setup().

```
00005                                                                      {
00006   //    cerr << "bonding " <<  Mol[atom_id1].Label() << "  TO  " << Mol[atom_id2].Label() <<endl;
00007   Mol[atom_id1].AddBond(atom_id2);
00008   Mol[atom_id2].AddBond(atom_id1);
00009
00010
```

4.58.2.3 int Residue::Compare (const Residue & *a*) const [inline]

Definition at line 154 of file residue.h.

```
00155 { return (id-a.id); }
00156
```

4.58.2.4 void Residue::Copy (const Residue & *res*)

Definition at line 399 of file residue.C.

Referenced by operator=().

```
00399                                         {
00400
00401   id = res.id;
00402   type = res.type;
00403   Type_index = res.Type_index;
00404   label = res.label;
00405   numAtoms = res.numAtoms;
00406   numBonds = res.numBonds;
00407   A_index = res.A_index;
00408   N_chain_atom = res.N_chain_atom;
00409   CA_chain_atom = res.CA_chain_atom;
00410   O_chain_atom = res.O_chain_atom;
00411   C_chain_atom = res.C_chain_atom;
00412   CB_chain_atom = res.CB_chain_atom;
00413   next_res = res.next_res;
00414   prev_res = res.prev_res;
00415
00416
```

4.58.2.5 const IntVec & Residue::GetAtomList () const [inline]

Definition at line 136 of file residue.h.

```
00137 {  return A_index; }
00138
```

4.58.2.6 int & Residue::ID () [inline]

Definition at line 141 of file residue.h.

```
00142 { return id; }
00143
```

4.58.2.7 const int & Residue::ID () const [inline]

need residue_registry need to point to atoms need to have atoms point to residue.

Definition at line 139 of file residue.h.

Referenced by ResidueRegistry::SetType_code_1(), and Molecule::readFasta().

```
00140 { return id; }
00141
```

4.58.2.8 int Residue::Input (istream & *is*) [inline]

Definition at line 161 of file residue.h.

```
00162 { is >> id ; return id; }
00163
```

4.58.2.9 int Residue::InputBinary (istream & *is*) [inline]

Definition at line 168 of file residue.h.

```
00169     { is >= id; is >= pos; return id; }
00170
```

4.58.2.10 DynString & Residue::Label () [inline]

Definition at line 151 of file residue.h.

```
00152 { return label; }
00153
```

4.58.2.11 const DynString & Residue::Label () const [inline]

Definition at line 149 of file residue.h.

Referenced by ResidueRegistry::SetType_code_1().

```
00150 { return label; }
00151
```

4.58.2.12 Residue::MAKECOMPARISONS_WITH_COMPARE (Residue)

4.58.2.13 Residue::MAKE_BIN_STREAM_OPERATORS (Residue)

4.58.2.14 Residue::MAKE_IO_STREAM_OPERATORS (Residue)

4.58.2.15 int Residue::Output (ostream & *os*) const `[inline]`

Definition at line 157 of file residue.h.

```
00158     { os << id << ' ' << pos; return id; }
00159
```

4.58.2.16 int Residue::OutputBinary (ostream & *os*) const `[inline]`

Definition at line 164 of file residue.h.

```
00165     { os <= id; os <= pos; return id; }
00166
```

4.58.2.17 int Residue::Setup (Molecule & *Mol*, int *start_atom*, Residue * *prev_res_in*, Residue * *next_res_in*)

sets up appropraite atoms in **Molecule** (p. 356) adds bond information within atoms returns last atom handled+1 (the next unhandled atom), UNINITIALIZED_ATOM_ID on err

Definition at line 31 of file residue.C.

Referenced by Molecule::readFasta().

```
00031                                                                                            {
00032   A_index.Resize(0);
00033
00034   next_res = next_res_in;
00035   prev_res = prev_res_in;
00036
00037   DynString N_type_atom('N');
00038   DynString C_type_atom('C');
00039   DynString O_type_atom('O');
00040   DynString S_type_atom('S');
00041
00042
00043   if (Type_index >= 20) {
00044     AddAtom(Mol, start_atom+0, C_type_atom, "Dummy Carbon");
00045     N_chain_atom = CA_chain_atom = O_chain_atom = CA_chain_atom =
00046       CB_chain_atom = start_atom+0;
00047
00048     // connect residues
00049     if (prev_res && prev_res->C_chain_atom >= 0)
00050       AddBond(Mol, prev_res->C_chain_atom, start_atom);
00051
00052     // doing both to ensure connection in non ordered entry case
00053     if (next_res && next_res->N_chain_atom >= 0)
00054       AddBond(Mol, start_atom+2, next_res->N_chain_atom);
00055
00056   } else {
00057
00058     // add the protein chain atoms
```

```
    AddAtom(Mol, start_atom+0, N_type_atom, "N_chain");
    N_chain_atom = start_atom+0;

    AddAtom(Mol, start_atom+1, C_type_atom, "CA_chain");
    CA_chain_atom = start_atom+1;

    AddAtom(Mol, start_atom+2, C_type_atom, "C_chain");
    C_chain_atom = start_atom+2;

    AddAtom(Mol, start_atom+3, O_type_atom, "O_chain");
    O_chain_atom = start_atom+3;

    CB_chain_atom = start_atom+4; // reset for Glycine


    // connect residues
    if (prev_res && prev_res->C_chain_atom >= 0)
      AddBond(Mol, prev_res->C_chain_atom, start_atom); // C(prev) -> N

    // add the standard chain bonds
    AddBond(Mol, start_atom, start_atom+1); // N -> CA
    AddBond(Mol, start_atom+1, start_atom+2); // CA -> C
    AddBond(Mol, start_atom+2, start_atom+3); // C -> O

    // doing both to ensure connection in non ordered entry case
    if (next_res && next_res->N_chain_atom >= 0)
      AddBond(Mol, start_atom+2, next_res->N_chain_atom); // C -> N(next)

  }

  // switch adds stoms and bonds approrpiate for each residue type
  // & fixes CB in glycine

  switch (Type_index){ // here we encode residue differences

    //         0   1   2   3   4   5   6   7   8
    //        " N   CA  C   O   CB ", /* ala */
    //        " N   CA  C   O   CB  SG ", /* cys */
    //        " N   CA  C   O   CB  CG  OD1 OD2", /* asp */
    //        " N   CA  C   O   CB  CG  CD  OE1 OE2", /* glu */
    //        " N   CA  C   O   CB  CG  CD1 CD2 CE1 CE2 CZ ", /* phe */
    //        " N   CA  C   O   C  ", /* gly */

    // 0   A  alanine
  case 0:
    AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
    AddBond(Mol, CA_chain_atom, CB_chain_atom);
    break;

    // 1   C  cystine
  case 1:
    AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
    AddAtom(Mol, start_atom+5, S_type_atom, "SG");
    AddBond(Mol, CA_chain_atom, CB_chain_atom);
    AddBond(Mol, start_atom+5, CB_chain_atom);
    break;

    // 2   D  aspartate asp
  case 2:
    AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
    AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
    AddAtom(Mol, start_atom+6, O_type_atom, "OD1");  // O_delta1 atom
    AddAtom(Mol, start_atom+7, O_type_atom, "OD2");  // O_delta2 atom
    AddBond(Mol, CA_chain_atom, CB_chain_atom);
    AddBond(Mol, CB_chain_atom, start_atom+5);
```

```
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);

  break;
  // 3   E  glutamate
case 3:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, C_type_atom, "CD");  // C_delta atom
  AddAtom(Mol, start_atom+7, O_type_atom, "OE1");  // O_epsilon1 atom
  AddAtom(Mol, start_atom+8, O_type_atom, "OE2");  // O_epsilon2 atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+6, start_atom+7);
  AddBond(Mol, start_atom+6, start_atom+8);

  break;
  // 4   F  phenylalanine
case 4:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, C_type_atom, "CD1");  // C_delta atom
  AddAtom(Mol, start_atom+7, C_type_atom, "CD2");  // C_delta atom
  AddAtom(Mol, start_atom+8, C_type_atom, "CE1");  // C_epsilon atom
  AddAtom(Mol, start_atom+9, C_type_atom, "CE2");  // C_epsilon atom
  AddAtom(Mol, start_atom+10, C_type_atom, "CZ");  // C_zeta atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);
  AddBond(Mol, start_atom+6, start_atom+8);
  AddBond(Mol, start_atom+7, start_atom+9);
  AddBond(Mol, start_atom+8, start_atom+10);
  AddBond(Mol, start_atom+9, start_atom+10);

  break;
  // 5   G  glycine
case 5:
  CB_chain_atom = CA_chain_atom; // not perfect, but close.

  break;

  //         0   1   2   3   4   5   6   7   8   9
  //       " N   CA  C   O   CB  CG  ND1 CD2 CE1 NE2", /* his */
  //       " N   CA  C   O   CB  CG1 CG2 CD1", /* ile */
  //       " N   CA  C   O   CB  CG  CD  CE  NZ ", /* lys */
  //       " N   CA  C   O   CB  CG  CD1 CD2", /* leu */

  // 6   H  histidine
case 6:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, N_type_atom, "ND1");  // N_delta atom
  AddAtom(Mol, start_atom+7, C_type_atom, "CD2");  // C_delta atom
  AddAtom(Mol, start_atom+8, C_type_atom, "CE1");  // C_epsilon atom
  AddAtom(Mol, start_atom+9, N_type_atom, "NE2");  // N_epsilon atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);
  AddBond(Mol, start_atom+6, start_atom+8);
  AddBond(Mol, start_atom+7, start_atom+9);
  AddBond(Mol, start_atom+8, start_atom+9);
  break;
```

```

    // 7   I  isoleucine
case 7:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG1");  // C_gamma atom
  AddAtom(Mol, start_atom+6, C_type_atom, "CG2");  // C_gamma atom
  AddAtom(Mol, start_atom+7, C_type_atom, "CD1");  // C_delta atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, CB_chain_atom, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);
  break;

    // 8   K  lysine
case 8:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, C_type_atom, "CD");  // C_delta atom
  AddAtom(Mol, start_atom+7, C_type_atom, "CE");  // C_epsilon atom
  AddAtom(Mol, start_atom+8, N_type_atom, "NZ");  // N_zeta atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+6, start_atom+7);
  AddBond(Mol, start_atom+7, start_atom+8);
  break;

    // 9   L  leucine
case 9:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, C_type_atom, "CD1");  // C_delta1 atom
  AddAtom(Mol, start_atom+7, C_type_atom, "CD2");  // C_delta2 atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);
  break;

  //       " N   CA  C   O   CB  CG  SD  CE ", /* met */
  //       " N   CA  C   O   CB  CG  OD1 ND2", /* asn */
    // 10  M  methionine
case 10:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, S_type_atom, "SD");  // sulphur
  AddAtom(Mol, start_atom+7, C_type_atom, "CE");  // C_epsilon atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+6, start_atom+7);
  break;

    // 11  N  asparagine  asn
case 11:
  AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
  AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
  AddAtom(Mol, start_atom+6, O_type_atom, "OD1");  // O_delta1 atom
  AddAtom(Mol, start_atom+7, N_type_atom, "ND2");  // N_delta2 atom
  AddBond(Mol, CA_chain_atom, CB_chain_atom);
  AddBond(Mol, CB_chain_atom, start_atom+5);
  AddBond(Mol, start_atom+5, start_atom+6);
  AddBond(Mol, start_atom+5, start_atom+7);
  break;

```

```
    //        0   1   2   3   4   5   6   7   8   9
    //       " N   CA  C   O   CB  CG  CD ", /* pro */
    //       " N   CA  C   O   CB  CG  CD  OE1 NE2", /* gln */
    //       " N   CA  C   O   CB  CG  CD  NE  CZ  NH1 NH2", /* arg */
    //       " N   CA  C   O   CB  OG ", /* ser */

    //  12  P  proline
case 12:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
      AddAtom(Mol, start_atom+6, C_type_atom, "CD");  // C_delta atom
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, start_atom+5, start_atom+6);
      AddBond(Mol, start_atom+6, N_chain_atom);
      break;

    //  13  Q  glutamine
case 13:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
      AddAtom(Mol, start_atom+6, C_type_atom, "CD");  // C_delta atom
      AddAtom(Mol, start_atom+7, O_type_atom, "OE1");  // O_epsilon atom
      AddAtom(Mol, start_atom+8, N_type_atom, "NE2");  // N_epsilon atom
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, start_atom+5, start_atom+6);
      AddBond(Mol, start_atom+6, start_atom+7);
      AddBond(Mol, start_atom+6, start_atom+8);
      break;

      //  14  R  arginine
case 14:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG");  // C_gamma atom
      AddAtom(Mol, start_atom+6, C_type_atom, "CD");  // C_delta atom
      AddAtom(Mol, start_atom+7, N_type_atom, "NE2");  // N_epsilon atom
      AddAtom(Mol, start_atom+8, C_type_atom, "CZ");  // C_zeta atom
      AddAtom(Mol, start_atom+9, N_type_atom, "NH1");
      AddAtom(Mol, start_atom+10, N_type_atom, "NH2");
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, start_atom+5, start_atom+6);
      AddBond(Mol, start_atom+6, start_atom+7);
      AddBond(Mol, start_atom+7, start_atom+8);
      AddBond(Mol, start_atom+8, start_atom+9);
      AddBond(Mol, start_atom+8, start_atom+10);
      break;

      //   15  S  serine
case 15:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, O_type_atom, "OG");  // O_gamma atom
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      break;

    //        0   1   2   3   4   5   6   7   8   9   10  11  12  13
    //       " N   CA  C   O   CB  OG1 CG2", /* thr */
    //       " N   CA  C   O   CB  CG1 CG2", /* val */
    //       " N   CA  C   O   CB  CG  CD1 CD2 NE1 CE2 CE3 CZ2 CZ3 CH2", /* trp */
    //       " N   CA  C   O   CB  CG  CD1 CD2 CE1 CE2 CZ  OH ", /* tyr */


    //  16  T  threonine
```

```
case 16:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, O_type_atom, "OG1");  // O_gamma atom
      AddAtom(Mol, start_atom+6, C_type_atom, "CG2");  // C_gamma atom
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, CB_chain_atom, start_atom+6);
      break;

    //  17  V  valine
case 17:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG1");  // C_gamma atom
      AddAtom(Mol, start_atom+6, C_type_atom, "CG2");  // C_gamma atom
      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, CB_chain_atom, start_atom+6);
      break;

    //   18  W  tryptophan
case 18:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG");
      AddAtom(Mol, start_atom+6, C_type_atom, "CD1");
      AddAtom(Mol, start_atom+7, C_type_atom, "CD2");
      AddAtom(Mol, start_atom+8, N_type_atom, "NE1");
      AddAtom(Mol, start_atom+9, C_type_atom, "CE2");
      AddAtom(Mol, start_atom+10, C_type_atom, "CE3");
      AddAtom(Mol, start_atom+11, C_type_atom, "CZ2");
      AddAtom(Mol, start_atom+12, C_type_atom, "CZ3");
      AddAtom(Mol, start_atom+13, C_type_atom, "CH2");

      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, start_atom+5, start_atom+6);
      AddBond(Mol, start_atom+5, start_atom+7);
      AddBond(Mol, start_atom+6, start_atom+8);
      AddBond(Mol, start_atom+8, start_atom+10);
      AddBond(Mol, start_atom+7, start_atom+10);
      AddBond(Mol, start_atom+7, start_atom+9);
      AddBond(Mol, start_atom+10, start_atom+12);
      AddBond(Mol, start_atom+9, start_atom+11);
      AddBond(Mol, start_atom+11, start_atom+13);
      AddBond(Mol, start_atom+12, start_atom+13);

      break;
    //   19  Y  tyrosine
case 19:
      AddAtom(Mol, start_atom+4, C_type_atom, "CB");  // C_beta atom
      AddAtom(Mol, start_atom+5, C_type_atom, "CG");
      AddAtom(Mol, start_atom+6, C_type_atom, "CD1");
      AddAtom(Mol, start_atom+7, C_type_atom, "CD2");
      AddAtom(Mol, start_atom+8, C_type_atom, "CE1");
      AddAtom(Mol, start_atom+9, C_type_atom, "CE2");
      AddAtom(Mol, start_atom+10, C_type_atom, "CZ");
      AddAtom(Mol, start_atom+11, O_type_atom, "OH");

      AddBond(Mol, CA_chain_atom, CB_chain_atom);
      AddBond(Mol, CB_chain_atom, start_atom+5);
      AddBond(Mol, start_atom+5, start_atom+6);
      AddBond(Mol, start_atom+5, start_atom+7);
      AddBond(Mol, start_atom+6, start_atom+8);
      AddBond(Mol, start_atom+7, start_atom+9);
      AddBond(Mol, start_atom+8, start_atom+10);
      AddBond(Mol, start_atom+9, start_atom+10);
```

```
00384         AddBond(Mol, start_atom+10, start_atom+11);
00385
00386       break;
00387       //  20  X  Dummy
00388   case 20:
00389     // do nothing
00390       break;
00391   }
00392
```

4.58.2.18 DynString & Residue::Type () [inline]

Definition at line 146 of file residue.h.

```
00147 { return type; }
00148
```

4.58.2.19 const DynString & Residue::Type () const [inline]

Definition at line 144 of file residue.h.

Referenced by ResidueRegistry::SetType_code_1().

```
00145 { return type; }
00146
```

4.58.2.20 jVector& Residue::operator * ()

4.58.2.21 const jVector& Residue::operator * () const

4.58.2.22 const Residue & Residue::operator= (const Residue & *res*)

Definition at line 396 of file residue.C.

```
00397   { Copy(res); return *this; }
00398
```

4.58.3 Friends And Related Function Documentation

4.58.3.1 class Molecule [friend]

Definition at line 29 of file residue.h.

4.58.3.2 class ResidueRegistry [friend]

Definition at line 30 of file residue.h.

4.58.4 Member Data Documentation

4.58.4.1 IntVec Residue::A_index [private]

Definition at line 102 of file residue.h.

4.58.4.2 int Residue::CA_chain_atom [private]

Definition at line 108 of file residue.h.

4.58.4.3 int Residue::CB_chain_atom [private]

Definition at line 111 of file residue.h.

4.58.4.4 int Residue::C_chain_atom [private]

Definition at line 110 of file residue.h.

4.58.4.5 int Residue::N_chain_atom [private]

Definition at line 107 of file residue.h.

4.58.4.6 int Residue::O_chain_atom [private]

Definition at line 109 of file residue.h.

4.58.4.7 int Residue::Type_index [private]

Definition at line 94 of file residue.h.

4.58.4.8 int Residue::id [private]

Definition at line 91 of file residue.h.

4.58.4.9 DynString Residue::label [private]

Definition at line 96 of file residue.h.

4.58.4.10 Residue * Residue::next_res [private]

Definition at line 113 of file residue.h.

4.58.4.11 int Residue::numAtoms [private]

Definition at line 99 of file residue.h.

4.58.4.12 int Residue::numBonds [private]

Definition at line 100 of file residue.h.

4.58.4.13 Residue * Residue::prev_res [private]

Definition at line 114 of file residue.h.

4.58.4.14 DynString Residue::type [private]

Definition at line 97 of file residue.h.

The documentation for this class was generated from the following files:

- **residue.h**
- **residue.C**

4.59 ResidueRegistry Class Reference

#include <residue_registry.h>

Static Public Methods

- int **CheckType_code_1** (**DynString** &in_label)

  checks if this is a valid 1 code, return -1 or species number (position in arrays below).

- int **SetType_code_1** (**Residue** &r, **DynString** &in_label)

  sets reside to label type.

Static Public Attributes

- const char* **names** [num_known_residue_species]

  names of the residues.

- const char* **code_1** [num_known_residue_species]

  names of the residues.

- const char* **code_3** [num_known_residue_species]

  names of the residues.

- const int **numAtoms** [num_known_residue_species]

  number of atoms in the residues.

- const int **numBonds** [num_known_residue_species]

  number of atoms in the residues.

4.59.1 Member Function Documentation

4.59.1.1 int ResidueRegistry::CheckType_code_1 (DynString & *in_label*) [inline, static]

checks if this is a valid 1 code, return -1 or species number (position in arrays below).

Definition at line 91 of file residue_registry.h.

Referenced by Molecule::readFasta().

```
00091                                                                  {
00092   for (int i = 0; i < num_known_residue_species; i++){
00093     if (in_label == ResidueRegistry::code_1[i])
00094       return i;
00095   }
00096
```

4.59.1.2 int ResidueRegistry::SetType_code_1 (Residue & *r*, DynString & *in_type*) [inline, static]

sets reside to label type.

Definition at line 100 of file residue_registry.h.

Referenced by Molecule::readFasta().

```
00100                                                                        {
00101   for (int i = 0; i < num_known_residue_species; i++){
00102     if (in_type == ResidueRegistry::code_1[i]){
00103       r.Type_index = i;
00104       r.Type() = in_type;
00105       r.Label() = names[i];
00106       r.numAtoms = numAtoms[i];
00107       r.numBonds = numBonds[i];
00108 
00109       return r.ID();
00110     }
00111   }
00112 
```

4.59.2 Member Data Documentation

4.59.2.1 const char * ResidueRegistry::code_1[num_known_residue_species] [static]

Initial value:

```
 {
  "A", "C", "D", "E", "F", "G", "H", "I", "K", "L", \
  "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y", \
  "X"
}
```

names of the residues.

Definition at line 174 of file globals.C.

4.59.2.2 const char * ResidueRegistry::code_3[num_known_residue_species] [static]

Initial value:

```
 {
  "ALA", "CYS", "ASP", "GLU", "PHE", "GLY", "HIS", "ILE", "LYS", "LEU", \
  "MET", "ASN", "PRO", "GLN", "ARG", "SER", "THR", "VAL", "TRP", "TYR", \
  "DUM"
}
```

names of the residues.

Definition at line 181 of file globals.C.

4.59.2.3 const char * ResidueRegistry::names[num_known_residue_species] [static]

Initial value:

```
 {
  "alanine","cystine","aspartate","glutamate","phenylalanine","glycine", \
  "histidine","isoleucine","lysine","leucine","methionine", \
  "asparagine","proline","glutamine","arginine","serine","threonine", \
  "valine","tryptophan","tyrosine","dummy"
}
```

names of the residues.

Definition at line 166 of file globals.C.

4.59.2.4 const int ResidueRegistry::numAtoms[num_known_residue_species] [static]

Initial value:

```
 {
  5,6,8,9,11,4,10,8,9,8,8,8,7,9,11,6,7,7,14,12,  1
}
```

number of atoms in the residues.

Definition at line 188 of file globals.C.

4.59.2.5 const int ResidueRegistry::numBonds[num_known_residue_species] [static]

Initial value:

```
 {

  1,2,4,5, 8,0, 7,4,5,4,4,4,4,5, 7,2,3,3,12, 9,  0

}
```

number of atoms in the residues.

Definition at line 213 of file globals.C.

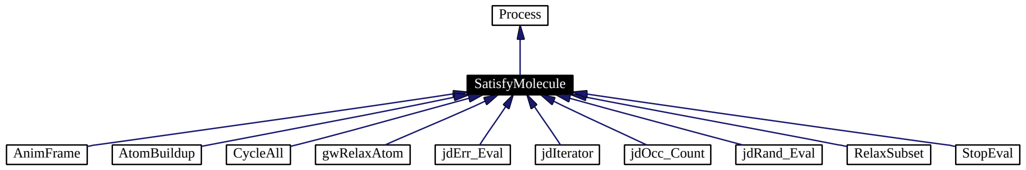
The documentation for this class was generated from the following files:

- **residue_registry.h**
- **globals.C**

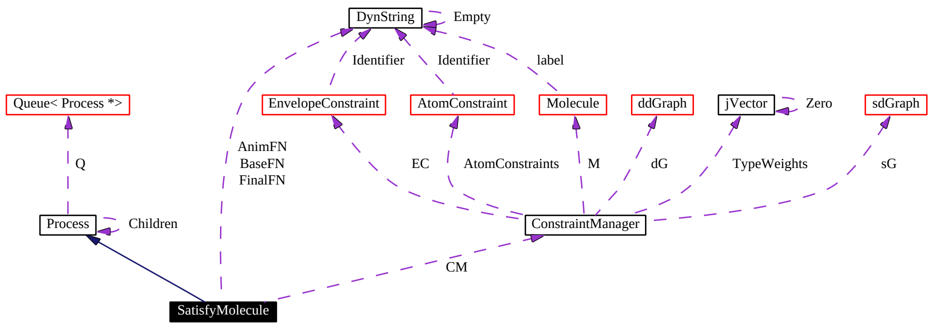
4.60 SatisfyMolecule Class Reference

`#include <satisfy_molecule.h>`

Inheritance diagram for SatisfyMolecule



Collaboration diagram for SatisfyMolecule:



Public Methods

- virtual **~SatisfyMolecule** ()

  destructor.

- void **setMolecule** (const **Molecule** &mol)
- const **Molecule**& **getMolecule** () const

Static Public Attributes

- **DynString BaseFN** = "Base"
- **DynString AnimFN** = "Anim"
- **DynString FinalFN** = "Final"

Protected Methods

- **SatisfyMolecule** (**ConstraintManager** &cm)

  Constructors.

- **SatisfyMolecule** (const SatisfyMolecule &sm)

Protected Attributes

- **ConstraintManager**∗ **CM**

4.60.1 Detailed Description

//

File: satisfy_molucle.h

This class is a place-holder class. It combines the **ConstraintManager** (p. 97) and the **Process** (p. 391) classes. It is very much like the ManipulateMolecule class except that all accesses to the **Molecule** (p. 356) are done through the **ConstraintManager** (p. 97) that is associated with an instance of the process. This allows the **ConstraintManager** (p. 97) to perform smart updates and fast computations (wrt to satisfy() and dSatisfy() calls).

//

Definition at line 23 of file satisfy_molecule.h.

4.60.2 Constructor & Destructor Documentation

4.60.2.1 SatisfyMolecule::~SatisfyMolecule () `[inline, virtual]`

destructor.

Definition at line 26 of file satisfy_molecule.h.

```
00026                              { CM = 0; }
00027
```

4.60.2.2 SatisfyMolecule::SatisfyMolecule (ConstraintManager & *cm*) `[inline, protected]`

Constructors.

Definition at line 47 of file satisfy_molecule.h.

```
00047                                         { CM = &cm; }
00048
```

4.60.2.3 SatisfyMolecule::SatisfyMolecule (const SatisfyMolecule & *sm*) `[inline, protected]`

Definition at line 48 of file satisfy_molecule.h.

```
00049   { CM = sm.CM; }
00050
```

4.60.3 Member Function Documentation

4.60.3.1 const Molecule & SatisfyMolecule::getMolecule () const `[inline]`

Definition at line 35 of file satisfy_molecule.h.

```
00036   { return CM->getMolecule(); }
00037
```

4.60.3.2 void SatisfyMolecule::setMolecule (const Molecule & *mol*) [inline]

Definition at line 33 of file satisfy_molecule.h.

```
00034   { CM->setMolecule(mol); }
00035
```

4.60.4 Member Data Documentation

4.60.4.1 DynString SatisfyMolecule::AnimFN = "Anim" [static]

Definition at line 44 of file globals.C.

4.60.4.2 DynString SatisfyMolecule::BaseFN = "Base" [static]

Reimplemented in **jdErr_Eval** (p. 314), **jdOcc_Count** (p. 320), and **jdRand_Eval** (p. 323).

Definition at line 43 of file globals.C.

4.60.4.3 ConstraintManager * SatisfyMolecule::CM [protected]

Definition at line 51 of file satisfy_molecule.h.

4.60.4.4 DynString SatisfyMolecule::FinalFN = "Final" [static]

Definition at line 45 of file globals.C.

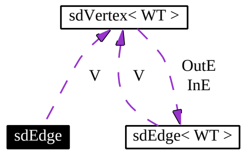
The documentation for this class was generated from the following files:

- **satisfy_molecule.h**
- **globals.C**

4.61 sdEdge Class Template Reference

`#include <sdvertex.h>`

Collaboration diagram for sdEdge:



Public Methods

- **sdEdge** ()
- **sdEdge** (const sdEdge< WT > &e)
- **~sdEdge** ()
- int **Compare** (const sdEdge< WT > &e) const
- **MAKECOMPARISONS_WITH_COMPARE** (sdEdge< WT >)

Public Attributes

- **sdVertex**<WT>* **V**
- WT **W**

Friends

- class **sdVertex**< **WT** >

template<class WT = double> class sdEdge

4.61.1 Constructor & Destructor Documentation

4.61.1.1 template<class WT = double> sdEdge<WT>::sdEdge<WT> () `[inline]`

Definition at line 31 of file sdvertex.h.

```
00031                          { V=0; }
00032
```

4.61.1.2 template<class WT = double> sdEdge<WT>::sdEdge<WT> (const sdEdge< WT > & *e*) `[inline]`

Definition at line 32 of file sdvertex.h.

```
00032                                          { V=e.V; W = e.W; }
00033
```

4.61.1.3 template<class WT = double> sdEdge<WT>::~sdEdge<WT> () [inline]

Definition at line 33 of file sdvertex.h.

```
00033                          { /* empty */ }
00034
```

4.61.2 Member Function Documentation

4.61.2.1 template<class WT> int sdEdge< WT >::Compare (const sdEdge< WT > & *e*) const [inline]

Definition at line 15 of file sdvertex.inc.

```
00016   { return (V->ID - e.V->ID); }
00017
```

4.61.2.2 template<class WT = double> sdEdge<WT>::MAKECOMPARISONS_WITH_COMPARE (sdEdge< WT >)

4.61.3 Friends And Related Function Documentation

4.61.3.1 template<class WT = double> class sdVertex [friend]

Definition at line 29 of file sdvertex.h.

4.61.4 Member Data Documentation

4.61.4.1 template<class WT = double> sdVertex< WT > * sdEdge<WT>::V

Definition at line 38 of file sdvertex.h.

4.61.4.2 template<class WT = double> WT sdEdge<WT>::W

Definition at line 40 of file sdvertex.h.

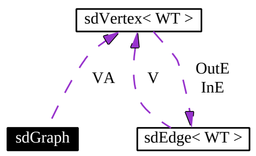
The documentation for this class was generated from the following files:

- **sdvertex.h**
- **sdvertex.inc**

4.62 sdGraph Class Template Reference

`#include <sdgraph.h>`

Collaboration diagram for sdGraph:



Public Methods

- **sdGraph** ()
- **sdGraph** (const sdGraph< DT, WT > &sdg)
- **sdGraph** (**ddGraph**< DT, WT > &ddg, bool KeepIds=false)
- **~sdGraph** ()
- int **NumVertices** () const
- int **NumEdges** () const
- void **Copy** (const sdGraph< DT, WT > &sdg)
- void **Copy** (**ddGraph**< DT, WT > &ddg, bool KeepIds=false)
- const sdGraph<DT,WT>& **operator=** (const sdGraph< DT, WT > &sdg)
- const sdGraph<DT,WT>& **operator=** (const **ddGraph**< DT, WT > &ddg)
- int **GetVertexIndex** (const DT &x) const
- int **HasVertex** (const DT &x) const
- int **HasEdge** (int i, int j) const
- int **OutDegree** (int i) const
- int **InDegree** (int i) const
- int **Degree** (int i) const
- const **sdVertex**<WT>& **GetVertex** (int i) const
- **sdVertex**<WT>& **GetVertex** (int i)
- const **sdVertex**<WT>& **operator**[] (int i) const
- **sdVertex**<WT>& **operator**[] (int i)
- const DT& **GetData** (int i) const
- DT& **GetData** (int i)
- const DT& **operator()** (int i) const
- DT& **operator()** (int i)
- const WT& **GetWeight** (int i, int j, int alt=0) const
- WT& **GetWeight** (int i, int j, int alt=0)
- const WT& **operator()** (int i, int j, int alt=0) const
- WT& **operator()** (int i, int j, int alt=0)
- sdGraph<DT,WT> **SubGraph** (const **SDG_SetInt** &vertset) const
- int **Output** (ostream &os) const
- int **Input** (istream &is)
- ostream& **PrintDotty** (ostream &os, int edges=0) const
- **KAdjMat** **AdjMatrix** () const
- **KAdjMat*** **TravMatrices** (int src, int k, **KAdjMat** *P=0) const
- **SDG_IntList**** **TravLists** (int src, int k, **KAdjMat** *P=0) const
- int **InputBinary** (istream &is)

- int **OutputBinary** (ostream &os) const
- **neighbor_iterator begin_neighbor** (int vndx=0, int rev=0)
- **df_iterator begin_df** (int vndx=0, int rev=0)
- **bf_iterator begin_bf** (int vndx=0, int rev=0)
- **path_iterator begin_path** (int src, int tgt, int len, **DLList**< int > *const *tl=0, int(*filter)(const sdGraph< DT, WT > &graph, const int *path, int ndx, int pathlen, void *arbitrary_criteria)=0, void *criteria=0)
- **jVector Dijkstra** (int vndx=0, int rev=0, int wc=0, int *pred=0) const
- int **DFTimeStamps** (int *discover, int *finish, int rev=0) const
- **SDG_SetSetInt SCC** () const
- **SDG_SetSetInt StronglyConnectedComponents** () const
- sdGraph<DT,WT> **SCCGraph** () const
- int **NumSCC** () const

Private Methods

- void **Clear** ()
- int **ValidVertex** (int i) const
- int **ValidData** (int i) const
- int **OutDegree** (const **sdVertex**< WT > *vert) const
- int **InDegree** (const **sdVertex**< WT > *vert) const
- int **Degree** (const **sdVertex**< WT > *vert) const
- void **DF_Visit** (int u, **sdg_color** *color, int *time, int *discover, int *finish, **SDG_SetInt** *treemembers, int rev) const
- int **Initialize_Single_Source** (int vndx, **jVector** &D, int *P) const
- void **Relax** (int u, int v, **jVector** &D, int *P, int rev, int wc) const

Private Attributes

- int **NumV**
- int **NumE**
- **sdVertex**<WT>** **VA**
- DT* **Data**

Friends

- class **ddGraph< DT, WT >**
- istream& **operator>=** (istream &is, sdGraph< DT, WT > &sdg)
- ostream& **operator** (ostream &os, const sdGraph< DT, WT > &sdg)

template<class DT, class WT = double> class sdGraph

4.62.1 Constructor & Destructor Documentation

4.62.1.1 template<class DT, class WT> sdGraph< DT, WT >::sdGraph<DT, WT> () `[inline]`

Definition at line 308 of file sdgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
  VA   = 0;
  Data = 0;

```

4.62.1.2 template<class DT, class WT> sdGraph< DT, WT >::sdGraph<DT, WT> (const sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 317 of file sdgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
  VA = 0;
  Data = 0;
  Copy(sdg);

```

4.62.1.3 template<class DT, class WT> sdGraph< DT, WT >::sdGraph<DT, WT> (ddGraph< DT, WT > & *ddg*, bool *KeepIds* = false) `[inline]`

Definition at line 328 of file sdgraph.inc.

```
{
  NumV = 0;
  NumE = 0;
  VA = 0;
  Data = 0;
  Copy(ddg, KeepIds);

```

4.62.1.4 template<class DT, class WT> sdGraph< DT, WT >::~sdGraph<DT, WT> () `[inline]`

Definition at line 338 of file sdgraph.inc.

```
{ Clear(); }

```

4.62.2 Member Function Documentation

4.62.2.1 template<class DT, class WT> KAdjMat sdGraph< DT, WT >::AdjMatrix () const `[inline]`

Definition at line 665 of file sdgraph.inc.

Referenced by TravLists(), and TravMatrices().

```
{
  KAdjMat A(NumV);
  for(int i=0; i<NumV; ++i)
      for(int j=0; j<VA[i]->NumO; ++j)
          A.Connect(i, VA[i]->OutE[j].V->ID);
  return A;

```

4.62.2.2 template<class DT, class WT> void sdGraph< DT, WT >::Clear () `[inline, private]`

Definition at line 14 of file sdgraph.inc.

Referenced by Copy(), InputBinary(), and ~sdGraph().

```
00015   {
00016     for(int i=0; i<NumV; ++i)
00017         delete VA[i];
00018     if(VA) { delete[] VA; VA = 0; }
00019     if(Data) { delete[] Data; Data = 0; }
00020
```

4.62.2.3 template<class DT, class WT> void sdGraph< DT, WT >::Copy (ddGraph< DT, WT > & *ddg*, bool *KeepIds* = false) `[inline]`

Definition at line 68 of file sdgraph.inc.

```
00069   {
00070     int numout, numin, ndx, current, neighbor;
00071     map< int, int > IdMap;
00072     map< DT, int > DataMap;
00073     sdVertex<WT> ** VA_tmp=0;
00074
00075     map<DT, ddVertex<WT>* >::const_iterator dynvert_it;
00076     set<ddEdge<WT> >::const_iterator eit;
00077     ddVertex<WT> * vert=0;
00078     ddEdge<WT> edge;
00079
00080     // Delete everything everything:
00081     //
00082     Clear();
00083
00084     if(ddg.NumV<1) return;
00085
00086     // Try to allocate space to hold all of the vertices and the data:
00087     //
00088     if(!(VA = new sdVertex<WT>*[ddg.NumV])) return;
00089
00090     if(!(Data = new DT[ddg.NumV]))
00091       { delete[] VA; VA = 0; return; }
00092
00093     // Temporary vertex storage:
00094     //
00095     if(!(VA_tmp = new sdVertex<WT>*[ddg.NumV]))
00096       { delete[] VA; delete[] Data; VA = 0; Data = 0; return; }
00097
00098     NumV = ddg.NumV;
00099
00100     // Initialize the vertices:
00101     //
00102     for(int i=0; i<NumV; ++i) VA[i] = 0;
00103
00104     // First, we need to loop through all the vertices in the dynamic graph
00105     // and find out how many edges each of them has (so we can create
00106     // enough space to hold everything.  While we're doing that, we might
00107     // as well copy the data into the static vertices.
00108     //
00109     int v=0;
00110
00111     if (KeepIds){
```

```
    int MinVertex = v;  // idgen has no interface for this -- assume zero
    int MaxVertex = ddg.IDG.Max();

    // next 12 lines are repetitive, and should be integrated w/ above -jd
    NumV = MaxVertex+1;

    delete [] VA;
    VA = new sdVertex<WT>*[NumV];

    delete [] Data;
    Data = Data = new DT[NumV];

    delete [] VA_tmp;
    VA_tmp = new sdVertex<WT>*[NumV];

    for(int i=0; i<NumV; ++i) VA[i] = 0;

    //      for(dynvert_it = ddg.VA.begin(); dynvert_it!=ddg.VA.end(); ++dynvert_it)
    for(int VertexNumber = MinVertex; VertexNumber < MaxVertex+1; VertexNumber ++)
      {
        if(v==NumV)
          {
            // This is an error condition:  the dynamic graph told us there
            // were only NumV vertices but here we are getting the NumV+1'th!
            //
            break;
          }

        if (ddg.HasVertex(VertexNumber)) {

          //    Data[v] = (*dynvert_it).first;
          Data[v] = ddg.GetData(VertexNumber);

          //    vert = (*dynvert_it).second;
          vert = ddg.GetVertex(Data[v]); //discards const on copy function ?? -jd

          // Find out how many outbound and inbound edges this vertex has:
          //
          numout = vert->OutDegree();
          numin  = vert->InDegree();

          // Create the vertex:
          //
          //      VA_tmp[v] = new sdVertex<WT>(vert->ID, numout, numin);
          VA_tmp[v] = new sdVertex<WT>(VertexNumber, numout, numin);

          // Remember that this data was associated with the v_th vertex:
          //
          DataMap[Data[v]] = v;

          // end of case where vertex VertexNumber' exists in ddg
        }         else            {
          // case where vertex VertexNumber' does not exist in ddg
          Data[v] = 0;
          VA_tmp[v] = new sdVertex<WT>(VertexNumber, 0, 0);
          DataMap[Data[v]] = -1;
        }

        v++;
      }

    // NB sorting routine has been removed (!)
    // this may effect other routines on sDgraphs
    // if sorting does not occur.   -jd

```

```
      for(v=0; v<NumV; ++v)
        {
          ndx = DataMap[Data[v]];
          VA[v] = VA_tmp[ndx];

          // This allows us to resolve where edges point:
          //
          IdMap[VA[v]->ID] = v;

          // Now set the new id simply to its index in the array:
          //
          VA[v]->ID = v;
        }
      delete[] VA_tmp;


    } else { // default:  do not keep same IDs

      for(dynvert_it = ddg.VA.begin(); dynvert_it!=ddg.VA.end(); ++dynvert_it)
        {
          if(v==NumV)
            {
              // This is an error condition:  the dynamic graph told us there
              // were only NumV vertices but here we are getting the NumV+1'th!
              //
              break;
            }

          Data[v] = ($\ast$dynvert_it).first;
          vert = ($\ast$dynvert_it).second;

          // Find out how many outbound and inbound edges this vertex has:
          //
          numout = vert->OutDegree();
          numin  = vert->InDegree();

          // Create the vertex:
          //
          VA_tmp[v] = new sdVertex<WT>(vert->ID, numout, numin);

          // Remember that this data was associated with the v_th vertex:
          //
          DataMap[Data[v]] = v;

          v++;
        }

      // Now that all the vertices are created we can sort the data and then
      // rearrange the vertices (to match up with their respective data):
      //
      QuickSort(Data, NumV);
      for(v=0; v<NumV; ++v)
        {
          ndx = DataMap[Data[v]];
          VA[v] = VA_tmp[ndx];

          // This allows us to resolve where edges point:
          //
          IdMap[VA[v]->ID] = v;

          // Now set the new id simply to its index in the array:
          //
          VA[v]->ID = v;
        }
      delete[] VA_tmp;
```

```
  } // end default case -- do not keep same IDs -jd

  // Now copy in all the edges.  This involves iterating through all the
  // vertices from the dynamic graph again and copying the edges into
  // the appropriate vertex.
  //

  int e;
  for(dynvert_it = ddg.VA.begin(); dynvert_it!=ddg.VA.end(); ++dynvert_it)
    {
      // Get the vertex pointer and then find out where the corresponding
      // vertex is located in the static array:
      //
      vert = ($\ast$dynvert_it).second;
      current = IdMap[vert->ID];

      // Iterate through the outbound edges
      //
      e = 0;
      for(eit=vert->OutE.begin(); eit!=vert->OutE.end(); ++eit)
        {
          // Get the edge and then find out the index (in the static array)
          // of the vertex this edge refers to:
          //
          edge     = *eit;
          neighbor = IdMap[edge.V->ID];

          // Add the edge to this vertex:
          //
          VA[current]->OutE[e].V = VA[neighbor];
          VA[current]->OutE[e].W = edge.W;
          e++;
        }

      // Iterate through the inbound edges
      //
      e = 0;
      for(eit=vert->InE.begin(); eit!=vert->InE.end(); ++eit)
        {
          edge     = *eit;
          neighbor = IdMap[edge.V->ID];

          // Add the to<-from edge to this vertex:
          //
          VA[current]->InE[e].V = VA[neighbor];
          VA[current]->InE[e].W = edge.W;
          e++;
        }

      // Sort the edges:
      //
      VA[current]->SortEdges();
    }

```

4.62.2.4 template<class DT, class WT> void sdGraph< DT, WT >::Copy (const sdGraph< DT, WT > & *sdg*) [inline]

Definition at line 25 of file sdgraph.inc.

Referenced by operator=(), and sdGraph().

```
{
  if(this == &sdg) return;

  // Delete everything:
  //
  Clear();

  if(sdg.NumV<1) return;
  if(!(VA = new sdVertex<WT>*[sdg.NumV])) return;
  if(!(Data = new DT[sdg.NumV]))
    { delete[] VA; VA = 0; return; }

  // Copy the vertices.
  for(int v=0; v<sdg.NumV; ++v)
      if(!(VA[v] = new sdVertex<WT>(sdg.VA[v]->ID, sdg.VA[v]->NumO, sdg.VA[v]->NumI)))
          { cerr << "sdGraph::Copy\n"; raise(SIGQUIT); }

  // Copy the edges.
  for(int v=0; v<sdg.NumV; ++v)
    {
      // Copy the outbound edges in.
      for(int e=0; e<sdg.VA[v]->NumO; ++e)
        {
          VA[v]->OutE[e].V = VA[(sdg.VA[v]->OutE[e].V->ID)];
          VA[v]->OutE[e].W = sdg.VA[v]->OutE[e].W;
        }

      // Copy the inbound edges in.
      for(int e=0; e<sdg.VA[v]->NumI; ++e)
        {
          VA[v]->InE[e].V = VA[(sdg.VA[v]->InE[e].V->ID)];
          VA[v]->InE[e].W = sdg.VA[v]->InE[e].W;
        }

      Data[v] = sdg.Data[v];
    }
```

4.62.2.5 template<class DT, class WT> int sdGraph< DT, WT >::DFTimeStamps (int ∗ *discover*, int ∗ *finish*, int *rev* = 0) const [inline]

Definition at line 107 of file sdgalg.inc.

Referenced by SCC().

```
{
  if(NumV<1) return 0;

  int time = -1;

  // Color all the vertices white to mark them as unvisited:
  //
  sdg_color color[NumV];
  for(int u=0; u<NumV; ++u)
      color[u] = sdg_white;

  for(int u=0; u<NumV; ++u)
      if(color[u] == sdg_white)
          DF_Visit(u, color, &time, discover, finish, 0, rev);
  return 1;
```

4.62.2.6 template<class DT, class WT> void sdGraph< DT, WT >::DF_Visit (int *u*, sdg_color ∗ *color*, int ∗ *time*, int ∗ *discover*, int ∗ *finish*, SDG_SetInt ∗ *treemembers*, int *rev*) const [inline, private]

Definition at line 69 of file sdgalg.inc.

Referenced by DFTimeStamps(), and SCC().

```
{
  color[u] = sdg_gray;        // vertex u has just been discovered.
  if(treemembers)
     treemembers->insert(u);
     // treemembers->Add(u);

  if(discover && time)
      discover[u] = ++($\ast$time);

  int v;
  // Explore edge u-v:
  if(!rev)
    {
      for(int i=0; i<VA[u]->NumO; ++i)
        {
          v = VA[u]->OutE[i].V->ID;
          if(color[v] == sdg_white)
              DF_Visit(v, color, time, discover, finish, treemembers, rev);
        }
    }
  else
    {
      for(int i=0; i<VA[u]->NumI; ++i)
        {
          v = VA[u]->InE[i].V->ID;
          if(color[v] == sdg_white)
              DF_Visit(v, color, time, discover, finish, treemembers, rev);
        }
    }

```

4.62.2.7 template<class DT, class WT = double> int sdGraph<DT, WT>::Degree (const sdVertex< WT > ∗ *vert*) const [private]

4.62.2.8 template<class DT, class WT> int sdGraph< DT, WT >::Degree (int *i*) const [inline]

Definition at line 392 of file sdgraph.inc.

```
{
  if(!ValidVertex(i))
    { cerr << "sdGraph::Degree\n"; raise(SIGQUIT); }

```

4.62.2.9 template<class DT, class WT> jVector sdGraph< DT, WT >::Dijkstra (int *vndx* = 0, int *rev* = 0, int *wc* = 0, int ∗ *pred* = 0) const [inline]

Definition at line 35 of file sdgalg.inc.

```
{
```

```
IntPriorityQueue<double> Q(NumV,1);
jVector D;
int u_ndx, num_nbrs, u_nbr;
double u_pri;

D.Resize(NumV);

Initialize_Single_Source(vndx,D,pred);

// Copy all the vertices into the queue:
for(int v=0; v<NumV; ++v)
    Q.Add(v, D[v] );

while(!Q.IsEmpty())
  {
    Q.Remove(u_ndx, u_pri);
    D[u_ndx] = u_pri;

    // Relax all the neighbors of u.
    num_nbrs = !rev ? GetVertex(u_ndx).NumO : GetVertex(u_ndx).NumI;
    for(int j=0; j<num_nbrs; ++j)
      {
        u_nbr = !rev ? GetVertex(u_ndx)[j].ID : GetVertex(u_ndx)(j).ID;
        Relax(u_ndx, u_nbr, D, pred, rev, wc);
        Q.Modify(u_nbr, D[u_nbr]);
      }
  }

```

4.62.2.10 template<class DT, class WT> DT & sdGraph< DT, WT >::GetData (int *i*) `[inline]`

Definition at line 432 of file sdgraph.inc.

```
{
  if(!ValidData(i))
    { cerr << "sdGraph::GetData\n"; cerr << NumV << ',' << i << '\n'; raise(SIGQUIT); }

```

4.62.2.11 template<class DT, class WT> const DT & sdGraph< DT, WT >::GetData (int *i*) const `[inline]`

Definition at line 424 of file sdgraph.inc.

Referenced by ConstraintManager::getAssocAtoms(), ConstraintManager::getConstraint(), ConstraintManager::getParents(), operator()(), and ConstraintManager::~ConstraintManager().

```
{
  if(!ValidData(i))
    { cerr << "const sdGraph::GetData\n"; cerr << i << '\n'; raise(SIGQUIT); }

```

4.62.2.12 template<class DT, class WT> sdVertex< WT > & sdGraph< DT, WT >::GetVertex (int *i*) `[inline]`

Definition at line 408 of file sdgraph.inc.

```
00409 {
00410   if(!ValidVertex(i))
00411     { cerr << i << ',' << NumV << ",sdGraph::GetVertex\n"; raise(SIGQUIT); }
00412
```

4.62.2.13 template<class DT, class WT> const sdVertex< WT > & sdGraph< DT, WT >::GetVertex (int *i*) const [inline]

Definition at line 400 of file sdgraph.inc.

Referenced by operator[]().

```
00401 {
00402   if(!ValidVertex(i))
00403     { cerr << "sdGraph::GetVertex\n"; raise(SIGQUIT); }
00404
```

4.62.2.14 template<class DT, class WT> int sdGraph< DT, WT >::GetVertexIndex (const DT & *x*) const [inline]

Definition at line 350 of file sdgraph.inc.

Referenced by HasVertex().

```
00351 { return BinSearch(Data, NumV, x); }
00352
```

4.62.2.15 template<class DT, class WT> WT & sdGraph< DT, WT >::GetWeight (int *i*, int *j*, int *alt* = 0) [inline]

Definition at line 458 of file sdgraph.inc.

```
00459 {
00460   if(!ValidVertex(i) || !ValidVertex(j))
00461     { cerr << "sdGraph::GetWeight\n"; raise(SIGQUIT); }
00462
```

4.62.2.16 template<class DT, class WT> const WT & sdGraph< DT, WT >::GetWeight (int *i*, int *j*, int *alt* = 0) const [inline]

Definition at line 448 of file sdgraph.inc.

Referenced by Relax(), and operator()().

```
00449 {
00450   if(!ValidVertex(i) || !ValidVertex(j))
00451     { cerr << "sdGraph::GetWeight\n"; raise(SIGQUIT); }
00452
```

4.62.2.17 template<class DT, class WT> int sdGraph< DT, WT >::HasEdge (int *i*, int *j*) const [inline]

Definition at line 358 of file sdgraph.inc.

Referenced by path_iterator::operator++().

```
{
  // Don't have the edge if we don't have one of the vertices:
  //
  if(i<0 || i>=NumV || j<0 || j>=NumV) return 0;

  return VA[i]->HasEdgeTo((*VA[j]));

```

4.62.2.18 template<class DT, class WT> int sdGraph< DT, WT >::HasVertex (const DT & *x*) const [inline]

Definition at line 354 of file sdgraph.inc.

Referenced by ConstraintManager::remove().

```
{ return (GetVertexIndex(x) != -1); }

```

4.62.2.19 template<class DT, class WT = double> int sdGraph<DT, WT>::InDegree (const sdVertex< WT > * *vert*) const [private]

4.62.2.20 template<class DT, class WT> int sdGraph< DT, WT >::InDegree (int *i*) const [inline]

Definition at line 384 of file sdgraph.inc.

```
{
  if(!ValidVertex(i))
    { cerr << "sdGraph::InDegree\n"; raise(SIGQUIT); }

```

4.62.2.21 template<class DT, class WT> int sdGraph< DT, WT >::Initialize_Single_Source (int *vndx*, jVector & *D*, int * *P*) const [inline, private]

Definition at line 5 of file sdgalg.inc.

```
{
  for(int i=0; i<NumV; ++i)
    {
      D[i] = INF;
      if(P) P[i] = -1;
    }

```

4.62.2.22 template<class DT, class WT> int sdGraph< DT, WT >::Input (istream & *is*) [inline]

Definition at line 880 of file sdgraph.inc.

```
{ return 1; }

```

4.62.2.23 template<class DT, class WT> int sdGraph< DT, WT >::InputBinary (istream & *is*) [inline]

Definition at line 769 of file sdgraph.inc.

Referenced by operator>=().

```
{
  // Delete everything.
  //
  Clear();

  // Read in "NumV":
  //
  is >= NumV;

  // Read in "NumE":
  //
  is >= NumE;

  int vid;
  // Read in "VA":
  //
  if(NumV>0)
    {
      // Create the vertices:
      //
      if(!(VA = new sdVertex<WT>*[NumV]))
          { cerr << "sdGraph::InputBinary#1\n"; raise(SIGQUIT); }
      for(int i=0; i<NumV; ++i)
        {
          if(!(VA[i] = new sdVertex<WT>))
            { cerr << "sdGraph::InputBinary#2\n"; raise(SIGQUIT); }
          VA[i]->ID = i;
          VA[i]->NumO = 0;
          VA[i]->NumI = 0;
          VA[i]->OutE = 0;
          VA[i]->InE = 0;
        }

      for(int i=0; i<NumV; ++i)
        {
          // Read in number of out-edges:
          //
          is >= VA[i]->NumO;

          // Read in number of in-edges:
          //
          is >= VA[i]->NumI;

          if(VA[i]->NumO>0 &&
             !(VA[i]->OutE = new sdEdge<WT>[VA[i]->NumO]))
              { cerr << "sdGraph::InputBinary#3\n"; raise(SIGQUIT); }

          if(VA[i]->NumI>0 &&
             !(VA[i]->InE = new sdEdge<WT>[VA[i]->NumI]))
              { cerr << "sdGraph::InputBinary#4\n"; raise(SIGQUIT); }

          // Read in out-edges:
          //
          if(VA[i]->NumO>0)
            {
              for(int j=0; j<VA[i]->NumO; ++j)
                {
                  // Read in the id and point to the correct vertex:
```

```
                    //
                    is >= vid;

                    if(vid < 0 || vid>=NumV)
                      { cerr << "sdGraph::InputBinary#5\n"; raise(SIGQUIT); }

                    VA[i]->OutE[j].V = VA[vid];

                    // Read in the weight of the edge:
                    //
                    is >= VA[i]->OutE[j].W;
                  }
              }

            // Read in in-edges:
            //
            if(VA[i]->NumI>0)
              {
                for(int j=0; j<VA[i]->NumI; ++j)
                  {
                    // Read in the id and point to the correct vertex:
                    //
                    is >= vid;

                    if(vid < 0 || vid>=NumV)
                      { cerr << "sdGraph::InputBinary#6\n"; raise(SIGQUIT); }

                    VA[i]->InE[j].V = VA[vid];

                    // Read in the weight of the edge:
                    //
                    is >= VA[i]->InE[j].W;
                  }
              }

          }
      }

    // Read in "Data":
    //
    if(NumV>0)
      {
        if(!(Data = new DT[NumV]))
            { cerr << "sdGraph::InputBinary#7\n"; raise(SIGQUIT); }
        for(int i=0; i<NumV; ++i)
            is >= Data[i];
      }

```

4.62.2.24 template<class DT, class WT> int sdGraph< DT, WT >::NumEdges () const [inline]

Definition at line 346 of file sdgraph.inc.

```
  { return NumE; }

```

4.62.2.25 template<class DT, class WT> int sdGraph< DT, WT >::NumSCC () const [inline]

Definition at line 357 of file sdgalg.inc.

```
00358   { return SCC().Number(); }
00359
```

4.62.2.26 template<class DT, class WT> int sdGraph< DT, WT >::NumVertices () const [inline]

Definition at line 342 of file sdgraph.inc.

Referenced by ConstraintManager::Number().

```
00343   { return NumV; }
00344
```

4.62.2.27 template<class DT, class WT = double> int sdGraph<DT, WT>::OutDegree (const sdVertex< WT > * *vert*) const [private]

4.62.2.28 template<class DT, class WT> int sdGraph< DT, WT >::OutDegree (int *i*) const [inline]

Definition at line 376 of file sdgraph.inc.

```
00377   {
00378     if(!ValidVertex(i))
00379       { cerr << "sdGraph::OutDegree\n"; raise(SIGQUIT); }
00380
```

4.62.2.29 template<class DT, class WT> int sdGraph< DT, WT >::Output (ostream & *os*) const [inline]

Definition at line 604 of file sdgraph.inc.

```
00605   {
00606     // Print the data, ID, and the number of outgoing/incoming edges
00607     // for each vertex.
00608     //
00609     os << V_BEGIN_DELIM << '\n';
00610     for(int v=0; v<NumV; ++v)
00611         os << '\t' << v << '\t' << Data[v] << '\n';
00612     os << V_END_DELIM << '\n';
00613
00614     // Print the edges of the vertices
00615     //
00616     os << E_BEGIN_DELIM << '\n';
00617     for(int v=0; v<NumV; ++v)
00618       {
00619         if(VA[v]) VA[v]->PrintOutEdges(os);
00620       }
00621
```

4.62.2.30 template<class DT, class WT> int sdGraph< DT, WT >::OutputBinary (ostream & *os*) const [inline]

Definition at line 884 of file sdgraph.inc.

```
{
  // Write "NumV":
  //
  os <= NumV;

  // Write "NumE":
  //
  os <= NumE;

  // Write "VA":
  //
  if(VA && NumV>0)
    {
      for(int i=0; i<NumV; ++i)
        {
          // Write the number of out edges.
          //
          os <= VA[i]->NumO;

          // Write the number of in edges.
          //
          os <= VA[i]->NumI;

          // Write the out edges (id then weight).
          //
          if(VA[i]->OutE && VA[i]->NumO > 0)
            {
              for(int j=0; j<VA[i]->NumO; ++j)
                {
                  os <= VA[i]->OutE[j].V->ID;
                  os <= VA[i]->OutE[j].W;
                }
            }

          // Write the in edges (id then weight).
          //
          if(VA[i]->InE && VA[i]->NumI > 0)
            {
              for(int j=0; j<VA[i]->NumI; ++j)
                {
                  os <= VA[i]->InE[j].V->ID;
                  os <= VA[i]->InE[j].W;
                }
            }
        }
    }

  // Write "Data":
  //
  if(Data && NumV>0)
    {
      for(int i=0; i<NumV; ++i)
          os <= Data[i];
    }

```

4.62.2.31 template<class DT, class WT> ostream & sdGraph< DT, WT >::PrintDotty (ostream & *os*, int *edges* = 0) const [inline]

Definition at line 626 of file sdgraph.inc.

Referenced by ConstraintManager::PrintGraph().

```
{
  int to;
  WT w;

  os << "digraph G {\n";
  os << "\tcenter = true;\n";
  os << "\tratio = compress;\n";
  os << "\tsize =\"7.5,10\";\n";
  // Define all the labels for the vertices:
  for(int i=0; i<NumV; ++i)
    {
      // GRAPE customization
      os << '\t' << i+1 << " [label = \"" << Data[i]->Label()  << "\"];\n";
      //      os << '\t' << i+1 << " [label = \"" << Data[i]   << "\"];\n";
    }

  // Print all the edges:
  for(int i=0; i<NumV; ++i)
    {
      for(int j=0; j<VA[i]->NumO; ++j)
        {
          to = VA[i]->OutE[j].V->ID;
          w  = VA[i]->OutE[j].W;
          os << '\t' << i+1 << " -> " << to+1;
          if(edges)
              os << " [label=\"" << w << "\"]";
          os << ";\n";
        }
    }
  os << "}
```

4.62.2.32 template<class DT, class WT> void sdGraph< DT, WT >::Relax (int *u*, int *v*, jVector & *D*, int ∗ *P*, int *rev*, int *wc*) const [inline, private]

Definition at line 18 of file sdgalg.inc.

```
{
  WT w = (wc==0) ? WT(1.0) : (
              (wc>0)  ? (!rev ? GetWeight(u,v,0) : GetWeight(v,u,0)) :
                        (!rev ? GetWeight(u,v,1) : GetWeight(v,u,1))
                              );
  if(D[v] > D[u] + ((double)w))
    {
      D[v] = D[u] + ((double)w);
      if(P) P[v] = u;
    }
```

4.62.2.33 template<class DT, class WT> SDG_SetSetInt sdGraph< DT, WT >::SCC () const [inline]

Definition at line 126 of file sdgalg.inc.

Referenced by NumSCC(), SCCGraph(), and StronglyConnectedComponents().

```
{
  SDG_SetSetInt scc;
  SDG_SetInt component;
  int u, * finish=0, * invfin = 0;
```

```
00131
00132     if(NumV<1)
00133         return scc;
00134
00135     if(!(finish = new int[NumV]))
00136         raise(SIGQUIT);
00137     if(!(invfin = new int[NumV]))
00138         raise(SIGQUIT);
00139
00140     DFTimeStamps(0,finish,0);
00141
00142     for(u=0; u<NumV; ++u)
00143         invfin[ finish[u] ] = u;
00144
00145     // Color all the vertices white to mark them as unvisited:
00146     //
00147     sdg_color color[NumV];
00148     for(u=0; u<NumV; ++u)
00149         color[u] = sdg_white;
00150
00151     for(int i=NumV-1; i>=0; --i)
00152       {
00153         u = invfin[i];
00154         if(color[u] == sdg_white)
00155           {
00156             DF_Visit(u, color, 0, 0, 0, &component, 1);
00157             // scc.Add(component.UniqueCopy());
00158             scc.insert(component);
00159             component.clear();
00160           }
00161       }
00162
```

4.62.2.34 template<class DT, class WT> sdGraph< DT, WT > sdGraph< DT, WT >::SCCGraph () const [inline]

Definition at line 172 of file sdgalg.inc.

```
00173   {
00174     sdGraph<DT,WT> sccgraph;
00175     SDG_SetSetInt sccset = SCC();
00176     SDG_SetSetInt::const_iterator it1;
00177     SDG_SetInt::const_iterator    it2;
00178     int * v2c = 0;              // Mapping from vertex id to component number.
00179     int * numoe = 0;            // Number of outbound edges for each component.
00180     int * numie = 0;            // Number of inbound edges for each component.
00181     int ** OutEdge=0;           // Adjacency matrix for outbound edges.
00182     int ** InEdge=0;            // Adjacency matrix for inbound edges.
00183     int u,v,e;                  // Vertex id's and edge iterator.
00184     int c=0;                    // Increments through components.
00185
00186     // cerr << "NumV = " << NumV << '\n';
00187     // cerr << "NumC = " << sccset.Number() << '\n';
00188     // cerr << "NumC = " << sccset.size() << '\n';
00189     // raise(SIGQUIT);
00190
00191     if(NumV < 1)
00192         return sccgraph;
00193
00194     // if(sccset.Number() == 1)
00195     if(sccset.size() == 1)
00196       {
```

```
          sccgraph.NumV = 1;
          if(!(sccgraph.VA = new sdVertex<WT>*[1]) ||
             !(sccgraph.VA[0] = new sdVertex<WT>))
              raise(SIGQUIT);
          if(!(sccgraph.Data = new DT[1]))
              raise(SIGQUIT);
          sccgraph.VA[0]->ID = 0;
          sccgraph.Data[0] = Data[0];
          return sccgraph;
        }

      // sccgraph.NumV = sccset.Number(); // Number of strongly conn. components.
      sccgraph.NumV = sccset.size();      // Number of strongly conn. components.

      // Allocate space.
      if(!(v2c = new int[NumV]))
          raise(SIGQUIT);
      if(!(numoe = new int[sccgraph.NumV]))
          raise (SIGQUIT);
      if(!(numie = new int[sccgraph.NumV]))
          raise (SIGQUIT);
      if(!(sccgraph.VA = new sdVertex<WT>*[sccgraph.NumV]))
          raise(SIGQUIT);
      if(!(sccgraph.Data = new DT[sccgraph.NumV]))
          raise(SIGQUIT);
      if(!(OutEdge = new int*[sccgraph.NumV]))
          raise(SIGQUIT);
      if(!(InEdge = new int*[sccgraph.NumV]))
          raise(SIGQUIT);

      // Initialize the adjacency matrices.
      for(u=0; u<sccgraph.NumV; ++u)
        {
          if(!(OutEdge[u] = new int[sccgraph.NumV]))
              raise(SIGQUIT);
          if(!(InEdge[u] = new int[sccgraph.NumV]))
              raise(SIGQUIT);

          for(v=0; v<sccgraph.NumV; ++v)
            {
              OutEdge[u][v] = 0;
              InEdge[u][v]  = 0;
            }
          numoe[u] = 0;
          numie[u] = 0;
        }

      // Initialize the mapping from vertices to components.
      for(u=0; u<NumV; ++u)
          v2c[u] = -1;

      // Introduce vertices into the SCC-Graph.
      c=0;
      int got;
      for(it1 = sccset.begin(); it1!=sccset.end(); ++it1)
        {
          if(!(sccgraph.VA[c] = new sdVertex<WT>))
              raise(SIGQUIT);
          sccgraph.VA[c]->ID = c;

          // Record the vertices that belong to this component.
          got = 0;
          for(it2 = ($\ast$it1).begin(); it2!=($\ast$it1).end(); ++it2)
            {
              v2c[*it2] = c;
```

```

        // Copy one of the vertices' data into the component graph.
        if(!got)
          {
            sccgraph.Data[c] = Data[*it2];
            got = 1;
          }
      }
    ++c;
  }

// Make sure the mapping from vertices to components is complete:
//
for(u=0; u<NumV; ++u)
    if(v2c[u] == -1)
        raise(SIGQUIT);

// Create the adjacency matrices.
for(u=0; u<NumV; ++u)
  {
    // Find out which component this vertex belongs to.
    c = v2c[u];

    // Record the outbound component edges in the adjacency matrix.
    for(e=0; e<VA[u]->NumO; ++e)
      {
        v = VA[u]->OutE[e].V->ID;
        if((v2c[v] != c) && !OutEdge[c][v2c[v]])
          {
            OutEdge[c][v2c[v]] = 1;
            ++numoe[c];
          }
      }

    // Record the inbound component edges in the adjacency matrix.
    for(e=0; e<VA[u]->NumI; ++e)
      {
        v = VA[u]->InE[e].V->ID;
        if((v2c[v] != c) && !InEdge[c][v2c[v]])
          {
            InEdge[c][v2c[v]] = 1;
            ++numie[c];
          }
      }
  }

// Make the edges in the component graph.
for(u=0; u<sccgraph.NumV; ++u)
  {
    sccgraph.VA[u]->NumO = numoe[u];
    sccgraph.VA[u]->NumI = numie[u];

    // Create space for the edges.
    if((numoe[u]>0) && !(sccgraph.VA[u]->OutE = new sdEdge<WT>[numoe[u]]))
        raise(SIGQUIT);
    if((numie[u]>0) && !(sccgraph.VA[u]->InE = new sdEdge<WT>[numie[u]]))
        raise(SIGQUIT);

    // Read which edges to introduce from the adjacency matrix.
    c = 0;
    for(v=0; v<sccgraph.NumV; ++v)
      {
        if(OutEdge[u][v])
          {
            sccgraph.VA[u]->OutE[c].V = sccgraph.VA[v];
```

```
              sccgraph.VA[u]->OutE[c].W = 1.0;
              ++c;
            }
        }
      c = 0;
      for(v=0; v<sccgraph.NumV; ++v)
        {
          if(InEdge[u][v])
            {
              sccgraph.VA[u]->InE[c].V = sccgraph.VA[v];
              sccgraph.VA[u]->InE[c].W = 1.0;
              ++c;
            }
        }
    }

  delete[] v2c;
  delete[] numoe;
  delete[] numie;
  for(u=0; u<sccgraph.NumV; ++u)
    {
      delete[] OutEdge[u];
      delete[] InEdge[u];
    }
```

4.62.2.35 template<class DT, class WT> SDG_SetSetInt sdGraph< DT, WT >::StronglyConnectedComponents () const [inline]

Definition at line 168 of file sdgalg.inc.

```
{ return SCC(); }
```

4.62.2.36 template<class DT, class WT> sdGraph< DT, WT > sdGraph< DT, WT >::SubGraph (const SDG_SetInt & *vertset*) const [inline]

Definition at line 476 of file sdgraph.inc.

```
{
  sdGraph<DT,WT> sub;
  int * subid=0;              // Maps id's from this graph to the subgraph.
  int * vertex=0;             // Array of verts (copied from vertset).
  int u,v,e,i,nbr,numv;
  // SDG_SetInt::ConstIterator it;
  SDG_SetInt::const_iterator it;

  if(NumV<1) return sub;

  // Filter out any bad vertex indices from the set:
  numv=0;
  // for(it = vertset.GetConstIterator(); it; ++it)
  for(it = vertset.begin(); it!=vertset.end(); ++it)
      if(($\ast$it)>=0 && ($\ast$it)<NumV)
          ++numv;

  if(numv<1) return sub;

  // Make space for the vertices and the data.
  if(!(sub.VA = new sdVertex<WT>*[numv]))
```

```
      { cerr << "sdGraph::SubGraph#1\n"; raise(SIGQUIT); }
    if(!(sub.Data = new DT[numv]))
      { cerr << "sdGraph::SubGraph#2\n"; raise(SIGQUIT); }

    sub.NumV = numv;

    // Copy the vertices into a static array.
    if(!(vertex = new int[sub.NumV]))
      { cerr << "sdGraph::SubGraph#3\n"; raise(SIGQUIT); }
    u=0;
    // for(it.Reset(); it; ++it)
    for(it = vertset.begin(); it!=vertset.end(); ++it)
      {
        v = *it;
        if(v>=0 && v<NumV)
            vertex[u++] = v;
      }

    // Sort these vertices to make sure in increasing order.
    QuickSort(vertex,sub.NumV);

    if(!(subid = new int[NumV]))
      { cerr << "sdGraph::SubGraph#4\n"; raise(SIGQUIT); }

    for(v=0; v<NumV; ++v)       // Mark the id's as not existing in subgraph
        subid[v] = -1;

    // Determine the vertices from this graph to be included in the subgraph.
    for(u=0; u<sub.NumV; ++u)
      {
        v = vertex[u];
        if(!(sub.VA[u] = new sdVertex<WT>))
          { cerr << "sdGraph::SubGraph#5\n"; raise(SIGQUIT); }
        sub.VA[u]->ID = u;
        subid[v] = u;

        // Copy the data in.
        sub.Data[u] = Data[v];
      }

    // Introduce the edges into the subgraph.
    for(u=0; u<sub.NumV; ++u)
      {
        v = vertex[u];

        // Determine the number of outbound edges for vertex u.
        sub.VA[u]->NumO = 0;
        for(e=0; e<VA[v]->NumO; ++e)
          {
            nbr = subid[ VA[v]->OutE[e].V->ID ];
            if(nbr != -1)
                ++(sub.VA[u]->NumO);
          }

        // Determine the number of inbound edges for vertex u.
        sub.VA[u]->NumI = 0;
        for(e=0; e<VA[v]->NumI; ++e)
          {
            nbr = subid[ VA[v]->InE[e].V->ID ];
            if(nbr != -1)
                ++(sub.VA[u]->NumI);
          }

        // Allocate space for the edges.
        if(sub.VA[u]->NumO>0 &&
```

```
          !(sub.VA[u]->OutE = new sdEdge<WT>[sub.VA[u]->NumO]))
            { cerr << "sdGraph::SubGraph#6\n"; raise(SIGQUIT); }
        if(sub.VA[u]->NumI>0 &&
          !(sub.VA[u]->InE = new sdEdge<WT>[sub.VA[u]->NumI]))
            { cerr << "sdGraph::SubGraph#7\n"; raise(SIGQUIT); }

        // Copy the outbound edges.
        i=0;
        for(e=0; e<VA[v]->NumO; ++e)
          {
            nbr = subid[ VA[v]->OutE[e].V->ID ];
            if(nbr != -1)
              {
                sub.VA[u]->OutE[i].V = sub.VA[nbr];
                sub.VA[u]->OutE[i].W = VA[v]->OutE[e].W;
                ++i;
                ++(sub.NumE);
              }
          }

        // Copy the inbound edges.
        i=0;
        for(e=0; e<VA[v]->NumI; ++e)
          {
            nbr = subid[ VA[v]->InE[e].V->ID ];
            if(nbr != -1)
              {
                sub.VA[u]->InE[i].V = sub.VA[nbr];
                sub.VA[u]->InE[i].W = VA[v]->InE[e].W;
                ++i;
              }
          }
        cerr << "]\n";
      }

```

4.62.2.37 template<class DT, class WT> SDG_IntList ** sdGraph< DT, WT >::TravLists (int *src*, int *k*, KAdjMat * *P* = 0) const [inline]

Definition at line 706 of file sdgraph.inc.

Referenced by begin_path().

```
  {
    if(NumV<1 || k<1 || src<0 || src>=NumV) return 0;

    DLList<int> ** T;   // The traversal lists, originating at the source.
    int alloc = 0;      // Keeps track if we allocate space for P.

    if(!(T = new DLList<int>*[k-1])) return 0;
    for(int i=0; i<k; ++i)
      {
        if(!(T[i] = new DLList<int>[NumV]))
          {
            for(int j=0; j<i; ++j)
                delete[] T[j];
            delete[] T;
            return 0;
          }
      }

    if(!P)
      // Have to compute the the path matrices (0...k-1):
```

```
      {
        if(!(P = new KAdjMat[k]))
          { delete[] T; return 0; }
        P[0] = AdjMatrix();

        for(int i=1; i<k; ++i) P[i] = P[i-1]*P[0];
        alloc = 1;
      }

    ABS a;
    for(int i=0; i<k-1; ++i)
      {
        for(int j=0; j<NumV; ++j)
          {
            a = P[i][src]&P[0](j);
            for(int l=0; l<NumV; ++l)
              {
                if(a.IsSet((uint) l))
                    T[i][j].AddToBack(l);
              }
          }
      }
```

4.62.2.38 template<class DT, class WT> KAdjMat ∗ sdGraph< DT, WT >::TravMatrices (int *src*, int *k*, KAdjMat ∗ *P* = 0) const [inline]

Definition at line 675 of file sdgraph.inc.

```
  {
    if(NumV<1 || k<1 || src<0 || src>=NumV) return 0;

    KAdjMat * T;        // The traversal matrices, originating at the source.
    int alloc = 0;      // Keeps track if we allocate space for P.

    KAdjMat::DefaultSize(NumV);
    if(!(T = new KAdjMat[k-1])) return 0;
    if(!P)
      // Have to compute the the path matrices (0...k-1):
      {
        if(!(P = new KAdjMat[k]))
          { delete[] T; return 0; }
        P[0] = AdjMatrix();

        for(int i=1; i<k; ++i) P[i] = P[i-1]*P[0];
        alloc = 1;
      }
```

4.62.2.39 template<class DT, class WT> int sdGraph< DT, WT >::ValidData (int *i*) const [inline, private]

Definition at line 372 of file sdgraph.inc.

Referenced by GetData().

```
  { return (Data && i>=0 && i<NumV); }
```

4.62.2.40 template<class DT, class WT> int sdGraph< DT, WT >::ValidVertex (int *i*) const [inline, private]

Definition at line 368 of file sdgraph.inc.

Referenced by Degree(), GetVertex(), GetWeight(), InDegree(), and OutDegree().

```
{ return (VA && i>=0 && i<NumV && VA[i]); }

```

4.62.2.41 template<class DT, class WT = double> bf_iterator sdGraph<DT, WT>::begin_bf (int *vndx* = 0, int *rev* = 0) [inline]

Definition at line 210 of file sdgraph.h.

```
  { return bf_iterator(this, vndx, rev); }

```

4.62.2.42 template<class DT, class WT = double> df_iterator sdGraph<DT, WT>::begin_df (int *vndx* = 0, int *rev* = 0) [inline]

Definition at line 208 of file sdgraph.h.

```
  { return df_iterator(this, vndx, rev); }

```

4.62.2.43 template<class DT, class WT = double> neighbor_iterator sdGraph<DT, WT>::begin_neighbor (int *vndx* = 0, int *rev* = 0) [inline]

Definition at line 206 of file sdgraph.h.

Referenced by ConstraintManager::getParents().

```
  { return neighbor_iterator(this, vndx, rev); }

```

4.62.2.44 template<class DT, class WT = double> path_iterator sdGraph<DT, WT>::begin_path (int *src*, int *tgt*, int *len*, DLList< int > ∗const ∗ *tl* = 0, int(∗ *filter*)(const sdGraph< DT, WT > &graph, const int ∗path, int ndx, int pathlen, void ∗arbitrary_criteria) = 0, void ∗ *criteria* = 0) [inline]

Definition at line 212 of file sdgraph.h.

```
  {
    if(tl)
    return path_iterator((const sdGraph<DT,WT>*) this,
                         src, tgt,
                         len, tl, filter, criteria);
    path_iterator pi;

    //
    DLList<int> * const * trav = TravLists(src,len);
    pi   = path_iterator((const sdGraph<DT,WT>*) this, src, tgt,
```

```
                        len, trav, filter, criteria);

    for(int i=0; i<len-1; ++i) delete[] trav[i];
    delete[] trav;
    //
    return pi;
```

4.62.2.45 template<class DT, class WT> WT & sdGraph< DT, WT >::operator() (int *i*, int *j*, int *alt* = 0) `[inline]`

Definition at line 472 of file sdgraph.inc.

```
{ return GetWeight(i,j,alt); }
```

4.62.2.46 template<class DT, class WT> const WT & sdGraph< DT, WT >::operator() (int *i*, int *j*, int *alt* = 0) const `[inline]`

Definition at line 468 of file sdgraph.inc.

```
{ return GetWeight(i,j,alt); }
```

4.62.2.47 template<class DT, class WT> DT & sdGraph< DT, WT >::operator() (int *i*) `[inline]`

Definition at line 444 of file sdgraph.inc.

```
{ return GetData(i); }
```

4.62.2.48 template<class DT, class WT> const DT & sdGraph< DT, WT >::operator() (int *i*) const `[inline]`

Definition at line 440 of file sdgraph.inc.

```
{ return GetData(i); }
```

4.62.2.49 template<class DT, class WT> const sdGraph< DT, WT > & sdGraph< DT, WT >::operator= (const ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 304 of file sdgraph.inc.

```
{ Copy(ddg, true); return *this; }
```

4.62.2.50 template<class DT, class WT> const sdGraph< DT, WT > & sdGraph< DT, WT >::operator= (const sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 300 of file sdgraph.inc.

```
{ Copy(sdg); return *this; }

```

4.62.2.51 template<class DT, class WT> sdVertex< WT > & sdGraph< DT, WT >::operator[] (int *i*) `[inline]`

Definition at line 420 of file sdgraph.inc.

```
{ return GetVertex(i); }

```

4.62.2.52 template<class DT, class WT> const sdVertex< WT > & sdGraph< DT, WT >::operator[] (int *i*) const `[inline]`

Definition at line 416 of file sdgraph.inc.

```
{ return GetVertex(i); }

```

4.62.3 Friends And Related Function Documentation

4.62.3.1 template<class DT, class WT = double> class ddGraph `[friend]`

Definition at line 75 of file sdgraph.h.

4.62.3.2 template<class DT, class WT = double> ostream& operator (ostream & *os*, const sdGraph< DT, WT > & *sdg*) `[friend]`

4.62.3.3 template<class DT, class WT = double> istream& operator>= (istream & *is*, sdGraph< DT, WT > & *sdg*) `[friend]`

Definition at line 944 of file sdgraph.inc.

```
{ sdg.InputBinary(is); return is; }

```

4.62.4 Member Data Documentation

4.62.4.1 template<class DT, class WT = double> DT * sdGraph<DT, WT>::Data `[private]`

Definition at line 296 of file sdgraph.h.

4.62.4.2 template<class DT, class WT = double> int sdGraph<DT, WT>::NumE `[private]`

Definition at line 290 of file sdgraph.h.

4.62.4.3 template<class DT, class WT = double> int sdGraph<DT, WT>::NumV [private]

Definition at line 289 of file sdgraph.h.

4.62.4.4 template<class DT, class WT = double> sdVertex< WT > ** sdGraph<DT, WT>::VA [private]

Definition at line 293 of file sdgraph.h.

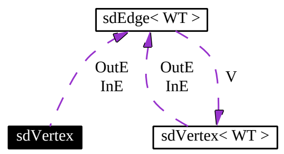
The documentation for this class was generated from the following files:

- **sdgraph.h**
- **sdgalg.inc**
- **sdgraph.inc**

4.63 sdVertex Class Template Reference

`#include <sdvertex.h>`

Collaboration diagram for sdVertex:



Public Methods

- **sdVertex** ()
- **sdVertex** (const sdVertex< WT > &sdv)
- **sdVertex** (int id, int num_out, int num_in)
- **~sdVertex** ()
- int **OutDegree** () const
- int **InDegree** () const
- int **Degree** () const
- void **SortEdges** ()
- void **PrintOutEdges** (ostream &os) const
- void **PrintInEdges** (ostream &os) const
- void **PrintEdges** (ostream &os) const
- const sdVertex<WT>& **operator=** (const sdVertex< WT > &sdv)
- int **Compare** (const sdVertex< WT > &v) const
- **MAKECOMPARISONS_WITH_COMPARE** (sdVertex< WT >)
- const sdVertex<WT>& **GetOutNeighbor** (int i) const
- sdVertex<WT>& **GetOutNeighbor** (int i)
- const sdVertex<WT>& **operator**[] (int i) const
- sdVertex<WT>& **operator**[] (int i)
- const sdVertex<WT>& **GetInNeighbor** (int i) const
- sdVertex<WT>& **GetInNeighbor** (int i)
- const sdVertex<WT>& **operator()** (int i) const
- sdVertex<WT>& **operator()** (int i)
- int **GetOutEdgeIndex** (const sdVertex< WT > &to) const
- int **GetOutEdgeIndex** (int to_id) const
- int **GetInEdgeIndex** (const sdVertex< WT > &from) const
- int **GetInEdgeIndex** (int from_id) const
- int **HasEdgeTo** (const sdVertex< WT > &to) const
- int **HasEdgeFrom** (const sdVertex< WT > &from) const
- const **sdEdge**<WT>& **GetOutEdge** (int i) const
- **sdEdge**<WT>& **GetOutEdge** (int i)
- const **sdEdge**<WT>& **GetInEdge** (int i) const
- **sdEdge**<WT>& **GetInEdge** (int i)
- const WT& **GetOutWeight** (int i) const
- WT& **GetOutWeight** (int i)
- const WT& **GetInWeight** (int i) const
- WT& **GetInWeight** (int i)

Public Attributes

- int **ID**
- int **NumO**
- int **NumI**
- **sdEdge**<WT>∗ **OutE**
- **sdEdge**<WT>∗ **InE**

Friends

- ostream& **operator** (ostream &os, const sdVertex< WT > &vert)
- int **Connected** (const sdVertex< WT > &v1, const sdVertex< WT > &v2)

template<class WT = double> class sdVertex

4.63.1 Constructor & Destructor Documentation

4.63.1.1 template<class WT = double> sdVertex<WT>::sdVertex<WT> ()

4.63.1.2 template<class WT = double> sdVertex<WT>::sdVertex<WT> (const sdVertex< WT > & *sdv*)

4.63.1.3 template<class WT = double> sdVertex<WT>::sdVertex<WT> (int *id*, int *num_out*, int *num_in*)

4.63.1.4 template<class WT> sdVertex< WT >::∼sdVertex<WT> () `[inline]`

Definition at line 81 of file sdvertex.inc.

```
00082   { delete[] OutE; delete[] InE; }
00083
```

4.63.2 Member Function Documentation

4.63.2.1 template<class WT> int sdVertex< WT >::Compare (const sdVertex< WT > & *v*) const `[inline]`

Definition at line 170 of file sdvertex.inc.

```
00171   { return (ID - v.ID); }
00172
```

4.63.2.2 template<class WT> int sdVertex< WT >::Degree () const `[inline]`

Definition at line 93 of file sdvertex.inc.

```
00094   { return OutDegree() + InDegree(); }
00095
```

4.63.2.3 template<class WT> sdEdge< WT > & sdVertex< WT >::GetInEdge (int *i*) [inline]

Definition at line 260 of file sdvertex.inc.

```
{ return InE[i]; }

```

4.63.2.4 template<class WT> const sdEdge< WT > & sdVertex< WT >::GetInEdge (int *i*) const [inline]

Definition at line 256 of file sdvertex.inc.

```
{ return InE[i]; }

```

4.63.2.5 template<class WT> int sdVertex< WT >::GetInEdgeIndex (int *from_id*) const [inline]

Definition at line 221 of file sdvertex.inc.

```
{
  if(!NumI) return -1;

  sdEdge<WT> edge; edge.V = new sdVertex<WT>(from_id, 0, 0);
  int index = BinSearch(InE, NumI, edge);
  delete edge.V;
  return index;

```

4.63.2.6 template<class WT> int sdVertex< WT >::GetInEdgeIndex (const sdVertex< WT > & *from*) const [inline]

Definition at line 232 of file sdvertex.inc.

Referenced by HasEdgeFrom().

```
{ return GetInEdgeIndex(from.ID); }

```

4.63.2.7 template<class WT> sdVertex< WT > & sdVertex< WT >::GetInNeighbor (int *i*) [inline]

Definition at line 194 of file sdvertex.inc.

```
{ return *(InE[i].V); }

```

4.63.2.8 template<class WT> const sdVertex< WT > & sdVertex< WT >::GetInNeighbor (int *i*) const `[inline]`

Definition at line 190 of file sdvertex.inc.

Referenced by operator()().

```
{ return *(InE[i].V); }

```

4.63.2.9 template<class WT> WT & sdVertex< WT >::GetInWeight (int *i*) `[inline]`

Definition at line 276 of file sdvertex.inc.

```
{ return InE[i].W; }

```

4.63.2.10 template<class WT> const WT & sdVertex< WT >::GetInWeight (int *i*) const `[inline]`

Definition at line 272 of file sdvertex.inc.

Referenced by sdGraph::GetWeight().

```
{ return InE[i].W; }

```

4.63.2.11 template<class WT> sdEdge< WT > & sdVertex< WT >::GetOutEdge (int *i*) `[inline]`

Definition at line 252 of file sdvertex.inc.

```
{ return OutE[i]; }

```

4.63.2.12 template<class WT> const sdEdge< WT > & sdVertex< WT >::GetOutEdge (int *i*) const `[inline]`

Definition at line 248 of file sdvertex.inc.

```
{ return OutE[i]; }

```

4.63.2.13 template<class WT> int sdVertex< WT >::GetOutEdgeIndex (int *to_id*) const `[inline]`

Definition at line 206 of file sdvertex.inc.

```
00207 {
00208   if(!NumO) return -1;
00209
00210   sdEdge<WT> edge; edge.V = new sdVertex<WT>(to_id, 0, 0);
00211   int index = BinSearch(OutE, NumO, edge);
00212   delete edge.V;
00213   return index;
00214
```

4.63.2.14 template<class WT> int sdVertex< WT >::GetOutEdgeIndex (const sdVertex< WT > & *to*) const `[inline]`

Definition at line 217 of file sdvertex.inc.

Referenced by HasEdgeTo().

```
00218 { return GetOutEdgeIndex(to.ID); }
00219
```

4.63.2.15 template<class WT> sdVertex< WT > & sdVertex< WT >::GetOutNeighbor (int *i*) `[inline]`

Definition at line 178 of file sdvertex.inc.

```
00179 { return *(OutE[i].V); }
00180
```

4.63.2.16 template<class WT> const sdVertex< WT > & sdVertex< WT >::GetOutNeighbor (int *i*) const `[inline]`

Definition at line 174 of file sdvertex.inc.

Referenced by operator[]().

```
00175 { return *(OutE[i].V); }
00176
```

4.63.2.17 template<class WT> WT & sdVertex< WT >::GetOutWeight (int *i*) `[inline]`

Definition at line 268 of file sdvertex.inc.

```
00269 { return OutE[i].W; }
00270
```

4.63.2.18 template<class WT> const WT & sdVertex< WT >::GetOutWeight (int *i*) const `[inline]`

Definition at line 264 of file sdvertex.inc.

Referenced by sdGraph::GetWeight().

```
00265 { return OutE[i].W; }
00266
```

4.63.2.19 template<class WT> int sdVertex< WT >::HasEdgeFrom (const sdVertex< WT > & *from*) const `[inline]`

Definition at line 240 of file sdvertex.inc.

Referenced by Connected().

```
00241   { return (GetInEdgeIndex(from.ID) != -1); }
00242
```

4.63.2.20 template<class WT> int sdVertex< WT >::HasEdgeTo (const sdVertex< WT > & *to*) const `[inline]`

Definition at line 236 of file sdvertex.inc.

Referenced by Connected(), and sdGraph::HasEdge().

```
00237   { return (GetOutEdgeIndex(to.ID) != -1); }
00238
```

4.63.2.21 template<class WT> int sdVertex< WT >::InDegree () const `[inline]`

Definition at line 89 of file sdvertex.inc.

Referenced by Degree().

```
00090   { return NumI; }
00091
```

4.63.2.22 template<class WT = double> sdVertex<WT>::MAKECOMPARISONS_WITH_COMPARE (sdVertex< WT >)

4.63.2.23 template<class WT> int sdVertex< WT >::OutDegree () const `[inline]`

Definition at line 85 of file sdvertex.inc.

Referenced by Degree().

```
00086   { return NumO; }
00087
```

4.63.2.24 template<class WT> void sdVertex< WT >::PrintEdges (ostream & *os*) const `[inline]`

Definition at line 118 of file sdvertex.inc.

```
00119   { PrintOutEdges(os); PrintInEdges(os); }
00120
```

4.63.2.25 template<class WT> void sdVertex< WT >::PrintInEdges (ostream & *os*) const [inline]

Definition at line 111 of file sdvertex.inc.

Referenced by PrintEdges().

```
{
  for(int i=0; i<NumI; ++i)
      os << '\t' << InE[i].V->ID << '\t' << ID << '\t' << InE[i].W << '\n';
```

4.63.2.26 template<class WT> void sdVertex< WT >::PrintOutEdges (ostream & *os*) const [inline]

Definition at line 104 of file sdvertex.inc.

Referenced by sdGraph::Output(), and PrintEdges().

```
{
  for(int i=0; i<NumO; ++i)
      os << '\t' << ID << '\t' << OutE[i].V->ID << '\t' << OutE[i].W << '\n';
```

4.63.2.27 template<class WT> void sdVertex< WT >::SortEdges () [inline]

Definition at line 97 of file sdvertex.inc.

Referenced by sdGraph::Copy().

```
{
  QuickSort(OutE, NumO);
  QuickSort(InE, NumI);
```

4.63.2.28 template<class WT> sdVertex< WT > & sdVertex< WT >::operator() (int *i*) [inline]

Definition at line 202 of file sdvertex.inc.

```
{ return GetInNeighbor(i); }
```

4.63.2.29 template<class WT> const sdVertex< WT > & sdVertex< WT >::operator() (int *i*) const [inline]

Definition at line 198 of file sdvertex.inc.

```
{ return GetInNeighbor(i); }
```

4.63.2.30 template<class WT> const sdVertex< WT > & sdVertex< WT >::operator= (const sdVertex< WT > & *sdv*) [inline]

Definition at line 126 of file sdvertex.inc.

```
{
  sdEdge<WT> * OutE_tmp = OutE, * InE_tmp = InE;

  if(this != &sdv)
    {
      if(sdv.NumO<1)
          OutE = 0;
      else if(!(OutE = new sdEdge<WT>[sdv.NumO]))
        {
          OutE = OutE_tmp;
          return *this;
        }
      if(sdv.NumI<1)
          InE = 0;
      else if(!(InE  = new sdEdge<WT>[sdv.NumI]))
        {
          delete[] OutE;
          OutE = OutE_tmp;
          InE  = InE_tmp;
          return *this;
        }
      delete[] OutE_tmp;
      delete[] InE_tmp;

      ID   = sdv.ID;
      NumO = sdv.NumO;
      NumI = sdv.NumI;

      for(int i=0; i<NumO; ++i)
        {
          OutE[i].V = sdv.OutE[i].V;
          OutE[i].W = sdv.OutE[i].W;
        }
      for(int i=0; i<NumI; ++i)
        {
          InE[i].V = sdv.InE[i].V;
          InE[i].W = sdv.InE[i].W;
        }
    }

```

4.63.2.31 template<class WT> sdVertex< WT > & sdVertex< WT >::operator[] (int *i*) [inline]

Definition at line 186 of file sdvertex.inc.

```
{ return GetOutNeighbor(i); }

```

4.63.2.32 template<class WT> const sdVertex< WT > & sdVertex< WT >::operator[] (int *i*) const [inline]

Definition at line 182 of file sdvertex.inc.

```
{ return GetOutNeighbor(i); }

```

4.63.3 Friends And Related Function Documentation

4.63.3.1 template<class WT = double> int Connected (const sdVertex< WT > & *v1*, const sdVertex< WT > & *v2*) [friend]

Definition at line 244 of file sdvertex.inc.

```
00245   { return (v1.HasEdgeTo(v2) || v1.HasEdgeFrom(v2)); }
00246
```

4.63.3.2 template<class WT = double> ostream& operator (ostream & *os*, const sdVertex< WT > & *vert*) [friend]

4.63.4 Member Data Documentation

4.63.4.1 template<class WT = double> int sdVertex<WT>::ID

Definition at line 52 of file sdvertex.h.

4.63.4.2 template<class WT = double> sdEdge< WT > ∗ sdVertex<WT>::InE

Definition at line 60 of file sdvertex.h.

4.63.4.3 template<class WT = double> int sdVertex<WT>::NumI

Definition at line 56 of file sdvertex.h.

4.63.4.4 template<class WT = double> int sdVertex<WT>::NumO

Definition at line 55 of file sdvertex.h.

4.63.4.5 template<class WT = double> sdEdge< WT > ∗ sdVertex<WT>::OutE

Definition at line 59 of file sdvertex.h.

The documentation for this class was generated from the following files:

- **sdvertex.h**
- **sdvertex.inc**

4.64 Segment Class Reference

Public Methods

- **Segment** ()
- **Segment** (double a, double b)
- ∼**Segment** ()
- const Segment& **operator=** (const Segment &s)

Public Attributes

- double **start**
- double **end**

Friends

- int **operator>** (const Segment &s1, const Segment &s2)
- int **operator<** (const Segment &s1, const Segment &s2)
- ostream& **operator<<** (ostream &os, const Segment &seg)

4.64.1 Constructor & Destructor Documentation

4.64.1.1 Segment::Segment () [inline]

Definition at line 23 of file segments.inc.

```
00023           {}
00024
```

4.64.1.2 Segment::Segment (double *a*, double *b*) [inline]

Definition at line 25 of file segments.inc.

```
00025                         {
00026     if (a<b){
00027       start = a;
00028       end = b;
00029     } else {
00030       start = b;
00031       end = a;
00032     }
00033
```

4.64.1.3 Segment::∼Segment () [inline]

Definition at line 35 of file segments.inc.

```
00036     {
00037
```

4.64.2 Member Function Documentation

4.64.2.1 const Segment & Segment::operator= (const Segment & *s*) [inline]

Definition at line 39 of file segments.inc.

```
{
  if (this != &s)
  {
    start = s.start;
    end = s.end;
    //      width = s.width;
  }
```

4.64.3 Friends And Related Function Documentation

4.64.3.1 int operator< (const Segment & *s1*, const Segment & *s2*) [friend]

Definition at line 62 of file segments.inc.

```
{
  // order on START value
  if(s1.start < s2.start) return 1;
  else if(s1.start > s2.start) return 0;
  // If the Segment have the same start position;
  if(s1.end < s2.end) return 1;
  else if(s1.end > s2.end) return 0;
  // If the Segment are the same
  return 0;
```

4.64.3.2 ostream & operator<< (ostream & *os*, const Segment & *seg*) [friend]

Definition at line 74 of file segments.inc.

```
{
  os << seg.start << " " << seg.end;// << " " << seg.width;
  return os;
```

4.64.3.3 int operator> (const Segment & *s1*, const Segment & *s2*) [friend]

Definition at line 50 of file segments.inc.

```
{
  // order on START value
  if(s1.start > s2.start) return 1;
  else if(s1.start < s2.start) return 0;
  // If the Segment have the same start position;
  if(s1.end > s2.end) return 1;
  else if(s1.end < s2.end) return 0;
  // If the Segment are the same
  return 0;
```

4.64.4 Member Data Documentation

4.64.4.1 double Segment::end

Definition at line 81 of file segments.inc.

4.64.4.2 double Segment::start

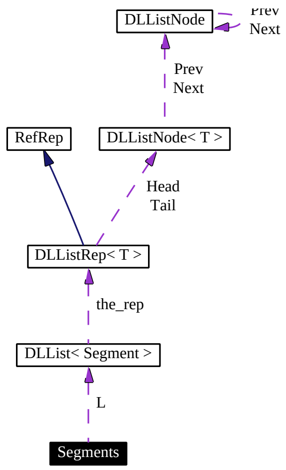
Definition at line 80 of file segments.inc.

The documentation for this class was generated from the following file:

- **segments.inc**

4.65 Segments Class Reference

Collaboration diagram for Segments:



Public Methods

- **Segments** ()
- ∼**Segments** ()
- double **GetNum** (void)
- **Segment GetSeg** (int j)
- void **Add** (double start, double end)
- void **Add** (**Segment** seg)
- void **QSortReduce** ()
- **Segment QSortReduceViol** (double testval)
- **Segment Pop** (void)
- int **Number** (void)
- void **Clear** ()

Private Methods

- bool **Overlap** (**Segment** &s1, **Segment** &s2)
- **Segment Join** (**Segment** &s1, **Segment** &s2)

Private Attributes

- **DLList<Segment> L**

4.65.1 Constructor & Destructor Documentation

4.65.1.1 Segments::Segments () `[inline]`

Definition at line 93 of file segments.inc.

```
00093             { }
00094
```

4.65.1.2 Segments::~Segments () [inline]

Definition at line 94 of file segments.inc.

```
00094                 { }
00095
```

4.65.2 Member Function Documentation

4.65.2.1 void Segments::Add (Segment *seg*) [inline]

Definition at line 109 of file segments.inc.

```
00109                        {
00110     L.AddToFront(seg);      // unsorted Segment list
00111
```

4.65.2.2 void Segments::Add (double *start*, double *end*) [inline]

Definition at line 104 of file segments.inc.

Referenced by gwNumerics::addLPERTSegments(), and gwNumerics::addVDWSegments().

```
00104                                     {
00105     Segment s(start,end);
00106     L.AddToFront(s);      // unsorted Segment list
00107
```

4.65.2.3 void Segments::Clear () [inline]

Definition at line 197 of file segments.inc.

```
00198     {L.Clear();}
00199
```

4.65.2.4 double Segments::GetNum (void) [inline]

Definition at line 96 of file segments.inc.

Referenced by gwNumerics::SetHCPosition2(), and gwNumerics::addLPERTSegments().

```
00096                       {
00097     return L.Number();
00098
```

4.65.2.5 Segment Segments::GetSeg (int *j*) [inline]

Definition at line 100 of file segments.inc.

Referenced by gwNumerics::SetHCPosition2(), and gwNumerics::addLPERTSegments().

```
00100                         {
00101     return L.Get(j);
00102
```

4.65.2.6 Segment Segments::Join (Segment & *s1*, Segment & *s2*) [inline, private]

Definition at line 211 of file segments.inc.

Referenced by QSortReduce().

```
00212     {
00213 
00214       Segment returnSegment;
00215       returnSegment.start = (s1.start>s2.start) ? s2.start : s1.start;
00216       returnSegment.end = (s1.end<s2.end) ? s2.end : s1.end;
00217       //      returnSegment.width = returnSegment.end - returnSegment.start;
00218       return returnSegment;
00219 
```

4.65.2.7 int Segments::Number (void) [inline]

Definition at line 194 of file segments.inc.

```
00195     {return(L.Number());}
00196 
```

4.65.2.8 bool Segments::Overlap (Segment & *s1*, Segment & *s2*) [inline, private]

Definition at line 202 of file segments.inc.

Referenced by QSortReduce().

```
00203     {
00204       if ((s1.start >= s2.start)&&(s1.start <= s2.end)) return true;
00205       if ((s2.start >= s1.start)&&(s2.start <= s1.end)) return true;
00206       if ((s1.end >= s2.start)&&(s1.end <= s2.end)) return true;
00207       if ((s2.end >= s1.start)&&(s2.end <= s1.end)) return true;
00208       return false;
00209 
```

4.65.2.9 Segment Segments::Pop (void) [inline]

Definition at line 191 of file segments.inc.

```
00192     {return(L.RemoveFromFront());}
00193 
```

4.65.2.10 void Segments::QSortReduce () [inline]

Definition at line 113 of file segments.inc.

Referenced by QSortReduceViol().

```
00113                     {
00114     int num = L.Number();
00115     if (num == 0) return;
00116     // Sort the list using another array
00117     Segment *segarray = new Segment[num];
```

```
  for (int i= 0;i<num;i++)
    segarray[i] = L.RemoveFromFront(); // Create array for quicksort
  RecQuickSort(segarray, 0, num-1);  // Sort it (from search-sort.h)

  L.Clear(); // sanity ck debug

  // Reduce as they are added back to the list.
  Segment TempSeg = segarray[0];
  int j = 1;
  // the next 9 lines are a gem. --jd
  while (j < num){
    if(Overlap(TempSeg,segarray[j])){
      TempSeg = Join(TempSeg,segarray[j++]);
    } else {
      L.AddToBack(TempSeg);
      TempSeg = segarray[j++];
    }
  }

```

4.65.2.11 Segment Segments::QSortReduceViol (double *testval*) [inline]

Definition at line 142 of file segments.inc.

Referenced by gwNumerics::SetHCPosition2().

```
00142                                           {
00143     // Returns segment that overlaps the testval.  This check REQUIES
00144     // that the seglist be sorted and reduced for efficient checking,
00145     // so this function begins with a call to QSortReduce.
00146 
00147     Segment testseg(testval,testval);
00148 
00149     QSortReduce();
00150 
00151 #ifdef JOB_VERBOSE2
00152     int num = L.Number();
00153     cerr << "final number segments = "<< num << "; "      ;
00154     for (int i=0;i<num;i++) cerr << "(" << L.Get(i) << ") " ;
00155     cerr << endl;
00156 #endif
00157 
00158     if (L.Number()>0){
00159       int segind = 0;
00160       testseg = L.Get(segind++);
00161 #ifdef JOB_VERBOSE2
00162       cerr << "walking up through segment: ("<< testseg <<") ;?test overlap "<<testval<< endl;
00163 #endif
00164       while (testseg.end < testval){
00165         if (segind >= L.Number()) break;
00166         testseg = L.Get(segind++);
00167       }
00168       // test for violation
00169       if ((testseg.start < testval) && (testseg.end > testval)){
00170 #ifdef JOB_VERBOSE2
00171     cerr << "Violating segment: ("<< testseg <<")"<< endl;
00172 #endif
00173 
00174         return testseg;
00175       }
00176       testseg.start = 0; // returns zero segment if ok
00177       testseg.end = 0;
00178     } else {
```

```
        // no segments at all in this class
        testseg.start = 0; // returns zero segment if ok
        testseg.end = 0;
    }

```

4.65.3 Member Data Documentation

4.65.3.1 DLList< Segment > Segments::L [private]

Definition at line 89 of file segments.inc.

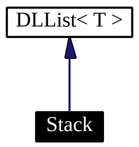
The documentation for this class was generated from the following file:

- **segments.inc**

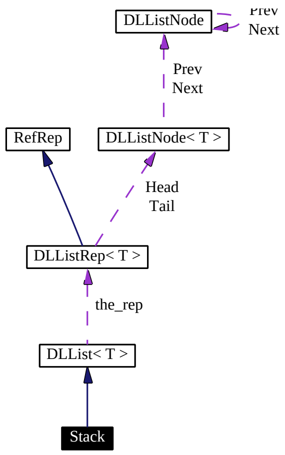
4.66 Stack Class Template Reference

`#include <stack.h>`

Inheritance diagram for Stack



Collaboration diagram for Stack:



Public Methods

- void **Push** (const T &data)
- void **Push** (const Stack< T > &stk)
- T **Pop** ()
- const T& **Peek** () const
- const Stack<T>& **operator+=** (const Stack< T > &stk)

Friends

- Stack<T> **operator+** (Stack< T > x, const Stack< T > &y)

template<class T> class Stack

4.66.1 Member Function Documentation

4.66.1.1 template<class T> const T & Stack<T>::Peek () const `[inline]`

Definition at line 22 of file stack.h.

Referenced by IdGen::Peek().

```
00022                                    { return Get(0); }
00023
```

4.66.1.2 template<class T> T Stack<T>::Pop () [inline]

Definition at line 21 of file stack.h.

Referenced by IdGen::Get(), and df_iterator::operator++().

```
00021                       { return RemoveFromFront(); }
00022
```

4.66.1.3 template<class T> void Stack<T>::Push (const Stack< T > & *stk*) [inline]

Definition at line 20 of file stack.h.

```
00020                                              { AddToFront(stk); }
00021
```

4.66.1.4 template<class T> void Stack<T>::Push (const T & *data*) [inline]

Definition at line 19 of file stack.h.

Referenced by IdGen::Free(), df_iterator::operator++(), and operator+=().

```
00019                                         { AddToFront(data); }
00020
```

4.66.1.5 template<class T> const Stack< T > & Stack<T>::operator+=<T> (const Stack< T > & *stk*) [inline]

Definition at line 23 of file stack.h.

```
00024                   { Push(stk); return *this; }
00025
```

4.66.2 Friends And Related Function Documentation

4.66.2.1 template<class T> Stack< T > operator+<T> (Stack< T > *x*, const Stack< T > & *y*) [friend]

Definition at line 26 of file stack.h.

```
00027                   { return x+=y; }
00028
```

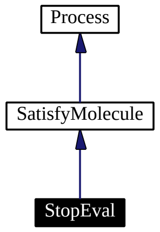
The documentation for this class was generated from the following file:

- **stack.h**

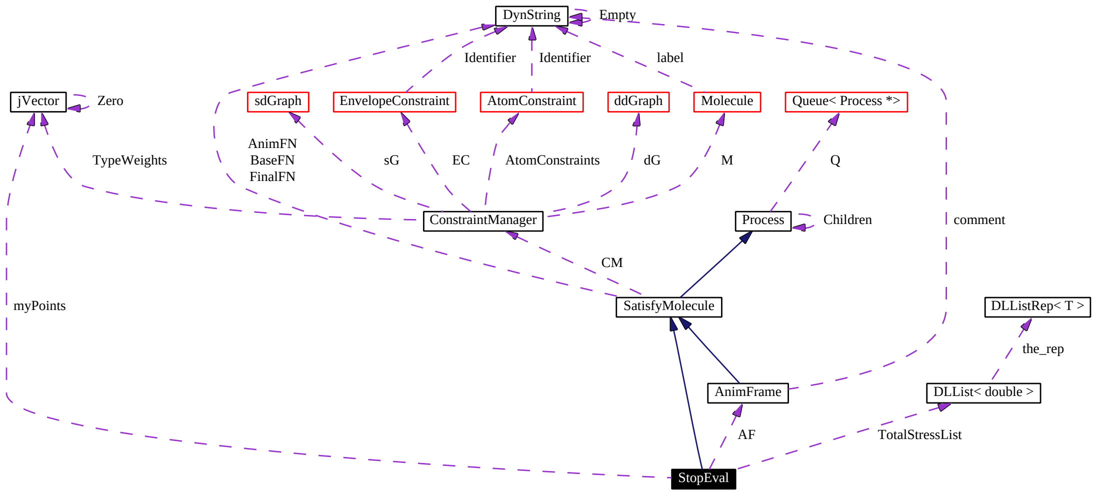
4.67 StopEval Class Reference

#include <jd_stop_eval.h>

Inheritance diagram for StopEval



Collaboration diagram for StopEval:



Public Methods

- **StopEval** (const StopEval &ca)
- **StopEval** (**ConstraintManager** &**CM**)
- ~**StopEval** ()
- int **run** ()
- int **check_stress_max_tot** (double, double)
- int **check_low_error** (double, double)
- void **cycle_header** (void)
- void **writebest** (void)

Static Public Attributes

- bool **Enable** = true
- bool **SolveOnly** = false
- bool **Rtp** = false

  Real time plotting output – single double outputs for plotting.

- bool **Quiet** = false

  run-time ajdustable flag to stop default output.

- bool **VerboseOn** = false

  run-time ajdustable flag to provide verbose output.

Private Attributes

- **AnimFrame**∗ **AF**
- ofstream∗ **of**
- **jVector**∗ **myPoints**
- double **LowErr**
- double **BestAveErr**
- double **BestMaxErr**

Static Private Attributes

- int **LookBack** = 50

  number of calls to StopEval to remember.

- **DLList**<double> **TotalStressList**
- double **MinTotalStress**
- double **MinofMaxStress**

  the overall min of the max on each list of stress calls on an iteration.

- double **MinofMaxAssocTotal**
- double **MinLookBackStress**
- double **MaxLookBackStress**
- double **MeanLookBackTotalStress**
- double **SumXX**

  Variables for linear regression these are maintained over the lookback calls to **StopEval::run**() (p. 493).

- double **SumXY**
- double **SumX**
- double **SumY**
- double **SumYY**
- double **Variance**
- int **NumCalls** = 0
- int **NumLprAdj** = 0
- int **NumLprLim** = 10
- int **LastLprAdj** = 0
- int **BestIter** = 0
- double **LookBackSlope**
- double **LookBackD**

4.67.1 Detailed Description

//

File: **jd_stop_eval.h**

Created by: Jon Dugan

Evaluates the molecule state to determine iteration exit conditions.

//

Definition at line 28 of file jd_stop_eval.h.

4.67.2 Constructor & Destructor Documentation

4.67.2.1 StopEval::StopEval (const StopEval & *se*)

Definition at line 35 of file jd_stop_eval.C.

```
  : SatisfyMolecule(*(se.CM))
{
  TotalStressList = se.TotalStressList;
  DIE(" Copy constructor for stop eval called ");
```

4.67.2.2 StopEval::StopEval (ConstraintManager & *cm_in*)

Definition at line 43 of file jd_stop_eval.C.

```
  : SatisfyMolecule(cm_in)
{

  AF = new AnimFrame(cm_in);
  of = 0;
  myPoints = new jVector [CM->numAtoms()];
  LowErr = INF;
  BestAveErr= INF;
  BestMaxErr = INF;
```

4.67.2.3 StopEval::~StopEval ()

Definition at line 55 of file jd_stop_eval.C.

```
{
  delete AF; AF = 0;
  if (of) delete of; of = 0;
  delete [] myPoints;
```

4.67.3 Member Function Documentation

4.67.3.1 int StopEval::check_low_error (double *max_err*, double *ave_err*)

Definition at line 62 of file jd_stop_eval.C.

Referenced by AtomBuildup::run().

```
00062                                                              {
00063   double err = 0.1 * ave_err + max_err;
00064   if (err < LowErr) {
00065     LowErr = err;
00066     BestIter = NumCalls;
00067     for (int i= 0; i < CM->numAtoms(); i ++){
00068       myPoints[i] = CM->getMolecule().getPosition(i); // save best positions
00069     }
00070     BestAveErr = ave_err;
00071     BestMaxErr = max_err;
00072     return 1;
00073   }
00074
```

4.67.3.2 int StopEval::check_stress_max_tot (double *max_test*, double *total_test*)

Definition at line 77 of file jd_stop_eval.C.

Referenced by AtomBuildup::run().

```
00078 {  // if either passes, stop
00079
00080   double max = -1*INF;
00081   double tot = 0;
00082   jVector currStressList = CM->getStressList();
00083
00084   for (int i =0; i <  currStressList.Length(); i++){
00085     if (max < currStressList[i]) max = currStressList[i];
00086     tot += currStressList[i];
00087   }
00088
```

4.67.3.3 void StopEval::cycle_header (void)

Definition at line 101 of file jd_stop_eval.C.

```
00101                               {
00102   cerr << "\nCycle   Avg Err    Max Err  Tot Err  %pert  Mean Total       Slope  Variance"
00103        << LookBack << "\n"
00104        << "-----   -------    -------  -------   ----  ----------       -----  -----------      "
00105        << endl;
00106
00107
```

4.67.3.4 int StopEval::run (void) [virtual]

Execute this process. Returns the exit status of the process (0 is success, non-zero is error).

Reimplemented from **Process** (p. 396).

Definition at line 109 of file jd_stop_eval.C.

```
00109                   {
00110 #ifdef JOB_VERBOSE
00111   cerr << "DIDJOB:\tStopEval::run()" << endl;
00112 #endif
```

```
  /*

//
//   // initialize max, mean and total stress
//   double maxStress = 0;
//   double totalStress = 0;
//   jVector currStressList = CM->getStressList();
//
//   for (int i =0; i <  currStressList.Length(); i++){
//     if (maxStress < currStressList[i]) maxStress = currStressList[i];
//     totalStress += currStressList[i];
//   }
//   double meanStress = totalStress / currStressList.Length();
//
//   double PercentLP = gwNumerics::NumLP/(double)(gwNumerics::NumLP + gwNumerics::NumNotLP);
//   //  cerr << endl <<", " << gwNumerics::get_NumLP() << ", " << gwNumerics::get_NumNotLP();
//   gwNumerics::reset_LP_Count(); //reset
//
//   NumCalls++;
//
//   if (!Quiet) {
//     if (NumCalls == 1){cycle_header();}
//     // ref see
//     // http://gcc.gnu.org/onlinedocs/iostream_3.html
//     //
//     cerr << endl ;
//     cerr << setw(4) << setiosflags(ios::left) << resetiosflags(ios::showpoint)
//       << NumCalls ;
//
//     cerr << resetiosflags(ios::left) << setiosflags(ios::showpoint)
//       << setw(11)  << setprecision(6)
//       << meanStress
//
//       << setw(11)  << setprecision(6)
//       << maxStress
//
//       << setw(9)  << setprecision(6)
//       << totalStress
//
//       << " "
//       << setw(6) << setprecision(2) << resetiosflags(ios::showpoint)
//       << 100*PercentLP
//       << setprecision(4);
//   }
//
//   if (Rtp) {
//     cout << totalStress << " " << endl;
//   }
//
//   // disable other stuff
//   //  return 0;
//
//
//   if (!Enable) return 0; //keep running
//   if (NumCalls < 5) return 0; //never quit before first 5 calls
//   // FIX  magix number == 5, make a static, a default val above and add to interface
//
// #ifdef JOB_VERBOSE2
//   cerr << "Value: \tStopEval::run() Max. Stress Atom Value(EvalCall) = " \
//        << maxStress << " ("<<NumCalls<<")"<< endl;
//   cerr << "Value: \tStopEval::run() Total Stress Value = "<< totalStress << endl;
// #endif
//
//   if (MinTotalStress > totalStress) MinTotalStress = totalStress;  // maintain mins
//   if (MinofMaxStress > maxStress) {
```

```
//     MinofMaxStress = maxStress ;
//     MinofMaxAssocTotal = totalStress;
//     BestIter = NumCalls;
//     for (int i= 0; i < CM->numAtoms(); i ++){
//       myPoints[i] = CM->getMolecule().getPosition(i); // save best positions
//     }
//   }
//
//   TotalStressList.AddToFront(totalStress);
//
//   int listlength = TotalStressList.Number();    //Must check this, may not eval every cycle.
//
//   if (listlength == LookBack) { //set up linear regression vars once
//
//     double tempX, tempY;
//     for (int i = 0; i < listlength; i++){
//       tempY = TotalStressList.Get(i);
//       tempX = (double) NumCalls - i;
//       MeanLookBackTotalStress += tempY/LookBack;
//       SumXX += tempX*tempX;
//       SumXY += tempX*tempY;
//       SumX  += tempX;
//       SumY  += tempY;
//       SumYY += tempY*tempY;
//
//       if (MinLookBackStress > tempY) MinLookBackStress = tempY;    // set init min/max over list
//       else if (MaxLookBackStress < tempY) MaxLookBackStress = tempY;
//
//     } // done lookback times
//
//     // now set init variance
//     Variance = ((LookBack*SumYY)-(SumY*SumY))/(LookBack*(LookBack-1));
//
//   } // done once to setup lookback vars
//
//   if (listlength > LookBack) { //ie listlength == lookback+1
//     // calcualte slope and mean over lookback length.  Currently I
//     // will be simply adding data to the end and removing it from the
//     // back to maintain the required data for the linear regression.
//     double OldYVal = TotalStressList.RemoveFromBack();
//     double OldX = (double) (NumCalls - LookBack);
//     double OldSumYY = SumYY;
//     double OldSumY = SumY;
//     if (VerboseOn){
//       cerr <<"StopEval:\t" ;
//       cerr << "Num: " << NumCalls <<"  \t";
//       cerr << "Added val: " << totalStress <<"  \t";
//       cerr << "Removed val: " << OldYVal << endl;
//     }
//
//     // handle min/max over list
//     if (MinLookBackStress > totalStress) MinLookBackStress = totalStress;
//     else if (MaxLookBackStress < totalStress) MaxLookBackStress = totalStress;
//     else {
//       if (MinLookBackStress == OldYVal) {
//      DLList<double>::ConstIterator it;
//      for(it = TotalStressList.GetConstIterator(); it; ++it)
//        if (MinLookBackStress > (*it)) MinLookBackStress = (*it);
//       }
//       else if (MaxLookBackStress == OldYVal){
//      DLList<double>::ConstIterator it;
//      for(it = TotalStressList.GetConstIterator(); it; ++it)
//        if (MaxLookBackStress < (*it)) MaxLookBackStress = (*it);
//       }
//     }
```

```
//
//
//     MeanLookBackTotalStress = MeanLookBackTotalStress + ((totalStress-OldYVal)/LookBack);
//
//     SumXX = SumXX + ((NumCalls)*(NumCalls) - OldX*OldX); // actually NumCalls-OldX=LookBack-1
//     SumXY = SumXY + (NumCalls*totalStress - OldX*OldYVal);
//     SumX  = SumX + (double) LookBack;
//     SumY  = SumY + (totalStress - OldYVal);
//     SumYY  = SumYY + (totalStress*totalStress-OldYVal*OldYVal);
//
//     LookBackD = LookBack*SumXX - SumX*SumX;
//     LookBackSlope = (LookBack* SumXY-SumX*SumY)/LookBackD;
//
//     Variance += (((LookBack*SumYY)-(SumY*SumY))-((LookBack*OldSumYY)-(OldSumY*OldSumY)))/(LookBack*(LookBack-1));
//     //    cerr << NumCalls << " " << totalSat << " " << Variance << endl;
//
//     if (VerboseOn){
//       cerr <<"StopEval:\t" ;
//       cerr << "Call Number: " << NumCalls <<"\t";
//       cerr << "OverallMin: " << MinTotalStress << "\t";
//       cerr << "MaxAtomStress: " << maxStress << "\t";
//       cerr << "Mean: " << MeanLookBackTotalStress << "\t";
//       cerr << "Slope: " << LookBackSlope << "\t";
//       cerr << "Variance: " << Variance << endl;
//     }
//
//     if (!Quiet){
//
//       cerr << setw(11)  << setiosflags(ios::showpoint)
//         << MeanLookBackTotalStress
//
//         << setw(11)
//         << LookBackSlope
//
//         << setw(10)
//         << Variance;
//
//     }
//
//     // CASE FOR SOLVING CONSTRAINTS ON ALL ATOMS
//     if (maxStress < 0.001 && Variance < 5)  {  //MAGIC NUMBERS!!
//       cerr << "\nSTOP by case 1 (solved)";
//       writebest();
//       return 1; //stop now
//     }
//
//     if (SolveOnly) return 0; //keep running, don't check further
//
//
//     //
//     // Simulated anealing type stuff with lpertfactor going up and down.
//     //
//
// //     if ((Variance < 5) && (LookBackSlope > -0.01) &&
// //   (NumCalls - LastLprAdj > LookBack) && (maxStress > 0.1))
// //
// //       {
// //   gwNumerics::Mul_LPertFacor(1.36);
// //
// //   NumLprAdj++;
// //   cerr << "\nLPR ADJUST ++  case 2a (stalled, no variance) cycle: " << NumCalls
// //        << "# " << NumLprAdj << endl;
// //
// //   LastLprAdj = NumCalls;
// //
```

```
// //   if (NumLprAdj == NumLprLim) {
// //     cerr << "\nLPR RESET =0.5  case 2a (stalled, not improving) cycle: " << NumCalls
// //          << "# " << NumLprAdj << endl;
// //     gwNumerics::Set_LPertFacor(1);
// //     CM->RandomPerturb(20);
// //     NumLprAdj = 0;
// //   }
// //   if (NumLprAdj > NumLprLim + 5) {
// //     cerr << "\nSTOP ++  case 2a (stalled, no variance) cycle: " << NumCalls
// //          << "# " << NumLprAdj << endl;
// //     writebest();
// //     return 1; // STOP
// //
// //       }
// //       return 0;
// //     }
//
//
// //     if ((
// //    (Variance > 5000) && (LookBackSlope > 8) &&
// //    (NumCalls - LastLprAdj >  LookBack) && (PercentLP > .2)
// //    ) || (
// //          (NumCalls - BestIter > 3 * LookBack)
// //          && (NumCalls - LastLprAdj >  LookBack )
// //         )
// //   ) {
// //       gwNumerics::Mul_LPertFacor(.81);
// //       NumLprAdj--;
// //
// //       cerr << "\nLPR ADJUST --  case 2b (stalled, not improving) cycle: " << NumCalls
// //      << "# " << NumLprAdj << endl;
// //
// //       LastLprAdj = NumCalls;
// //
// //       if (NumLprAdj == (-1)*NumLprLim) {
// //   cerr << "\nLPR RESET =3  case 2b (stalled, not improving) cycle: " << NumCalls
// //        << "# " << NumLprAdj << endl;
// //   gwNumerics::Set_LPertFacor(3);
// //   CM->RandomPerturb(20);
// //   NumLprAdj = 0;
// //       }
// //
// //       if (NumLprAdj < (-1)*NumLprLim - 7) {
// //   cerr << "\nSTOP --  case 2b (stalled, not improving) cycle: " << NumCalls
// //        << "# " << NumLprAdj << endl;
// //   writebest();
// //   return 1; // STOP
// //       }
// //       return 0;
// //     }
//
//
//
//
//     // getting worse
//     if ((LookBackSlope > 100) && (NumCalls - LastLprAdj >  LookBack ) &&
//      (MinTotalStress < MeanLookBackTotalStress/ 5) &&
//      (NumLprAdj < 1)) {
//       cerr << "\nSTOP by case 3 (getting worse) Slope = " << LookBackSlope << endl;
//       writebest();
//       return 1;
//     }
//
//
//   }
```

```
00373
```

4.67.3.5 void StopEval::writebest (void)

Definition at line 379 of file jd_stop_eval.C.

Referenced by AtomBuildup::run().

```
00379                              {
00380 
00381   int N = CM->numAtoms();
00382 
00383   DynString comment;
00384   DynString FN;
00385   comment = BaseFN +
00386     DynString(" ;Best Iteration ") + DynString(BestIter) +
00387     DynString(" ;Ave Err = ") + DynString (BestAveErr) +
00388     DynString(" ;Max Err = ") + DynString (BestMaxErr);
00389   FN = BaseFN + DynString(".best.xmol");
00390 
00391   char * s = &(FN[0]);  // ACK  not good, but it doesn't leak. -jd
00392   ofstream os(s);
00393 
00394   os << N << "\n";
00395   os << comment << "\n";
00396   for(int i=0; i<N; ++i)
00397     os << CM->getAtom(i).Type() <<" "<< myPoints[i] << endl;
00398 
00399 
00400 
```

4.67.4 Member Data Documentation

4.67.4.1 AnimFrame * StopEval::AF [private]

Definition at line 79 of file jd_stop_eval.h.

4.67.4.2 double StopEval::BestAveErr [private]

Definition at line 139 of file jd_stop_eval.h.

4.67.4.3 int StopEval::BestIter = 0 [static, private]

Definition at line 31 of file jd_stop_eval.C.

4.67.4.4 double StopEval::BestMaxErr [private]

Definition at line 140 of file jd_stop_eval.h.

4.67.4.5 bool StopEval::Enable = true [static]

Definition at line 69 of file globals.C.

4.67.4.6 int StopEval::LastLprAdj = 0 [static, private]

Definition at line 30 of file jd_stop_eval.C.

4.67.4.7 int StopEval::LookBack = 50 [static, private]

number of calls to StopEval to remember.

Definition at line 68 of file globals.C.

4.67.4.8 double StopEval::LookBackD [static, private]

Definition at line 136 of file jd_stop_eval.h.

4.67.4.9 double StopEval::LookBackSlope [static, private]

Definition at line 135 of file jd_stop_eval.h.

4.67.4.10 double StopEval::LowErr [private]

Definition at line 138 of file jd_stop_eval.h.

4.67.4.11 double StopEval::MaxLookBackStress [static, private]

Definition at line 111 of file jd_stop_eval.h.

4.67.4.12 double StopEval::MeanLookBackTotalStress [static, private]

mean of the lookback list vals on total stress, incrementally updated after lookback number of calls to stopeval

Definition at line 117 of file jd_stop_eval.h.

4.67.4.13 double StopEval::MinLookBackStress [static, private]

The Minimum (max) total Stress value observed over lookback calls to stop_eval.

Definition at line 110 of file jd_stop_eval.h.

4.67.4.14 double StopEval::MinTotalStress [static, private]

Definition at line 93 of file jd_stop_eval.h.

4.67.4.15 double StopEval::MinofMaxAssocTotal [static, private]

the asociated total stress on the saved "best" molecule defined by MinofMaxStress'

Definition at line 104 of file jd_stop_eval.h.

4.67.4.16 double StopEval::MinofMaxStress [static, private]

the overall min of the max on each list of stress calls on an iteration.

Definition at line 98 of file jd_stop_eval.h.

4.67.4.17 int StopEval::NumCalls = 0 [static, private]

Definition at line 27 of file jd_stop_eval.C.

4.67.4.18 int StopEval::NumLprAdj = 0 [static, private]

Definition at line 28 of file jd_stop_eval.C.

4.67.4.19 int StopEval::NumLprLim = 10 [static, private]

Definition at line 29 of file jd_stop_eval.C.

4.67.4.20 bool StopEval::Quiet = false [static]

run-time ajdustable flag to stop default output.

Definition at line 72 of file globals.C.

4.67.4.21 bool StopEval::Rtp = false [static]

Real time plotting output – single double outputs for plotting.

Definition at line 73 of file globals.C.

4.67.4.22 bool StopEval::SolveOnly = false [static]

Definition at line 70 of file globals.C.

4.67.4.23 double StopEval::SumX [static, private]

Definition at line 123 of file jd_stop_eval.h.

4.67.4.24 double StopEval::SumXX [static, private]

Variables for linear regression these are maintained over the lookback calls to **StopEval::run**() (p. 493).

Definition at line 121 of file jd_stop_eval.h.

4.67.4.25 double StopEval::SumXY [static, private]

Definition at line 122 of file jd_stop_eval.h.

4.67.4.26 double StopEval::SumY [static, private]

Definition at line 124 of file jd_stop_eval.h.

4.67.4.27 double StopEval::SumYY [static, private]

Definition at line 125 of file jd_stop_eval.h.

4.67.4.28 DLList< double > StopEval::TotalStressList [static, private]

list of total stress values kept over the last lookback number of calls to stop eval. Min, Mean and slope are used over this list.

Definition at line 33 of file jd_stop_eval.C.

4.67.4.29 double StopEval::Variance [static, private]

Definition at line 126 of file jd_stop_eval.h.

4.67.4.30 bool StopEval::VerboseOn = false [static]

run-time ajdustable flag to provide verbose output.

Definition at line 71 of file globals.C.

4.67.4.31 jVector * StopEval::myPoints [private]

Definition at line 81 of file jd_stop_eval.h.

4.67.4.32 ofstream * StopEval::of [private]

Definition at line 80 of file jd_stop_eval.h.

The documentation for this class was generated from the following files:

- **jd_stop_eval.h**
- **globals.C**
- **jd_stop_eval.C**

4.68 UnitWeight Class Reference

#include <graph_weights.h>

Public Methods

- **UnitWeight** ()
- **UnitWeight** (double)
- **UnitWeight** (int)
- **UnitWeight** (bool)
- const UnitWeight& **operator=** (const UnitWeight &)
- **operator bool** () const
- **operator int** () const
- **operator double** () const
- int **Output** (ostream &os) const
- int **Input** (istream &)
- int **OutputBinary** (ostream &) const
- int **InputBinary** (istream &)
- **MAKECOMPARISONS_NO_COMPARE** (UnitWeight)
- **MAKE_IO_STREAM_OPERATORS** (UnitWeight)
- **MAKE_BIN_STREAM_OPERATORS** (UnitWeight)

4.68.1 Constructor & Destructor Documentation

4.68.1.1 UnitWeight::UnitWeight () [inline]

Definition at line 11 of file graph_weights.h.

```
00011                                   {}
00012
```

4.68.1.2 UnitWeight::UnitWeight (double) [inline]

Definition at line 12 of file graph_weights.h.

```
00012                                         {}
00013
```

4.68.1.3 UnitWeight::UnitWeight (int) [inline]

Definition at line 13 of file graph_weights.h.

```
00013                                      {}
00014
```

4.68.1.4 UnitWeight::UnitWeight (bool) [inline]

Definition at line 14 of file graph_weights.h.

```
00014                                       {}
00015
```

4.68.2 Member Function Documentation

4.68.2.1 int UnitWeight::Input (istream &) [inline]

Definition at line 26 of file graph_weights.h.

```
00027                   { return 1; }
00028
```

4.68.2.2 int UnitWeight::InputBinary (istream &) [inline]

Definition at line 30 of file graph_weights.h.

```
00031                   { return 1; }
00032
```

4.68.2.3 UnitWeight::MAKECOMPARISONS_NO_COMPARE (UnitWeight)

4.68.2.4 UnitWeight::MAKE_BIN_STREAM_OPERATORS (UnitWeight)

4.68.2.5 UnitWeight::MAKE_IO_STREAM_OPERATORS (UnitWeight)

4.68.2.6 int UnitWeight::Output (ostream & *os*) const [inline]

Definition at line 24 of file graph_weights.h.

```
00025                   { os << "_1_"; return 1; }
00026
```

4.68.2.7 int UnitWeight::OutputBinary (ostream &) const [inline]

Definition at line 28 of file graph_weights.h.

```
00029                   { return 1; }
00030
```

4.68.2.8 UnitWeight::operator bool () const [inline]

Definition at line 20 of file graph_weights.h.

```
00020                                       { return true; }
00021
```

4.68.2.9 UnitWeight::operator double () const [inline]

Definition at line 22 of file graph_weights.h.

```
00022                                         { return 1.0; }
00023
```

4.68.2.10 UnitWeight::operator int () const [inline]

Definition at line 21 of file graph_weights.h.

```
00021                                        { return 1; }
00022
```

4.68.2.11 const UnitWeight & UnitWeight::operator= (const UnitWeight &) [inline]

Definition at line 16 of file graph_weights.h.

```
00017                   { return *this; }
00018
```

The documentation for this class was generated from the following file:

- **graph_weights.h**

4.69 z Class Reference

`#include <randomgenerator.h>`

Public Methods

- **z** ()
- ~**z** ()

Public Attributes

- int **seeded**
- double **u** [98]
- double **c**
- double **cd**
- double **cm**
- int **i97**
- int **j97**

4.69.1 Constructor & Destructor Documentation

4.69.1.1 z::z () `[inline]`

Definition at line 65 of file randomgenerator.h.

```
00065                   { seeded = 0; }
00066
```

4.69.1.2 z::~z () `[inline]`

Definition at line 66 of file randomgenerator.h.

```
00066                    {}
00067
```

4.69.2 Member Data Documentation

4.69.2.1 double z::c

Definition at line 69 of file randomgenerator.h.

4.69.2.2 double z::cd

Definition at line 69 of file randomgenerator.h.

4.69.2.3 double z::cm

Definition at line 69 of file randomgenerator.h.

4.69.2.4 int z::i97

Definition at line 70 of file randomgenerator.h.

4.69.2.5 int z::j97

Definition at line 70 of file randomgenerator.h.

4.69.2.6 int z::seeded

Definition at line 67 of file randomgenerator.h.

4.69.2.7 double z::u[98]

Definition at line 68 of file randomgenerator.h.

The documentation for this class was generated from the following file:

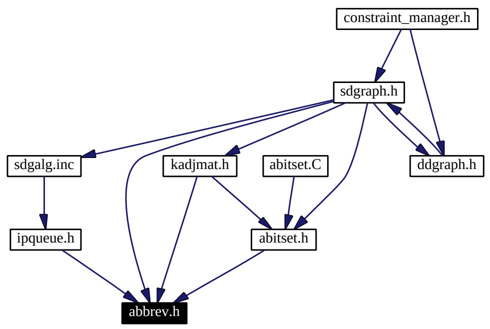
- **randomgenerator.h**

Chapter 5

GRAPE: Generalizable Replaceable Atomic Position Engine File Documentation

5.1 abbrev.h File Reference

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef unsigned int **uint**
- typedef short int **sint**
- typedef long int **lint**
- typedef unsigned short int **usint**
- typedef unsigned long int **ulint**
- typedef unsigned char **uchar**
- typedef class **ABitSet** **ABS**

5.1.1 Typedef Documentation

5.1.1.1 typedef class ABitSet ABS

Definition at line 17 of file abbrev.h.

5.1.1.2 typedef long int lint

Definition at line 13 of file abbrev.h.

5.1.1.3 typedef short int sint

Definition at line 12 of file abbrev.h.

5.1.1.4 typedef unsigned char uchar

Definition at line 16 of file abbrev.h.

5.1.1.5 typedef unsigned int uint

Definition at line 11 of file abbrev.h.

5.1.1.6 typedef unsigned long int ulint

Definition at line 15 of file abbrev.h.

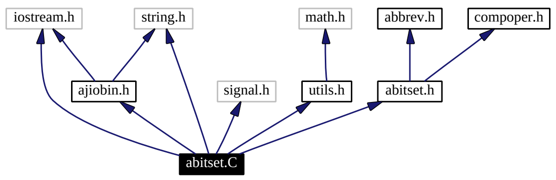
5.1.1.7 typedef unsigned short int usint

Definition at line 14 of file abbrev.h.

5.2 abitset.C File Reference

```
#include <iostream.h>
#include <string.h>
#include <signal.h>
#include "utils.h"
#include "ajiobin.h"
#include "abitset.h"
```

Include dependency graph for abitset.C:



Functions

- **uint b2c0** (**uint** i)
- **uint b2c1** (**uint** i)
- char **b2i** (**uint** i)
- ostream& **operator<<** (ostream &os, const **ABitSet** &abs)
- ostream& **OutputBinary** (ostream &os, const **ABitSet** &bs)
- istream& **InputBinary** (istream &is, **ABitSet** &bs)

Variables

- const char **all_set** [8] = {255,1,3,7,15,31,63,127}

5.2.1 Function Documentation

5.2.1.1 istream & InputBinary (istream & *is*, ABitSet & *bs*)

Definition at line 448 of file abitset.C.

Referenced by InputBinary(), and operator>=().

```
{
        if (!is)
                return is;

        uint    n;
        InputBinary(is,n);

        delete[] bs.data;
        bs.num = n;
        bs.data = new char[b2c1(n)];
        if (!bs.data)
                raise(SIGQUIT);

```

```
00462         is.read(bs.data, b2c1(bs.num));
00463         return is;
00464
```

5.2.1.2 ostream & OutputBinary (ostream & *os*, const ABitSet & *bs*)

Definition at line 436 of file abitset.C.

Referenced by OutputBinary(), and operator<=().

```
00437 {
00438         if (!os)
00439                 return os;
00440         OutputBinary(os,bs.num);
00441         if (!os)
00442                 return os;
00443         os.write(bs.data, b2c1(bs.num));
00444         return os;
00445
```

5.2.1.3 uint b2c0 (uint *i*) [inline]

Definition at line 13 of file abitset.C.

Referenced by ABitSet::Clear(), ABitSet::Invert(), ABitSet::Set(), and ABitSet::operator[]().

```
00014 {
00015         return i/8;
00016
```

5.2.1.4 uint b2c1 (uint *i*) [inline]

Definition at line 18 of file abitset.C.

Referenced by ABitSet::ABitSet(), ABitSet::ClearAll(), ABitSet::Compare(), ABitSet::Count(), ABitSet::Decrement(), ABitSet::Increment(), InputBinary(), ABitSet::InvertAll(), ABitSet::IsAllClear(), ABitSet::IsAllSet(), OutputBinary(), ABitSet::SetAll(), ABitSet::operator &=(), ABitSet::operator-=(), ABitSet::operator=(), ABitSet::operator^=(), and ABitSet::operator|=().

```
00019 {
00020         return (i+7)/8;
00021
```

5.2.1.5 char b2i (uint *i*) [inline]

Definition at line 23 of file abitset.C.

Referenced by ABitSet::Invert(), ABitSet::Set(), and ABitSet::operator[]().

```
00024 {
00025         return 1 << (i%8);
00026
```

5.2.1.6 ostream & operator<< (ostream & *os*, const ABitSet & *abs*)

Definition at line 367 of file abitset.C.

```
{
        os << '<';
        for (uint i=0; i<abs.num; ++i)
                os << (abs[i] ? '1' : '0');
        os << '>';
        return os;
```

5.2.2 Variable Documentation

5.2.2.1 const char all_set[8] = {255,1,3,7,15,31,63,127} `[static]`

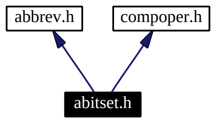
Definition at line 10 of file abitset.C.

5.3 abitset.h File Reference

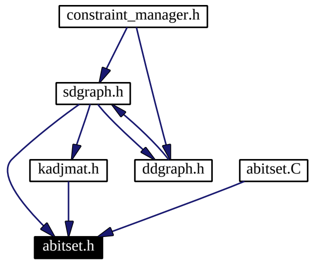
`#include "abbrev.h"`

`#include "compoper.h"`

Include dependency graph for abitset.h:



This graph shows which files directly or indirectly include this file:



Compounds

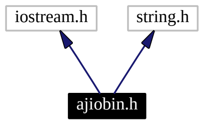
- class **ABitSet**
- class **ABitSet::Iterator**

5.4 ajiobin.h File Reference

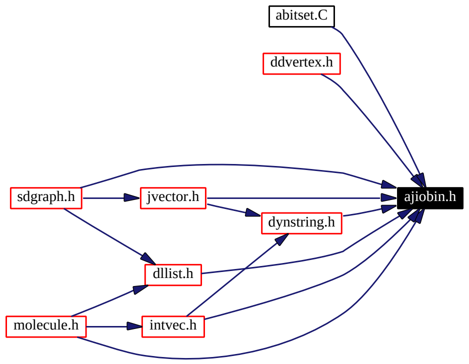
`#include <iostream.h>`

`#include <string.h>`

Include dependency graph for ajiobin.h:



This graph shows which files directly or indirectly include this file:



Functions

- ostream& **operator<=** (ostream &os, char c)
- ostream& **operator<=** (ostream &os, unsigned char c)
- ostream& **operator<=** (ostream &os, signed char c)
- ostream& **operator<=** (ostream &os, int i)
- ostream& **operator<=** (ostream &os, unsigned int i)
- ostream& **operator<=** (ostream &os, short s)
- ostream& **operator<=** (ostream &os, unsigned short s)
- ostream& **operator<=** (ostream &os, long l)
- ostream& **operator<=** (ostream &os, unsigned long l)
- ostream& **operator<=** (ostream &os, const char ∗cp)
- ostream& **operator<=** (ostream &os, bool b)
- ostream& **operator<=** (ostream &os, float f)
- ostream& **operator<=** (ostream &os, double d)
- ostream& **OutputBinary** (ostream &os, char c)
- ostream& **OutputBinary** (ostream &os, unsigned char c)
- ostream& **OutputBinary** (ostream &os, signed char c)
- ostream& **OutputBinary** (ostream &os, int i)
- ostream& **OutputBinary** (ostream &os, unsigned int i)
- ostream& **OutputBinary** (ostream &os, short s)
- ostream& **OutputBinary** (ostream &os, unsigned short s)

- ostream& **OutputBinary** (ostream &os, long l)
- ostream& **OutputBinary** (ostream &os, unsigned long l)
- ostream& **OutputBinary** (ostream &os, const char ∗cp)
- ostream& **OutputBinary** (ostream &os, bool b)
- ostream& **OutputBinary** (ostream &os, float f)
- ostream& **OutputBinary** (ostream &os, double d)
- istream& **operator>=** (istream &is, char &c)
- istream& **operator>=** (istream &is, unsigned char &c)
- istream& **operator>=** (istream &is, signed char &c)
- istream& **operator>=** (istream &is, int &i)
- istream& **operator>=** (istream &is, unsigned int &i)
- istream& **operator>=** (istream &is, short &s)
- istream& **operator>=** (istream &is, unsigned short &s)
- istream& **operator>=** (istream &is, long &l)
- istream& **operator>=** (istream &is, unsigned long &l)
- istream& **operator>=** (istream &is, char ∗cp)
- istream& **operator>=** (istream &is, bool &b)
- istream& **operator>=** (istream &is, float &f)
- istream& **operator>=** (istream &is, double &d)
- istream& **operator>=** (istream &is, long double &d)
- istream& **InputBinary** (istream &is, char &c)
- istream& **InputBinary** (istream &is, unsigned char &c)
- istream& **InputBinary** (istream &is, signed char &c)
- istream& **InputBinary** (istream &is, int &i)
- istream& **InputBinary** (istream &is, unsigned int &i)
- istream& **InputBinary** (istream &is, short &s)
- istream& **InputBinary** (istream &is, unsigned short &s)
- istream& **InputBinary** (istream &is, long &l)
- istream& **InputBinary** (istream &is, unsigned long &l)
- istream& **InputBinary** (istream &is, char ∗cp)
- istream& **InputBinary** (istream &is, bool &b)
- istream& **InputBinary** (istream &is, float &f)
- istream& **InputBinary** (istream &is, double &d)
- istream& **InputBinary** (istream &is, long double &d)

5.4.1 Function Documentation

5.4.1.1 istream & InputBinary (istream & *is*, long double & *d*) `[inline]`

Definition at line 352 of file ajiobin.h.

```
{
        return is.read(&d,sizeof(long double));

```

5.4.1.2 istream & InputBinary (istream & *is*, double & *d*) `[inline]`

Definition at line 347 of file ajiobin.h.

```
{
        return is.read(&d,sizeof(double));

```

5.4.1.3 istream & InputBinary (istream & *is*, float & *f*) [inline]

Definition at line 342 of file ajiobin.h.

```
{
        return is.read(&f,sizeof(float));

```

5.4.1.4 istream & InputBinary (istream & *is*, bool & *b*) [inline]

Definition at line 337 of file ajiobin.h.

```
{
        return is.read(&b,sizeof(bool));

```

5.4.1.5 istream & InputBinary (istream & *is*, char * *cp*) [inline]

Definition at line 329 of file ajiobin.h.

```
{
        int     len;
        is.read(&len,sizeof(int));
        is.read(cp,len);
        return is;

```

5.4.1.6 istream & InputBinary (istream & *is*, unsigned long & *l*) [inline]

Definition at line 324 of file ajiobin.h.

```
{
        return is.read(&l,sizeof(unsigned long));

```

5.4.1.7 istream & InputBinary (istream & *is*, long & *l*) [inline]

Definition at line 319 of file ajiobin.h.

```
{
        return is.read(&l,sizeof(long));

```

5.4.1.8 istream & InputBinary (istream & *is*, unsigned short & *s*) [inline]

Definition at line 314 of file ajiobin.h.

```
{
        return is.read(&s,sizeof(unsigned short));

```

5.4.1.9 istream & InputBinary (istream & *is*, short & *s*) [inline]

Definition at line 309 of file ajiobin.h.

```
{
        return is.read(&s,sizeof(short));

```

5.4.1.10 istream & InputBinary (istream & *is*, unsigned int & *i*) [inline]

Definition at line 304 of file ajiobin.h.

```
{
        return is.read(&i,sizeof(unsigned int));

```

5.4.1.11 istream & InputBinary (istream & *is*, int & *i*) [inline]

Definition at line 299 of file ajiobin.h.

```
{
        return is.read(&i,sizeof(int));

```

5.4.1.12 istream & InputBinary (istream & *is*, signed char & *c*) [inline]

Definition at line 294 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(signed char));

```

5.4.1.13 istream & InputBinary (istream & *is*, unsigned char & *c*) [inline]

Definition at line 289 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(unsigned char));

```

5.4.1.14 istream & InputBinary (istream & *is*, char & *c*) [inline]

Definition at line 284 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(char));

```

5.4.1.15 ostream & OutputBinary (ostream & *os*, double *d*) [inline]

Definition at line 188 of file ajiobin.h.

```
{
        os.write(&d,sizeof(double));
        return os;
```

5.4.1.16 ostream & OutputBinary (ostream & *os*, float *f*) [inline]

Definition at line 182 of file ajiobin.h.

```
{
        os.write(&f,sizeof(float));
        return os;
```

5.4.1.17 ostream & OutputBinary (ostream & *os*, bool *b*) [inline]

Definition at line 176 of file ajiobin.h.

```
{
        os.write(&b,sizeof(bool));
        return os;
```

5.4.1.18 ostream & OutputBinary (ostream & *os*, const char * *cp*) [inline]

Definition at line 168 of file ajiobin.h.

```
{
        int     len = strlen(cp)+1;
        os.write(&len,sizeof(int));
        os.write(cp,len);
        return os;
```

5.4.1.19 ostream & OutputBinary (ostream & *os*, unsigned long *l*) [inline]

Definition at line 162 of file ajiobin.h.

```
{
        os.write(&l,sizeof(unsigned long));
        return os;
```

5.4.1.20 ostream & OutputBinary (ostream & *os*, long *l*) [inline]

Definition at line 156 of file ajiobin.h.

```
{
        os.write(&l,sizeof(long));
        return os;
```

5.4.1.21 ostream & OutputBinary (ostream & *os*, unsigned short *s*) `[inline]`

Definition at line 150 of file ajiobin.h.

```
{
        os.write(&s,sizeof(unsigned short));
        return os;
```

5.4.1.22 ostream & OutputBinary (ostream & *os*, short *s*) `[inline]`

Definition at line 144 of file ajiobin.h.

```
{
        os.write(&s,sizeof(short));
        return os;
```

5.4.1.23 ostream & OutputBinary (ostream & *os*, unsigned int *i*) `[inline]`

Definition at line 138 of file ajiobin.h.

```
{
        os.write(&i,sizeof(unsigned int));
        return os;
```

5.4.1.24 ostream & OutputBinary (ostream & *os*, int *i*) `[inline]`

Definition at line 132 of file ajiobin.h.

```
{
        os.write(&i,sizeof(int));
        return os;
```

5.4.1.25 ostream & OutputBinary (ostream & *os*, signed char *c*) `[inline]`

Definition at line 126 of file ajiobin.h.

```
{
        os.write(&c,sizeof(signed char));
        return os;
```

5.4.1.26 ostream & OutputBinary (ostream & *os*, unsigned char *c*) `[inline]`

Definition at line 120 of file ajiobin.h.

```
{
        os.write(&c,sizeof(unsigned char));
        return os;
```

5.4.1.27 ostream & OutputBinary (ostream & *os*, char *c*) [inline]

Definition at line 114 of file ajiobin.h.

```
{
        os.write(&c,sizeof(char));
        return os;
```

5.4.1.28 ostream & operator<= (ostream & *os*, double *d*) [inline]

Definition at line 98 of file ajiobin.h.

```
{
        os.write(&d,sizeof(double));
        return os;
```

5.4.1.29 ostream & operator<= (ostream & *os*, float *f*) [inline]

Definition at line 92 of file ajiobin.h.

```
{
        os.write(&f,sizeof(float));
        return os;
```

5.4.1.30 ostream & operator<= (ostream & *os*, bool *b*) [inline]

Definition at line 86 of file ajiobin.h.

```
{
        os.write(&b,sizeof(bool));
        return os;
```

5.4.1.31 ostream & operator<= (ostream & *os*, const char * *cp*) [inline]

Definition at line 78 of file ajiobin.h.

```
{
        int     len = strlen(cp)+1;
        os.write(&len,sizeof(int));
        os.write(cp,len);
        return os;
```

5.4.1.32 ostream & operator<= (ostream & *os*, unsigned long *l*) [inline]

Definition at line 72 of file ajiobin.h.

```
{
        os.write(&l,sizeof(unsigned long));
        return os;
```

5.4.1.33 ostream & operator<= (ostream & *os*, long *l*) [inline]

Definition at line 66 of file ajiobin.h.

```
00067 {
00068         os.write(&l,sizeof(long));
00069         return os;
00070
```

5.4.1.34 ostream & operator<= (ostream & *os*, unsigned short *s*) [inline]

Definition at line 60 of file ajiobin.h.

```
00061 {
00062         os.write(&s,sizeof(unsigned short));
00063         return os;
00064
```

5.4.1.35 ostream & operator<= (ostream & *os*, short *s*) [inline]

Definition at line 54 of file ajiobin.h.

```
00055 {
00056         os.write(&s,sizeof(short));
00057         return os;
00058
```

5.4.1.36 ostream & operator<= (ostream & *os*, unsigned int *i*) [inline]

Definition at line 48 of file ajiobin.h.

```
00049 {
00050         os.write(&i,sizeof(unsigned int));
00051         return os;
00052
```

5.4.1.37 ostream & operator<= (ostream & *os*, int *i*) [inline]

Definition at line 42 of file ajiobin.h.

```
00043 {
00044         os.write(&i,sizeof(int));
00045         return os;
00046
```

5.4.1.38 ostream & operator<= (ostream & *os*, signed char *c*) [inline]

Definition at line 36 of file ajiobin.h.

```
00037 {
00038         os.write(&c,sizeof(signed char));
00039         return os;
00040
```

5.4.1.39 ostream & operator<= (ostream & *os*, unsigned char *c*) [inline]

Definition at line 30 of file ajiobin.h.

```
{
        os.write(&c,sizeof(unsigned char));
        return os;
```

5.4.1.40 ostream & operator<= (ostream & *os*, char *c*) [inline]

Definition at line 24 of file ajiobin.h.

```
{
        os.write(&c,sizeof(char));
        return os;
```

5.4.1.41 istream & operator>= (istream & *is*, long double & *d*) [inline]

Definition at line 275 of file ajiobin.h.

```
{
        return is.read(&d,sizeof(long double));
```

5.4.1.42 istream & operator>= (istream & *is*, double & *d*) [inline]

Definition at line 270 of file ajiobin.h.

```
{
        return is.read(&d,sizeof(double));
```

5.4.1.43 istream & operator>= (istream & *is*, float & *f*) [inline]

Definition at line 265 of file ajiobin.h.

```
{
        return is.read(&f,sizeof(float));
```

5.4.1.44 istream & operator>= (istream & *is*, bool & *b*) [inline]

Definition at line 260 of file ajiobin.h.

```
{
        return is.read(&b,sizeof(bool));
```

5.4.1.45 istream & operator>= (istream & *is*, char ∗ *cp*) [inline]

Definition at line 252 of file ajiobin.h.

```
{
        int     len;
        is.read(&len,sizeof(int));
        is.read(cp,len);
        return is;
```

5.4.1.46 istream & operator>= (istream & *is*, unsigned long & *l*) [inline]

Definition at line 247 of file ajiobin.h.

```
{
        return is.read(&l,sizeof(unsigned long));
```

5.4.1.47 istream & operator>= (istream & *is*, long & *l*) [inline]

Definition at line 242 of file ajiobin.h.

```
{
        return is.read(&l,sizeof(long));
```

5.4.1.48 istream & operator>= (istream & *is*, unsigned short & *s*) [inline]

Definition at line 237 of file ajiobin.h.

```
{
        return is.read(&s,sizeof(unsigned short));
```

5.4.1.49 istream & operator>= (istream & *is*, short & *s*) [inline]

Definition at line 232 of file ajiobin.h.

```
{
        return is.read(&s,sizeof(short));
```

5.4.1.50 istream & operator>= (istream & *is*, unsigned int & *i*) [inline]

Definition at line 227 of file ajiobin.h.

```
{
        return is.read(&i,sizeof(unsigned int));
```

5.4.1.51 istream & operator>= (istream & *is*, int & *i*) [inline]

Definition at line 222 of file ajiobin.h.

```
{
        return is.read(&i,sizeof(int));

```

5.4.1.52 istream & operator>= (istream & *is*, signed char & *c*) [inline]

Definition at line 217 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(signed char));

```

5.4.1.53 istream & operator>= (istream & *is*, unsigned char & *c*) [inline]

Definition at line 212 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(unsigned char));

```

5.4.1.54 istream & operator>= (istream & *is*, char & *c*) [inline]

Definition at line 207 of file ajiobin.h.

```
{
        return is.read(&c,sizeof(char));

```

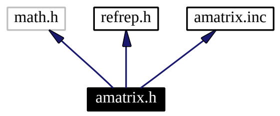
5.5 amatrix.h File Reference

`#include <math.h>`

`#include <refrep.h>`

`#include "amatrix.inc"`

Include dependency graph for amatrix.h:

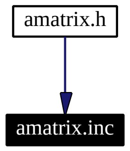


Compounds

- class **AMatrix**
- class **AMatrix::AMatrixRep**

5.6 amatrix.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- ostream& **operator<=** (ostream &os, const **AMatrix** &m)
- istream& **operator>=** (istream &is, **AMatrix** &m)

5.6.1 Function Documentation

5.6.1.1 ostream & operator<= (ostream & *os*, const AMatrix & *m*) `[inline]`

Definition at line 404 of file amatrix.inc.

```
00405 { return OutputBinary(os,m); }
00406
```

5.6.1.2 istream & operator>= (istream & *is*, AMatrix & *m*) `[inline]`

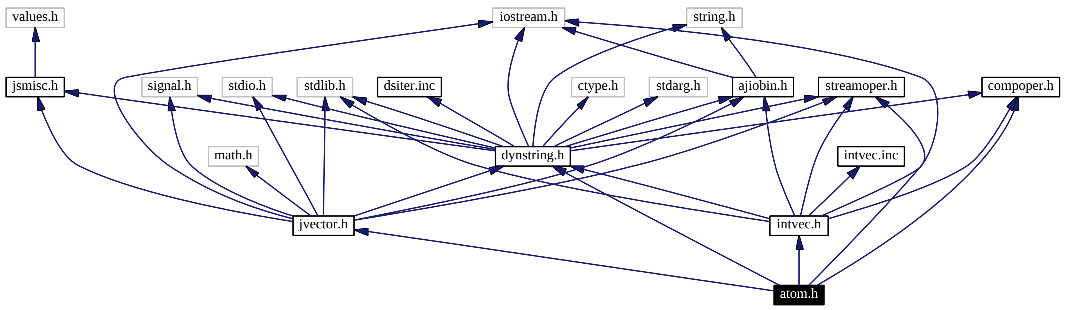
Definition at line 408 of file amatrix.inc.

```
00409 { return InputBinary(is,m); }
00410
```

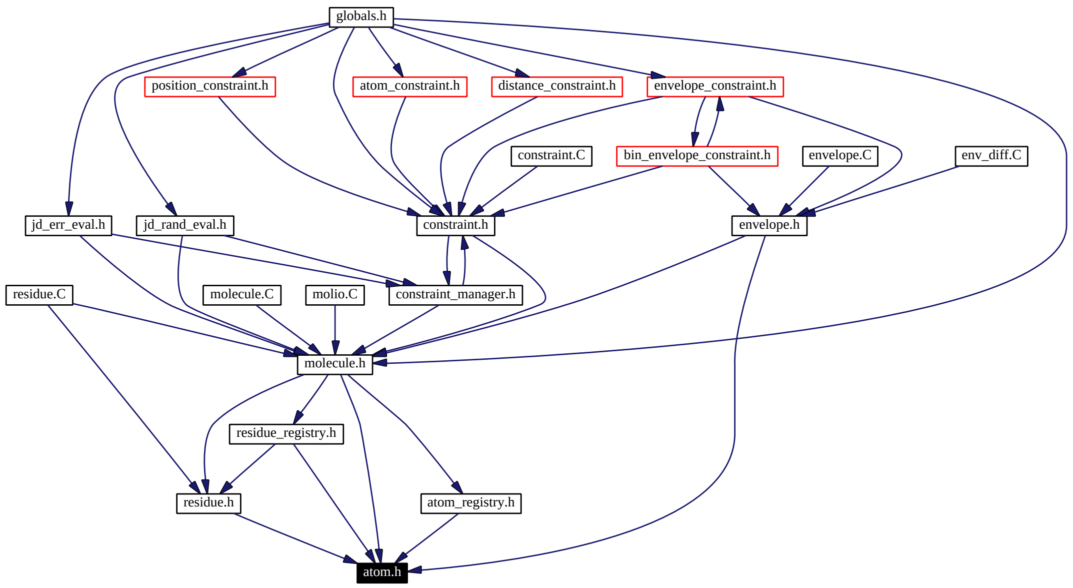
5.7 atom.h File Reference

```
#include "dynstring.h"
#include "intvec.h"
#include "jvector.h"
#include "compoper.h"
#include "streamoper.h"
```

Include dependency graph for atom.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Atom**

Defines

- #define **UNINITIALIZED_ATOM_ID** -1

- #define **MAX_NUM_BONDS** 14;

5.7.1 Define Documentation

5.7.1.1 #define MAX_NUM_BONDS 14;

Definition at line 29 of file atom.h.

5.7.1.2 #define UNINITIALIZED_ATOM_ID -1

//

File: **atom.h**

Used to save information about one atom.

Positions are no longer stored in **Atom** (p. 43)' instead the molecule is resposible for them.

So now **Atom** (p. 43) class doesn't hold any useful information except ID Eventually It will hold VDW radius, atom type, local perturbation settings, chain identifier, residue/nucleotide number – all things intrinsic to the atom. (-jd)

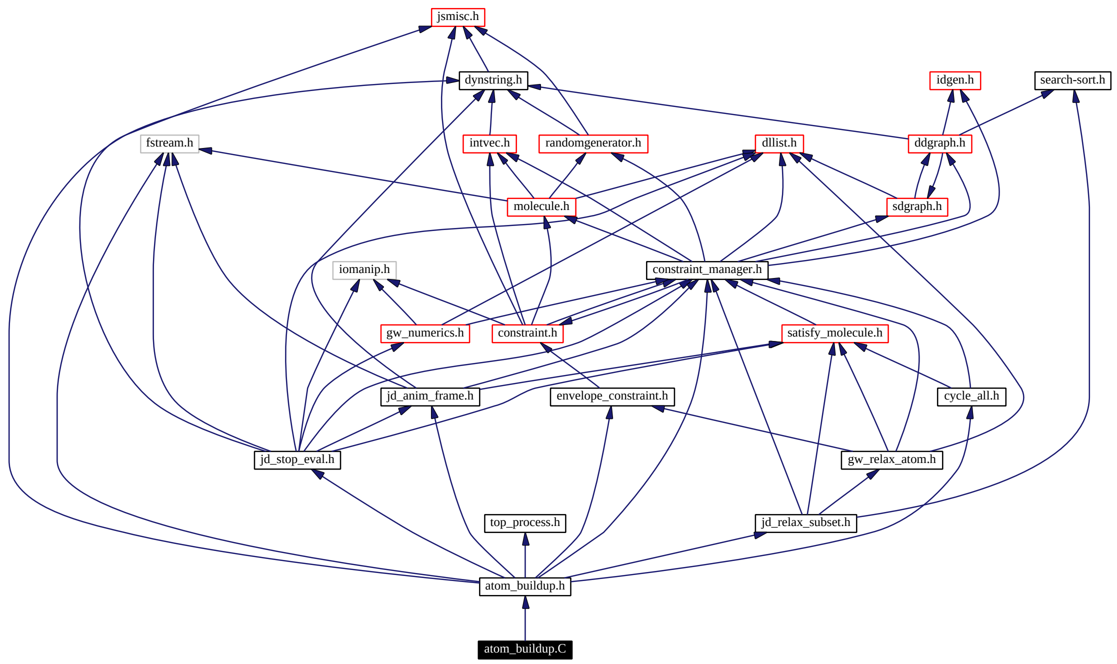
//

Definition at line 28 of file atom.h.

5.8 atom_buildup.C File Reference

`#include "atom_buildup.h"`

Include dependency graph for atom_buildup.C:



5.9 atom_buildup.h File Reference

`#include <fstream.h>`

`#include "constraint_manager.h"`

`#include "top_process.h"`

`#include "dynstring.h"`

`#include "cycle_all.h"`

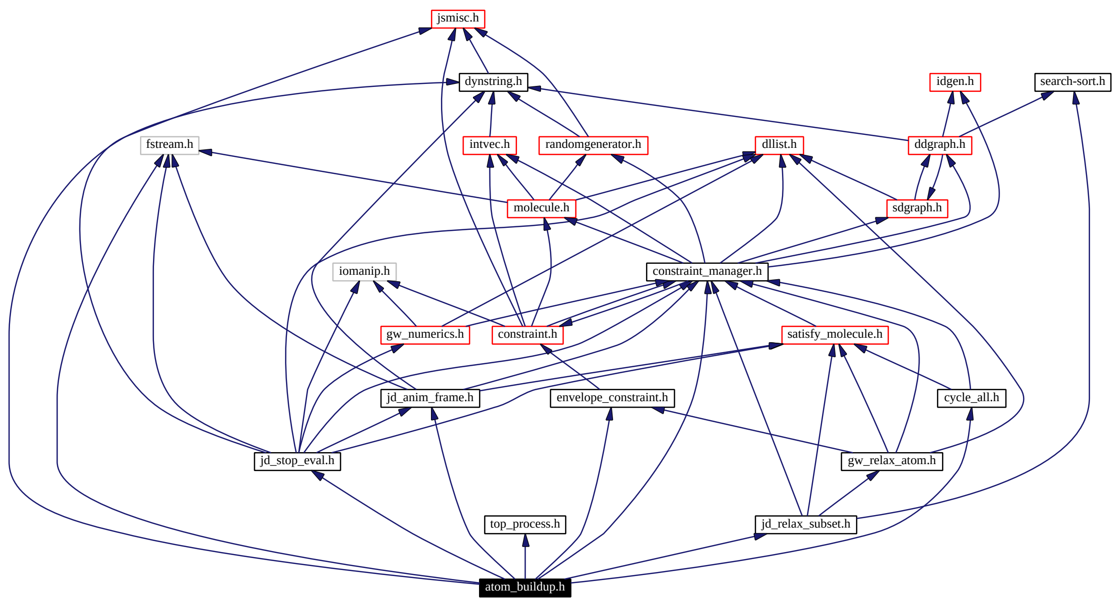
`#include "jd_stop_eval.h"`

`#include "jd_anim_frame.h"`

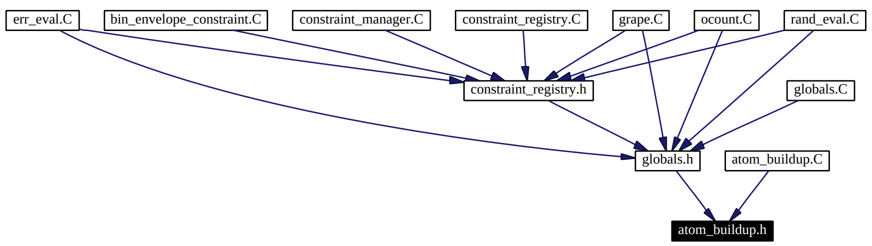
`#include "jd_relax_subset.h"`

`#include "envelope_constraint.h"`

Include dependency graph for atom_buildup.h:



This graph shows which files directly or indirectly include this file:



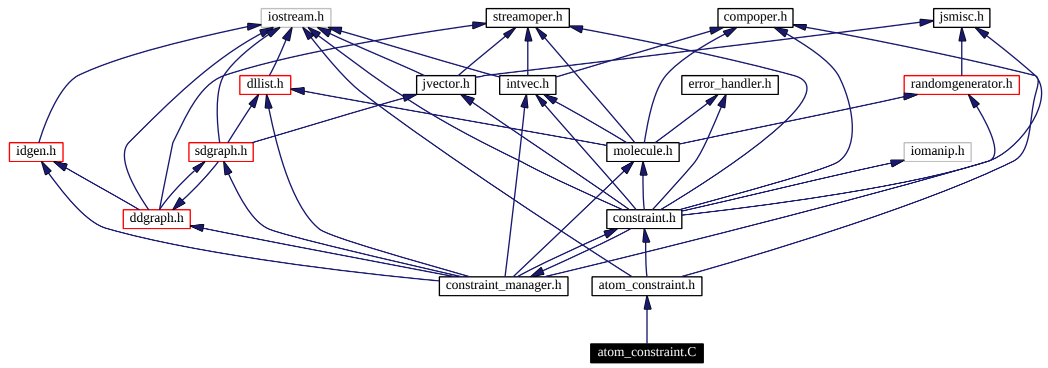
Compounds

- class **AtomBuildup**

5.10 atom_constraint.C File Reference

`#include "atom_constraint.h"`

Include dependency graph for atom_constraint.C:



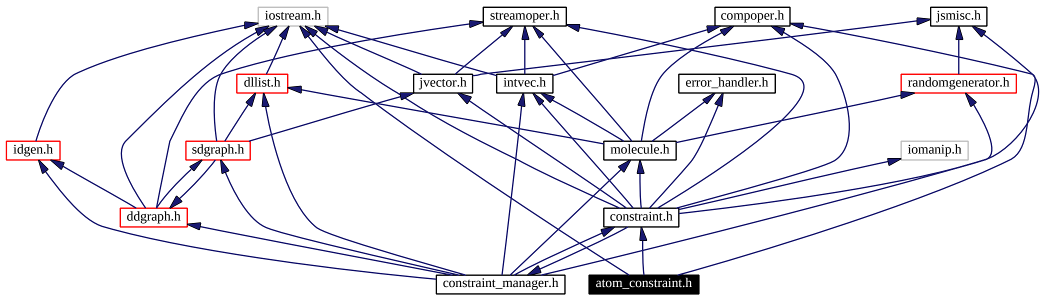
5.11 atom_constraint.h File Reference

`#include <iostream.h>`

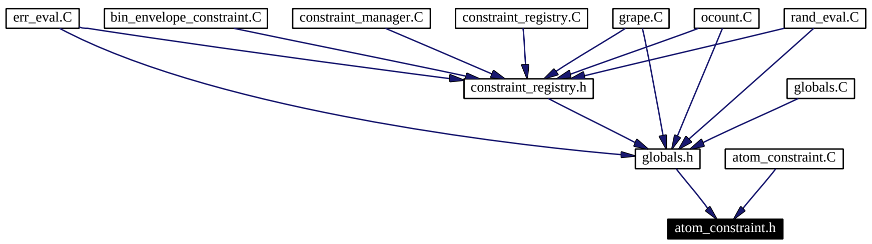
`#include "constraint.h"`

`#include "compoper.h"`

Include dependency graph for atom_constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **AtomConstraint**

5.12 atom_registry.C File Reference

5.13 atom_registry.h File Reference

`#include <signal.h>`

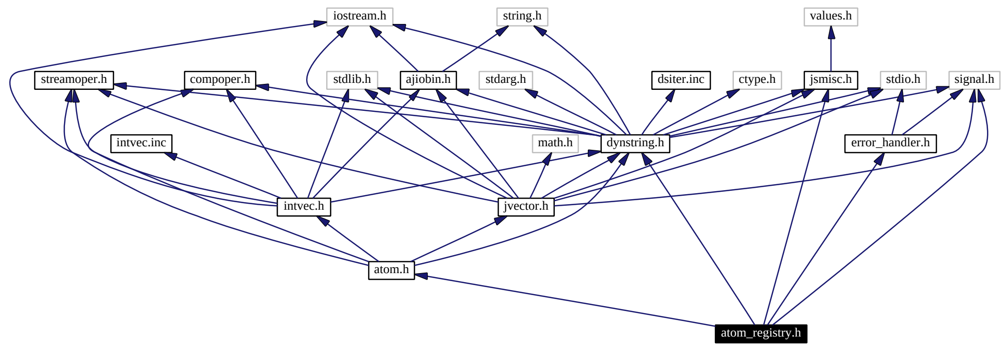
`#include "dynstring.h"`

`#include "atom.h"`

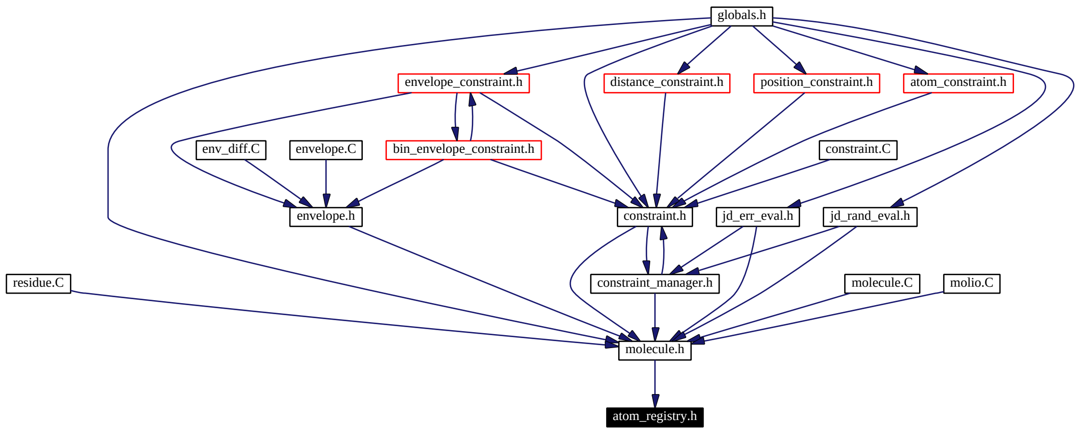
`#include "jsmisc.h"`

`#include "error_handler.h"`

Include dependency graph for atom_registry.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **AtomRegistry**

Defines

- #define **num_known_atom_species** 10

5.13.1 Define Documentation

5.13.1.1 #define num_known_atom_species 10

//

File: **atom_registry.h**

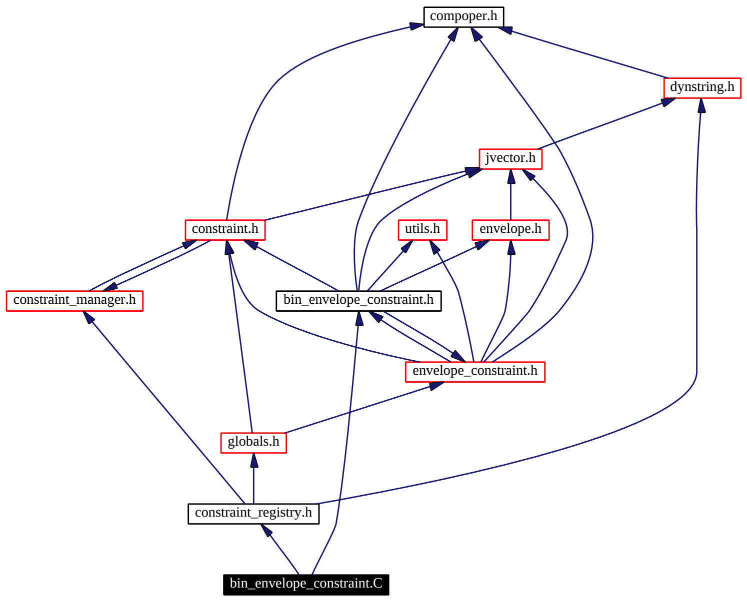
//

Definition at line 20 of file atom_registry.h.

5.14 bin_envelope_constraint.C File Reference

```
#include "bin_envelope_constraint.h"
#include "constraint_registry.h"
```

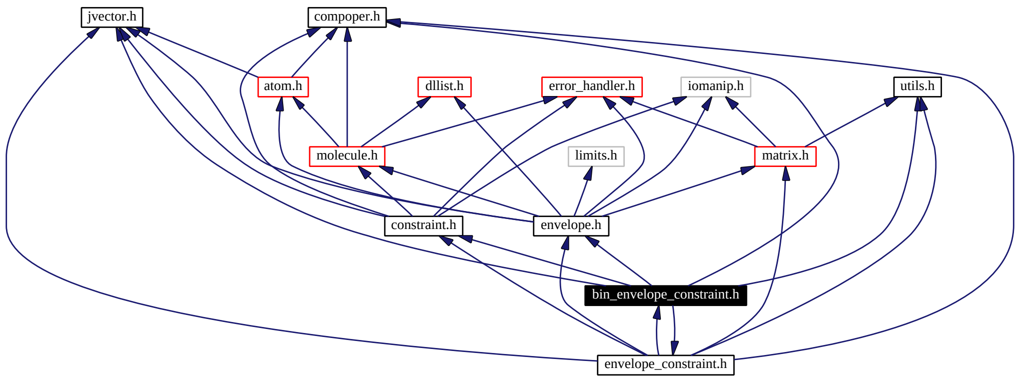
Include dependency graph for bin_envelope_constraint.C:



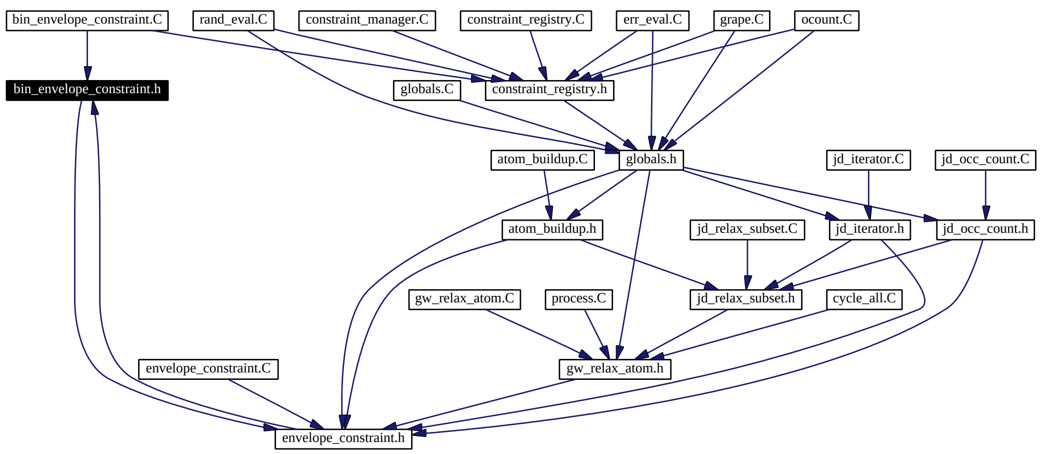
5.15 bin_envelope_constraint.h File Reference

```
#include "envelope.h"
#include "envelope_constraint.h"
#include "constraint.h"
#include "compoper.h"
#include "jvector.h"
#include "utils.h"
```

Include dependency graph for bin_envelope_constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **BinEnvelopeConstraint**

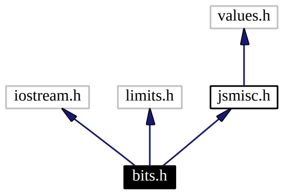
5.16 bits.h File Reference

`#include <iostream.h>`

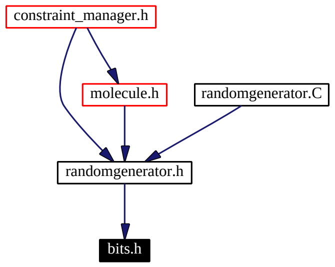
`#include <limits.h>`

`#include "jsmisc.h"`

Include dependency graph for bits.h:



This graph shows which files directly or indirectly include this file:



Functions

- **uint reverse_bits** (**uint** i)
- ostream& **print_bits** (ostream &os, **uint** v)

5.16.1 Function Documentation

5.16.1.1 ostream & print_bits (ostream & *os*, uint *v*) `[inline]`

Definition at line 25 of file bits.h.

```
00026 {
00027   uint mask=(uint)1<<31;
00028   for(uint i=1; i<=32; ++i)
00029     {
00030       os << ((v & mask) == 0) ? 0 : 1;
00031       v <<=1;
00032       if(i%8 == 0 && i != 32)
00033           os << ' ';
00034     }
00035
```

5.16.1.2 uint reverse_bits (uint *i*) [inline]

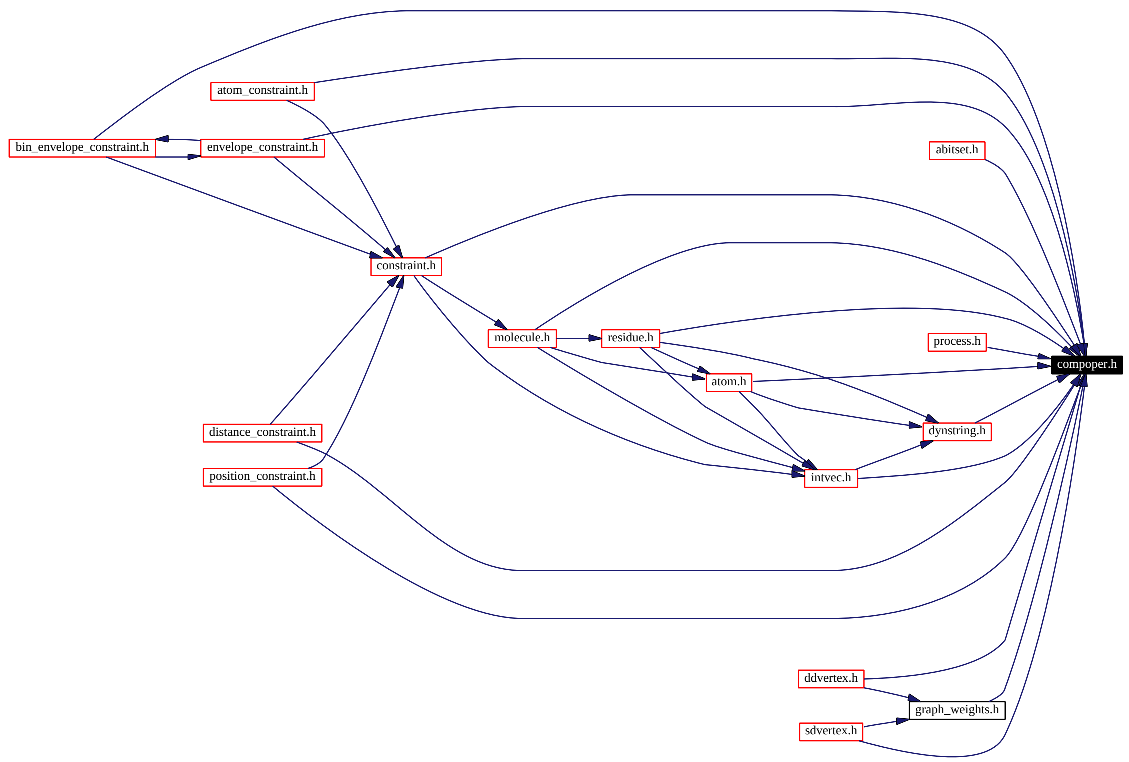
Definition at line 9 of file bits.h.

Referenced by RanBaseClass::Seed().

```
{
  uint rev=0;
  uint numbits = (uint) sizeof(uint)*8;
  int bit;

  for(uint j=0; j<numbits; ++j)
    {
      bit = i & 0x00000001;
      bit <<= numbits-j-1;
      rev |= bit;
      i >>= 1;
    }
```

5.17 compoper.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **MAKECOMPARISONS_NO_COMPARE**(CLASSTYPE_XX)
- #define **MAKECOMPARISONS_WITH_COMPARE**(CLASSTYPE_XX)

5.17.1 Define Documentation

5.17.1.1 #define MAKECOMPARISONS_NO_COMPARE(CLASSTYPE_XX)

Value:

```
bool    operator == (const CLASSTYPE_XX& co) const \
        { return this == &co; } \
bool    operator != (const CLASSTYPE_XX& co) const \
        { return this != &co; } \
bool    operator <= (const CLASSTYPE_XX& co) const \
        { return this <= &co; } \
bool    operator >= (const CLASSTYPE_XX& co) const \
        { return this >= &co; } \
bool    operator < (const CLASSTYPE_XX& co) const \
        { return this < &co; } \
bool    operator > (const CLASSTYPE_XX& co) const \
        { return this > &co; }
```

Definition at line 14 of file compoper.h.

5.17.1.2 #define MAKECOMPARISONS_WITH_COMPARE(CLASSTYPE_XX)

Value:

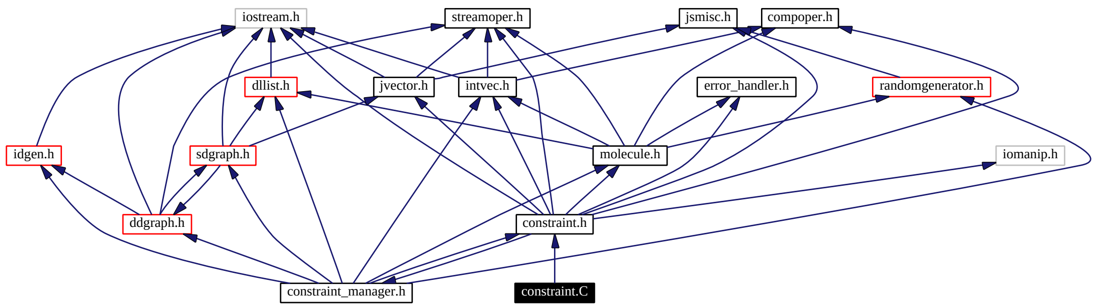
```
bool    operator == (const CLASSTYPE_XX& co) const \
        { return Compare(co) == 0; } \
bool    operator != (const CLASSTYPE_XX& co) const \
        { return Compare(co) != 0; } \
bool    operator <= (const CLASSTYPE_XX& co) const \
        { return Compare(co) <= 0; } \
bool    operator >= (const CLASSTYPE_XX& co) const \
        { return Compare(co) >= 0; } \
bool    operator < (const CLASSTYPE_XX& co) const \
        { return Compare(co) < 0; } \
bool    operator > (const CLASSTYPE_XX& co) const \
        { return Compare(co) > 0; }
```

Definition at line 28 of file compoper.h.

5.18 constraint.C File Reference

`#include "constraint.h"`

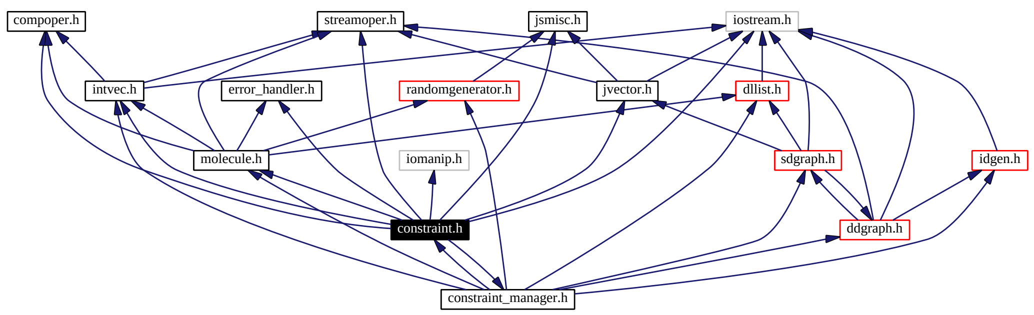
Include dependency graph for constraint.C:



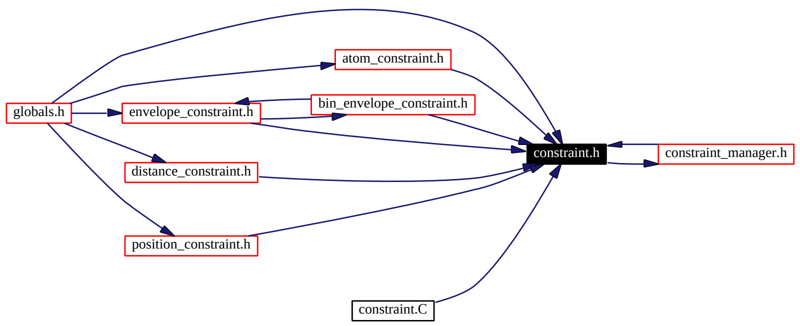
5.19 constraint.h File Reference

```
#include <iostream.h>
#include <iomanip.h>
#include "constraint_manager.h"
#include "molecule.h"
#include "jvector.h"
#include "intvec.h"
#include "compoper.h"
#include "streamoper.h"
#include "jsmisc.h"
#include "error_handler.h"
```

Include dependency graph for constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Constraint**

Defines

- #define **UNINITIALIZED_CONSTRAINT_ID** -1

- #define **__CONSTRAINT_DEFINED__**

5.19.1 Define Documentation

5.19.1.1 #define UNINITIALIZED_CONSTRAINT_ID -1

//

File: **constraint.h**

Abstract constraint class. All other (real) constraint types are classes that inherit from this class.

//

Definition at line 27 of file constraint.h.

5.19.1.2 #define __CONSTRAINT_DEFINED__

Value:

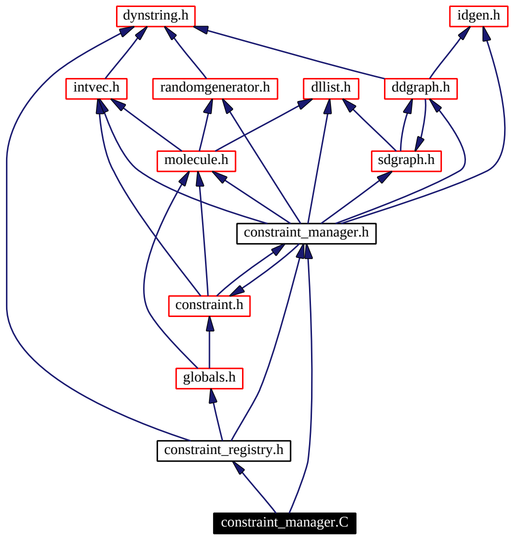
Definition at line 325 of file constraint.h.

5.20 constraint_manager.C File Reference

`#include "constraint_manager.h"`

`#include "constraint_registry.h"`

Include dependency graph for constraint_manager.C:



5.21 constraint_manager.h File Reference

`#include "dllist.h"`

`#include "intvec.h"`

`#include "molecule.h"`

`#include "ddgraph.h"`

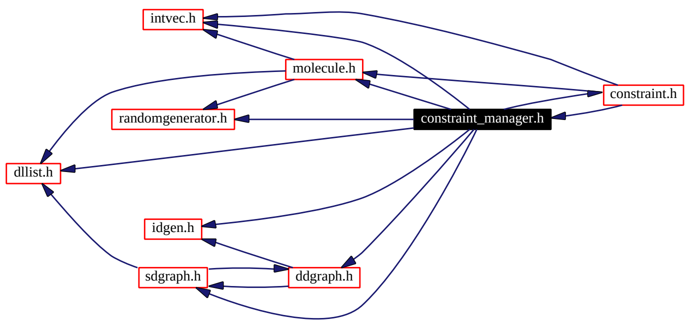
`#include "sdgraph.h"`

`#include "idgen.h"`

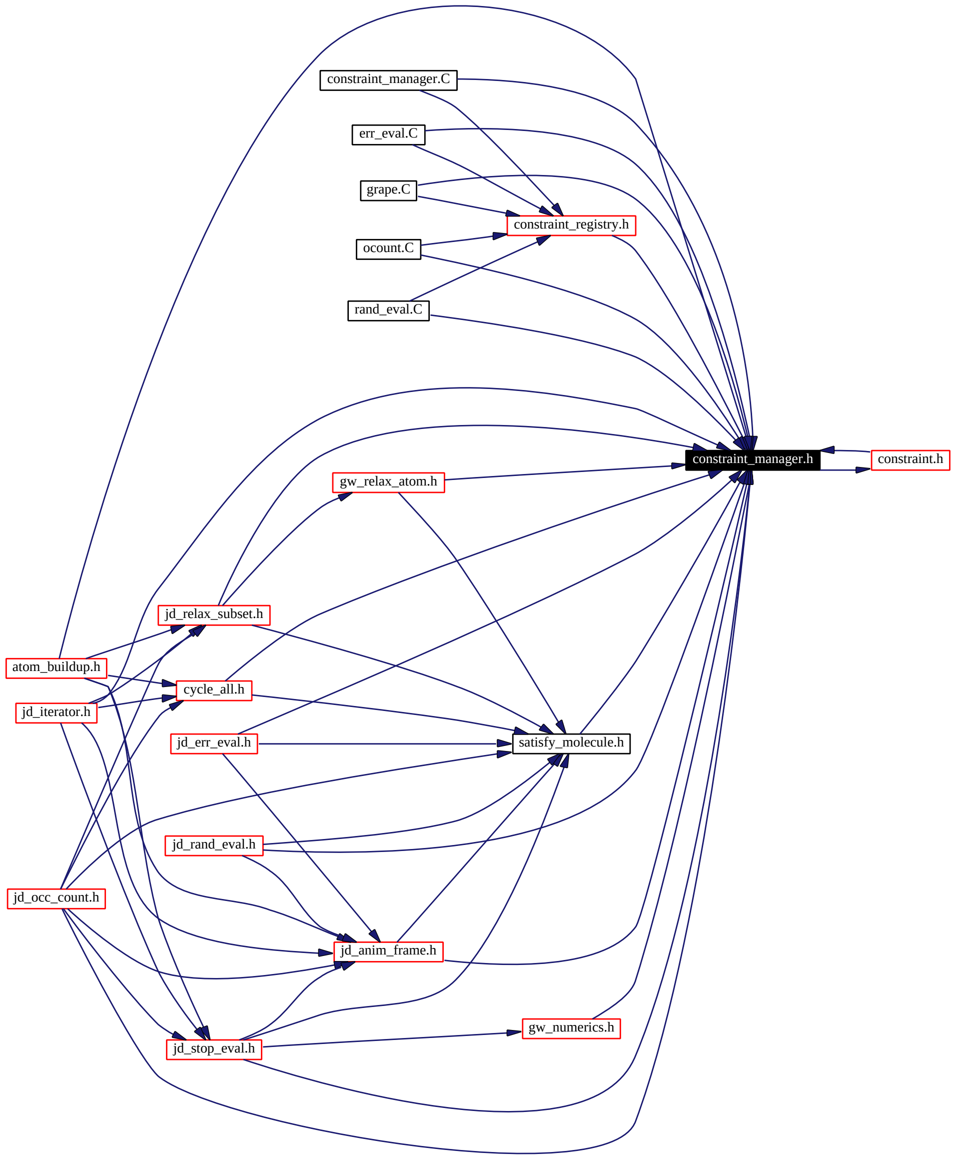
`#include "randomgenerator.h"`

`#include <constraint.h>`

Include dependency graph for constraint_manager.h:



This graph shows which files directly or indirectly include this file:



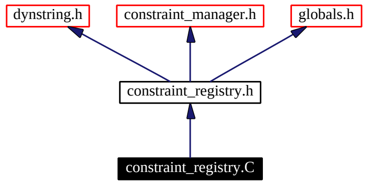
Compounds

- class **ConstraintManager**

5.22 constraint_registry.C File Reference

```
#include "constraint_registry.h"
```

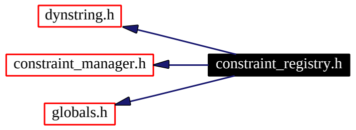
Include dependency graph for constraint_registry.C:



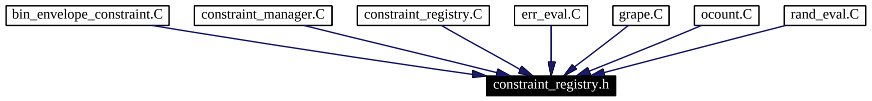
5.23 constraint_registry.h File Reference

```
#include "dynstring.h"
#include "constraint_manager.h"
#include "globals.h"
```

Include dependency graph for constraint_registry.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ConstraintRegistry**

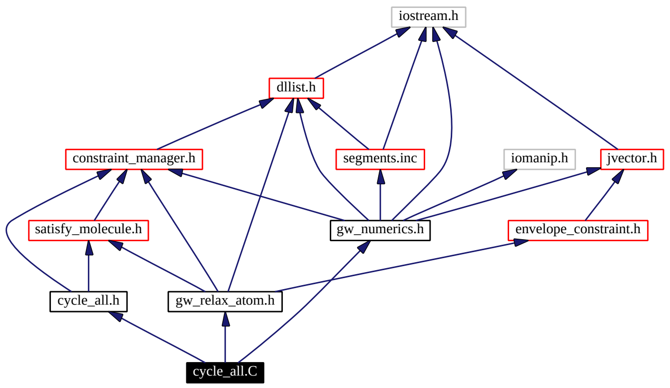
5.24 cycle_all.C File Reference

`#include "cycle_all.h"`

`#include "gw_numerics.h"`

`#include "gw_relax_atom.h"`

Include dependency graph for cycle_all.C:

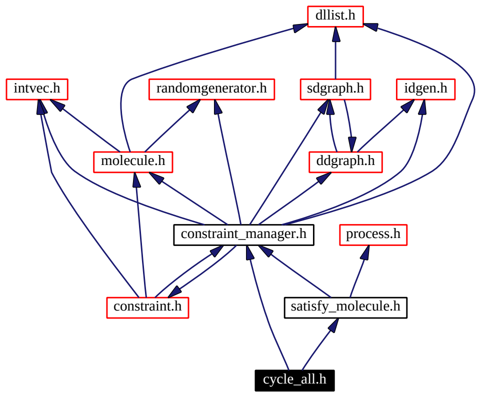


5.25 cycle_all.h File Reference

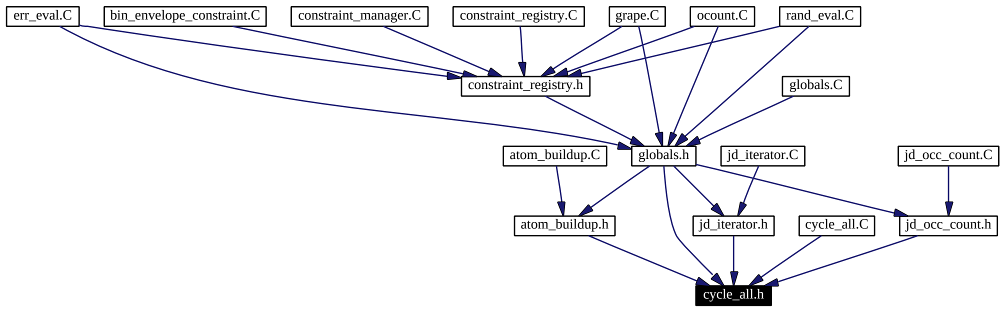
`#include "constraint_manager.h"`

`#include "satisfy_molecule.h"`

Include dependency graph for cycle_all.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **CycleAll**

5.26 ddgraph.h File Reference

`#include <iostream.h>`

`#include <map>`

`#include <signal.h>`

`#include "idgen.h"`

`#include "dynstring.h"`

`#include "search-sort.h"`

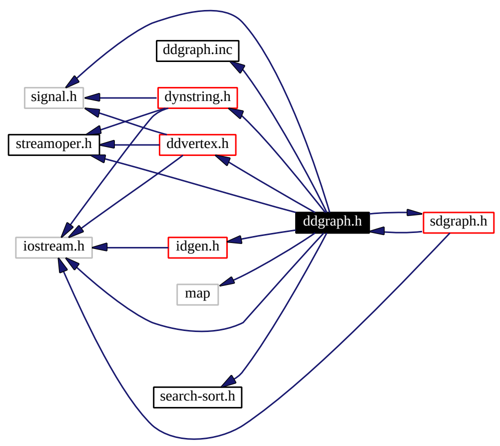
`#include "streamoper.h"`

`#include "ddvertex.h"`

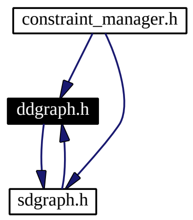
`#include "sdgraph.h"`

`#include <ddgraph.inc>`

Include dependency graph for ddgraph.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ddGraph**

Enumerations

- enum **GraphStatus** { **NoVertices** = -10 , **VertexBad** = - 9 , **VertexInG** = - 8 , **VertexNotInG** = - 7 , **VertexAddFailure** = - 6 , **NoEdges** = - 5 , **EdgeBad** = - 4 , **EdgeInG** = - 3 , **EdgeNotInG** = - 2 , **EdgeAddFailure** = - 1 }

Functions

- ostream& **PrintGStatus** (ostream &os, const **GraphStatus** &gs)

Variables

- const **DynString V_BEGIN_DELIM** = "{"
- const **DynString V_END_DELIM** = "}"
- const **DynString E_BEGIN_DELIM** = "["
- const **DynString E_END_DELIM** = "]"

5.26.1 Enumeration Type Documentation

5.26.1.1 enum GraphStatus

Enumeration values:

NoVertices

VertexBad

VertexInG

VertexNotInG

VertexAddFailure

NoEdges

EdgeBad

EdgeInG

EdgeNotInG

EdgeAddFailure

Definition at line 47 of file ddgraph.h.

```
{
        NoVertices              = -10 ,
        VertexBad               = - 9 ,
        VertexInG               = - 8 ,
        VertexNotInG            = - 7 ,
        VertexAddFailure        = - 6 ,
        NoEdges                 = - 5 ,
        EdgeBad                 = - 4 ,
        EdgeInG                 = - 3 ,
        EdgeNotInG              = - 2 ,
        EdgeAddFailure          = - 1

```

5.26.2 Function Documentation

5.26.2.1 ostream & PrintGStatus (ostream & *os*, const GraphStatus & *gs*) [inline]

Definition at line 1 of file ddgraph.inc.

```
{
  os << "<<";
  if(gs >= 0) os << "Operation successful.";
  else
    {
      switch(gs)
        {
          case NoVertices:
              os << "There were no vertices in G!";
              break;
          case VertexBad:
              os << "NULL vertex access attempted!";
              break;
          case VertexInG:
              os << "The vertex was already in G!";
              break;
          case VertexNotInG:
              os << "The vertex was not found in G!";
              break;
          case VertexAddFailure:
              os << "The vertex could not be added to G!";
              break;
          case NoEdges:
              os << "There were no edges in G!";
              break;
          case EdgeBad:
              os << "NULL edge access attempted!";
              break;
          case EdgeInG:
              os << "The edge was already in G!";
              break;
          case EdgeNotInG:
              os << "The edge was not found in G!";
              break;
          case EdgeAddFailure:
              os << "The edge could not be added to G!";
              break;
          default:
              os << "Some unkown error occurred!";
              break;
        }
     }

```

5.26.3 Variable Documentation

5.26.3.1 const DynString E_BEGIN_DELIM = "["

Definition at line 63 of file ddgraph.h.

5.26.3.2 const DynString E_END_DELIM = "]"

Definition at line 64 of file ddgraph.h.

5.26.3.3 const DynString V_BEGIN_DELIM = "{"

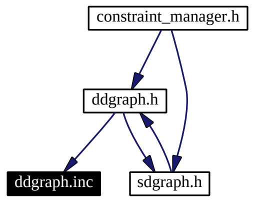
Definition at line 61 of file ddgraph.h.

5.26.3.4 const DynString V_END_DELIM = "}"

Definition at line 62 of file ddgraph.h.

5.27 ddgraph.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- ostream& **PrintGStatus** (ostream &os, const **GraphStatus** &gs)
- istream& **InputBinary** (istream &is, **ddGraph**< DT, WT > &ddg)
- ostream& **OutputBinary** (ostream &os, const **ddGraph**< DT, WT > &ddg)
- istream& **operator>=** (istream &is, **ddGraph**< DT, WT > &ddg)
- template<class DT, class WT> ostream& **operator<=** (ostream &os, const **ddGraph**< DT, WT > &ddg)

5.27.1 Function Documentation

5.27.1.1 istream & InputBinary (istream & *is*, ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 632 of file ddgraph.inc.

```
{
  sdGraph<DT,WT> G;
  if(InputBinary(is,G).bad())
      return is;
  ddg = G;
  return is;

```

5.27.1.2 ostream & OutputBinary (ostream & *os*, const ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 642 of file ddgraph.inc.

```
{
  sdGraph<DT,WT> G(ddg);
  return OutputBinary(os,G);

```

5.27.1.3 ostream& PrintGStatus (ostream & *os*, const GraphStatus & *gs*) `[inline]`

Definition at line 1 of file ddgraph.inc.

```
{
  os << "<<";
  if(gs >= 0) os << "Operation successful.";
  else
    {
      switch(gs)
        {
          case NoVertices:
              os << "There were no vertices in G!";
              break;
          case VertexBad:
              os << "NULL vertex access attempted!";
              break;
          case VertexInG:
              os << "The vertex was already in G!";
              break;
          case VertexNotInG:
              os << "The vertex was not found in G!";
              break;
          case VertexAddFailure:
              os << "The vertex could not be added to G!";
              break;
          case NoEdges:
              os << "There were no edges in G!";
              break;
          case EdgeBad:
              os << "NULL edge access attempted!";
              break;
          case EdgeInG:
              os << "The edge was already in G!";
              break;
          case EdgeNotInG:
              os << "The edge was not found in G!";
              break;
          case EdgeAddFailure:
              os << "The edge could not be added to G!";
              break;
          default:
              os << "Some unkown error occurred!";
              break;
        }
    }
```

5.27.1.4 template<class DT, class WT> ostream & operator<= (ostream & *os*, const ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 653 of file ddgraph.inc.

```
{ return OutputBinary(os,ddg); }
```

5.27.1.5 istream & operator>= (istream & *is*, ddGraph< DT, WT > & *ddg*) `[inline]`

Definition at line 649 of file ddgraph.inc.

```
{ return IntputBinary(is,ddg); }
```

5.28 ddvertex.h File Reference

`#include <set>`

`#include <signal.h>`

`#include <iostream.h>`

`#include "ajiobin.h"`

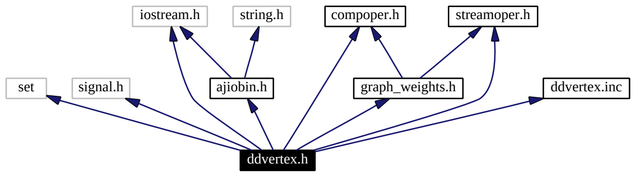
`#include "compoper.h"`

`#include "streamoper.h"`

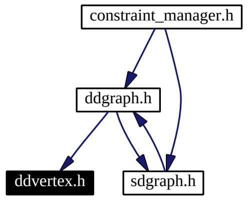
`#include "graph_weights.h"`

`#include <ddvertex.inc>`

Include dependency graph for ddvertex.h:



This graph shows which files directly or indirectly include this file:

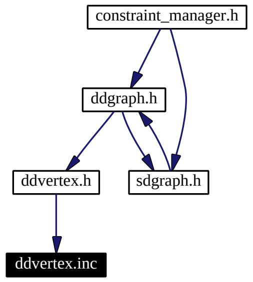


Compounds

- class **ddEdge**
- class **ddVertex**

5.29 ddvertex.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- template<class WT> ostream& **operator**<< (ostream &os, const **ddVertex**< WT > &vert)

5.29.1 Function Documentation

5.29.1.1 template<class WT> ostream & operator<< (ostream & *os*, const ddVertex< WT > & *vert*) `[inline]`

Definition at line 121 of file ddvertex.inc.

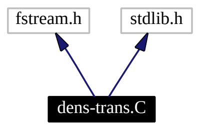
```
00122   {
00123     os << '\t' << vert.ID << '\n'
00124        << '\t' << '>' << ' ' << vert.OutE.size() << '\n';
00125     vert.PrintOutEdges(os);
00126     os << '\t' << '<' << ' ' << vert.InE.size()  << '\n';
00127     vert.PrintInEdges(os);
00128     return os;
00129
```

5.30 dens-trans.C File Reference

#include <fstream.h>

#include <stdlib.h>

Include dependency graph for dens-trans.C:



Functions

- int **main** (int argc, char **argv)

5.30.1 Function Documentation

5.30.1.1 int main (int *argc*, char ** *argv*)

Definition at line 4 of file dens-trans.C.

```
00004                                 {
00005
00006   if (argc != 2) {
00007     cerr << "must supply the filename as arg 1" << endl;
00008     exit (1);
00009   }
00010
00011   ifstream in(argv[1]);
00012
00013   if (!in){
00014     cerr << "Unable to open " << argv[1] << endl;
00015     exit (2);
00016   }
00017
00018   char d[255];
00019   char temp[255];
00020
00021   int xdim, ydim, zdim;
00022   double minx, miny, minz;
00023   double maxx, maxy, maxz;
00024
00025   in.getline(d,255); // # AVS
00026   cout << d << endl;
00027   in.getline(d,255); // ndim
00028   cout << d << endl;
00029   //dim1
00030   in.ignore(5);
00031   in >>  xdim;
00032   cout << "xdim= "<< xdim << endl;
00033   //dim2
00034   in.ignore(6); // +\n
00035   in >>  ydim;
00036   cout << "ydim= "<< ydim << endl;
00037   //dim3
00038   in.ignore(6); // +\n
00039   in >>  zdim;
```

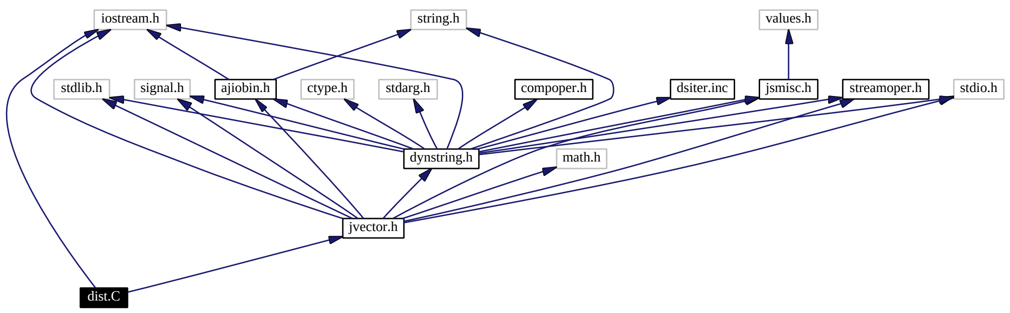
```
cout << "zdim= "<< zdim << endl;

cerr << "dim: " << xdim << ", "<< ydim << ", "<< zdim << endl;

in.getline(d,255); // \n

in.getline(d,255); // nspace
cout << d << endl;
in.getline(d,255); // vclean
cout << d << endl;
in.getline(d,255); // data
cout << d << endl;
in.getline(d,255); // field
cout << d << endl;

// mins
in.ignore (9);
in >>  minx >> miny >> minz;
cout << "min_ext= " << minx << " " << miny << " " << minz << endl;
cerr << "min: " <<  minx << ", " << miny << ", " << minz << endl;

// maxs
in.ignore (10);
in >>  maxx >> maxy >> maxz;
cout << "max_ext= " << maxx << " " << maxy << " " << maxz << endl;
cerr << "max: "  <<  maxx << ", " << maxy << ", " << maxz << endl;

in.getline(d,255); // nspace

in.getline(d,255); // # end of header
cout << d << endl;
in.ignore(2); // ^L^L  #!!##! WHATEVER -- drop these;

float val;
long read = 0;

for (int zh =0;zh<zdim ;zh++){
  for (int yh =0;yh<ydim ;yh++){
    for (int xh =0;xh<xdim ;xh++){
      in.read(&val,sizeof(float));
      if (!in.good()){
        cerr << "reading error!\n read "<< read << " entries, expected "<< xdim*ydim*zdim << endl;
        exit(3);
      }
      read++;
      cout << val << " ";
    }
  }
}

```

5.31 dist.C File Reference

```
#include <iostream.h>
```

```
#include "jvector.h"
```

Include dependency graph for dist.C:



Functions

- int **main** (int argc, char $**$argv)

5.31.1 Function Documentation

5.31.1.1 int main (int *argc*, char $**$ *argv*)

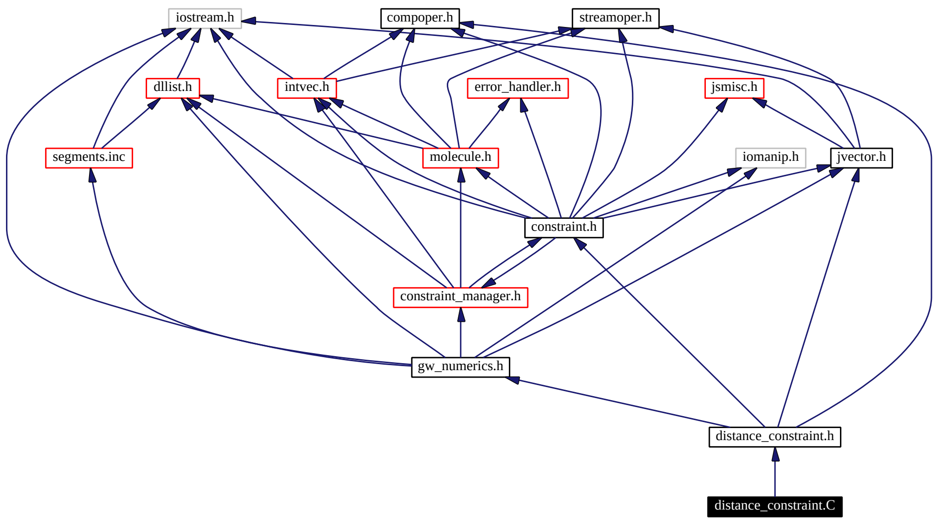
Definition at line 5 of file dist.C.

```
{
  jVector v1;
  jVector v2;

  while (1){

    cout << "Vector1:";
    cin >> v1;
    cout << "Vector2:";
    cin >> v2;
    cout << "dist= " << euclid(v1,v2) << endl << endl;


  }

```

5.32 distance_constraint.C File Reference

`#include "distance_constraint.h"`

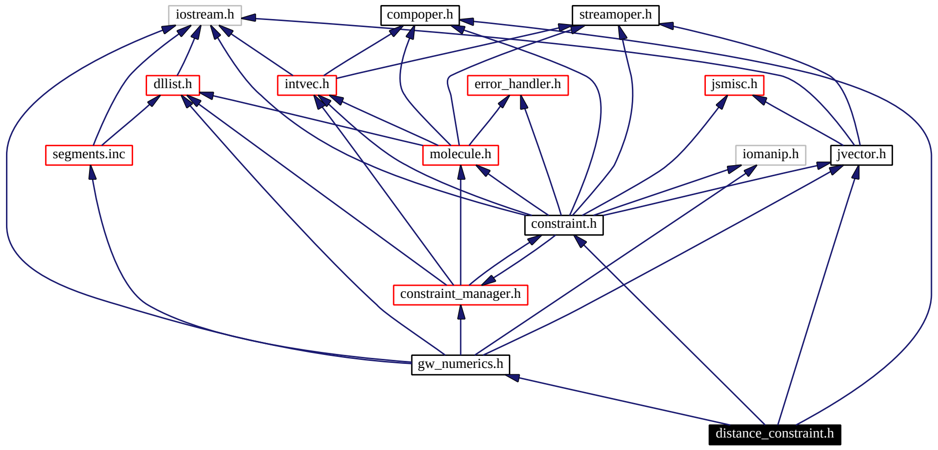
Include dependency graph for distance_constraint.C:



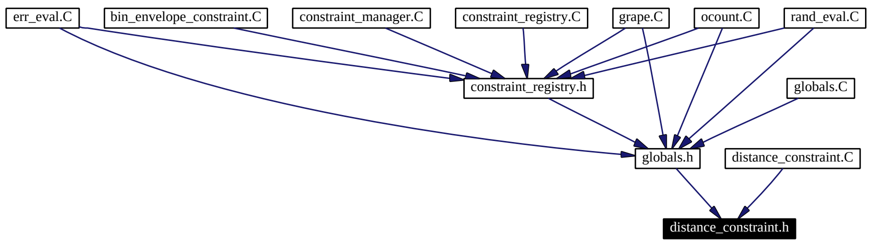
5.33 distance_constraint.h File Reference

```
#include "constraint.h"
#include "compoper.h"
#include "jvector.h"
#include "gw_numerics.h"
```

Include dependency graph for distance_constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **DistanceConstraint**

5.34 dllist.h File Reference

`#include <iostream.h>`

`#include <signal.h>`

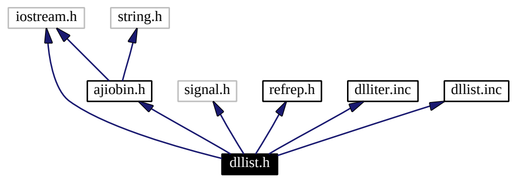
`#include "refrep.h"`

`#include "ajiobin.h"`

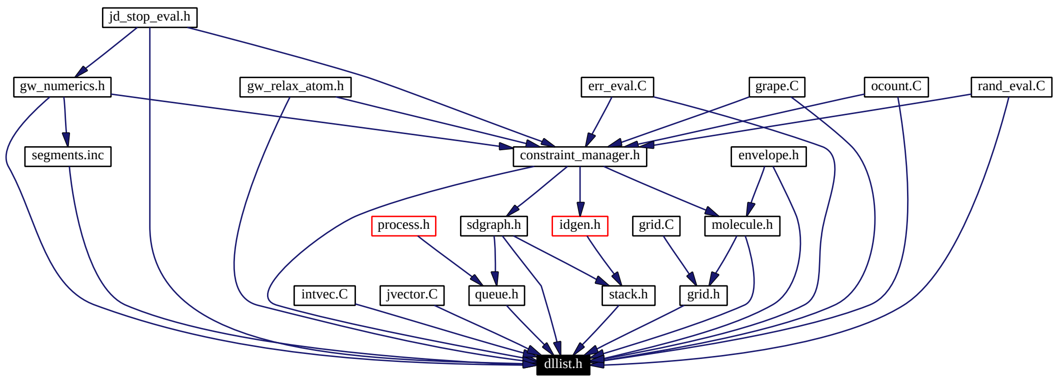
`#include <dlliter.inc>`

`#include "dllist.inc"`

Include dependency graph for dllist.h:



This graph shows which files directly or indirectly include this file:

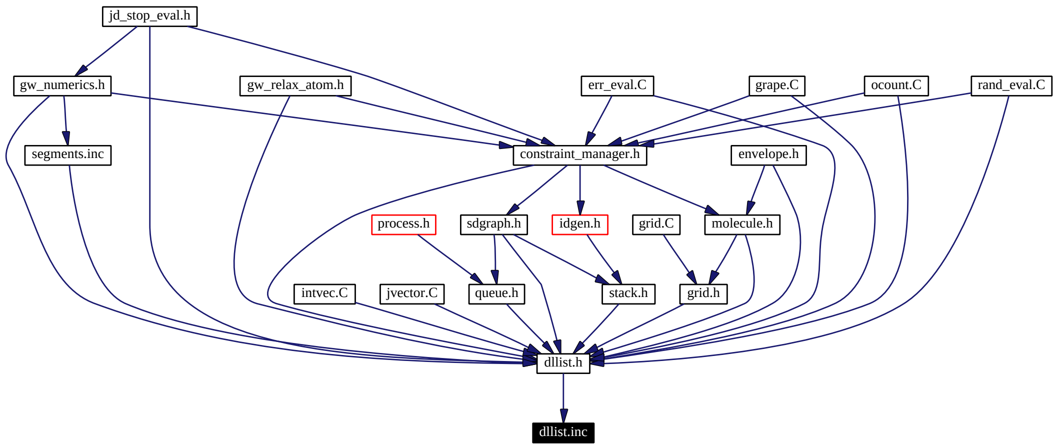


Compounds

- class **DLList**
- class **DLListNode**
- class **DLListRep**

5.35 dllist.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- template<class T> ostream& **operator<<** (ostream &os, const **DLList**< T > &dll)
- istream& **InputBinary** (istream &is, **DLList**< T > &dll)
- ostream& **OutputBinary** (ostream &os, const **DLList**< T > &dll)
- istream& **operator>>=** (istream &is, **DLList**< T > &dll)
- template<class T> ostream& **operator<<=** (ostream &os, const **DLList**< T > &dll)

5.35.1 Function Documentation

5.35.1.1 istream & InputBinary (istream & *is*, DLList< T > & *dll*) `[inline]`

Definition at line 389 of file dllist.inc.

```
{
  // Delete everything:
  //
  dll.Clear();

  // Read in the number of elements:
  //
  int numel;
  if(InputBinary(is,numel).bad())
      return is;

  T stuff;
  for(int i=0; i<numel; ++i)
    {
      if(InputBinary(is,stuff).bad())
          return is;
      dll.AddToBack(stuff);
    }

```

5.35.1.2 ostream & OutputBinary (ostream & *os*, const DLList< T > & *dll*) `[inline]`

Definition at line 412 of file dllist.inc.

```
{
  // Write the number of elements:
  //
  if(OutputBinary(os, dll.Number()).bad())
     return os;

  for(DLList<T>::Iterator it = dll.GetIterator(); it; ++it)
     if(OutputBinary(os, *it).bad())
        return os;
  return os;

```

5.35.1.3 template<class T> ostream & operator<< (ostream & *os*, const DLList< T > & *dll*) `[inline]`

Definition at line 385 of file dllist.inc.

```
{ return dll.Print(os); }

```

5.35.1.4 template<class T> ostream & operator<= (ostream & *os*, const DLList< T > & *dll*) `[inline]`

Definition at line 431 of file dllist.inc.

```
{ return OutputBinary(os,dll); }

```

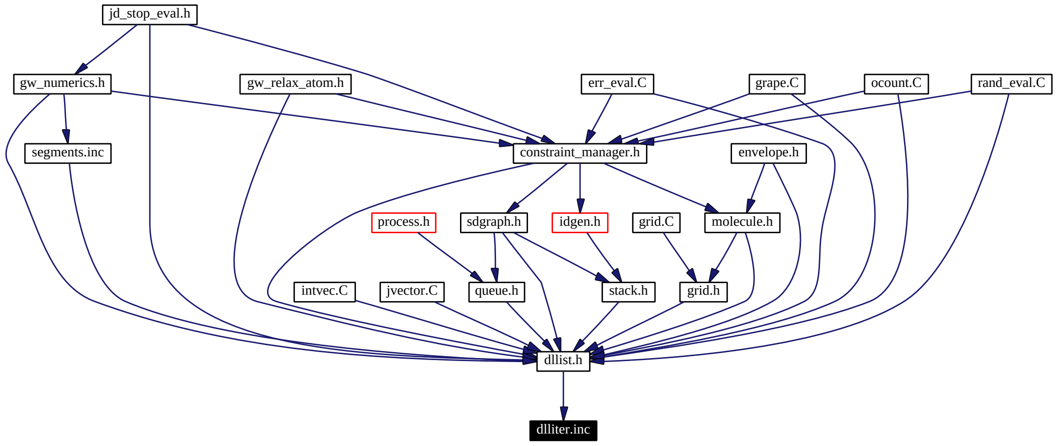
5.35.1.5 istream & operator>= (istream & *is*, DLList< T > & *dll*) `[inline]`

Definition at line 427 of file dllist.inc.

```
{ return InputBinary(os,dll); }

```

5.36 dlliter.inc File Reference

This graph shows which files directly or indirectly include this file:



Compounds

- class **ConstIterator**
- class **Iterator**

5.37 dsiter.inc File Reference

This graph shows which files directly or indirectly include this file:

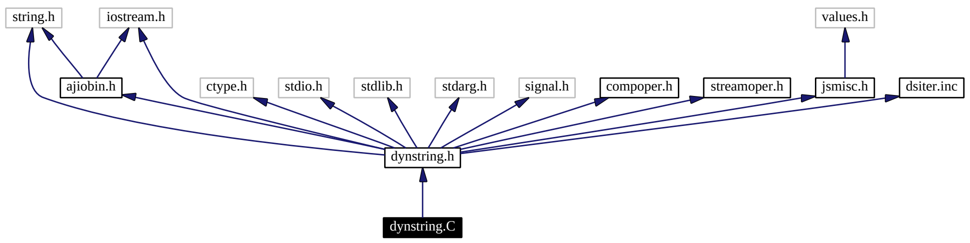
Compounds

- class **DSIterator**

5.38 dynstring.C File Reference

`#include "dynstring.h"`

Include dependency graph for dynstring.C:



Functions

- void **SetDSDelim** (const **DynString** &delim)
- void **AddDSDelim** (char delim)
- void **RemDSDelim** (char delim)
- **DynString GetDSDelim** ()
- **DynString IntToDS** (int i)
- **DynString Reproduce** (char c, int n)

5.38.1 Function Documentation

5.38.1.1 void AddDSDelim (char *delim*)

Definition at line 45 of file dynstring.C.

```
{ DynString::Delimiter[(uint) delim] = 1; }

```

5.38.1.2 DynString GetDSDelim ()

Definition at line 51 of file dynstring.C.

Referenced by DSIterator::DSIterator(), jVector::dynread(), and IntVec::dynread().

```
{
  DynString delim;
  delim.DestResizeToFit(256);
  int j=0;
  for(uint i=0; i<256; ++i)
    {
      if(DynString::Delimiter[i])
        {
          delim.innards[j] = (char) i;
          ++j;
        }
    }

```

5.38.1.3 DynString IntToDS (int *i*)

Definition at line 351 of file dynstring.C.

Referenced by jdIterator::run(), and AtomBuildup::run().

```
{
  char s[100]; s[0] = '\0';
  sprintf(s, "%i", i);
  DynString ds(s);
  return ds;

```

5.38.1.4 void RemDSDelim (char *delim*)

Definition at line 47 of file dynstring.C.

```
{ DynString::Delimiter[(uint) delim] = 0; }

```

5.38.1.5 DynString Reproduce (char *c*, int *n*)

Definition at line 745 of file dynstring.C.

```
{
  DynString str;
  if(!str.DestResizeToFit(n)) return str;
  for(int i=0; i<n; ++i) str.innards[i] = c;
  str.innards[n] = '\0';
  str.L = n;
  return str;

```

5.38.1.6 void SetDSDelim (const DynString & *delim*)

Definition at line 34 of file dynstring.C.

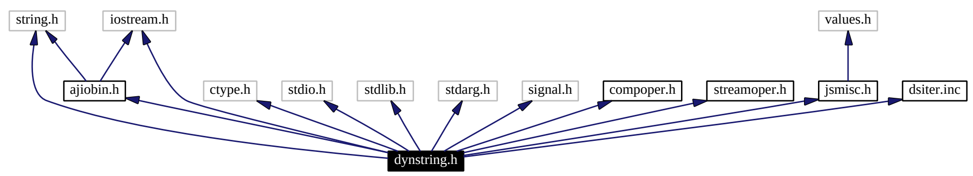
Referenced by jVector::dynread(), and IntVec::dynread().

```
{
  uint j;
  for(int i=0; i<256; ++i) DynString::Delimiter[i] = 0;
  for(int i=0; i<delim.L; ++i)
    {
      j = (uint) delim.innards[i];
      DynString::Delimiter[j] = 1;
    }

```

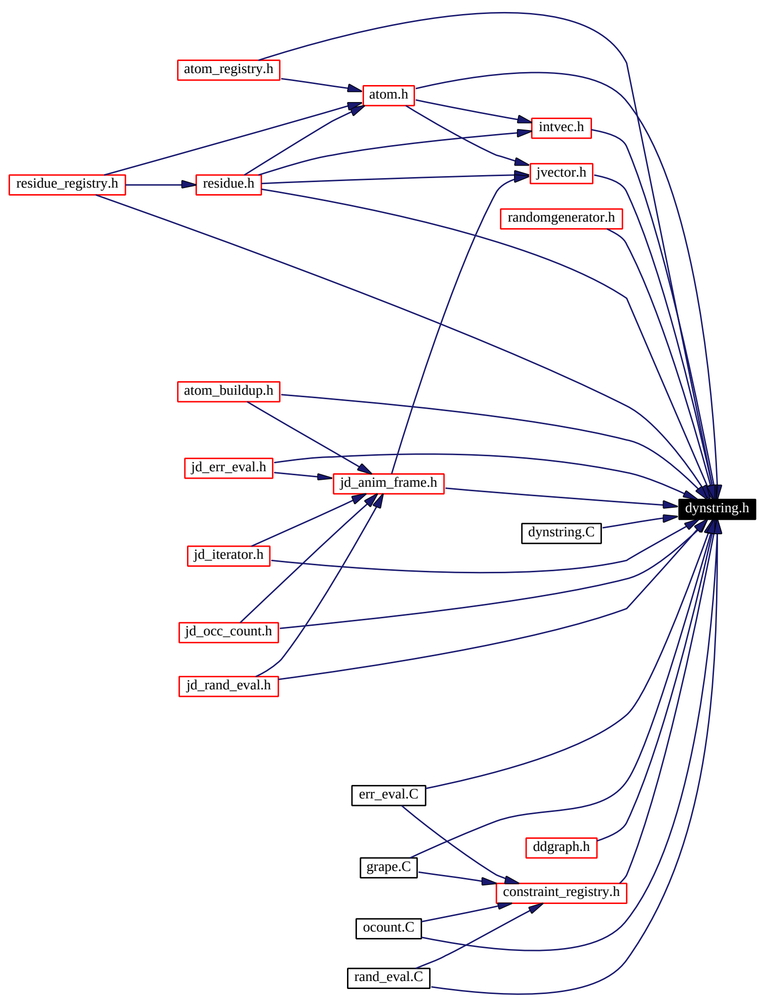
5.39 dynstring.h File Reference

```
#include <string.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <signal.h>
#include <iostream.h>
#include "compoper.h"
#include "streamoper.h"
#include "jsmisc.h"
#include "ajiobin.h"
#include <dsiter.inc>
```

Include dependency graph for dynstring.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **DynString**

Typedefs

- typedef **DynString DS**

5.39.1 Typedef Documentation

5.39.1.1 typedef DynString DS

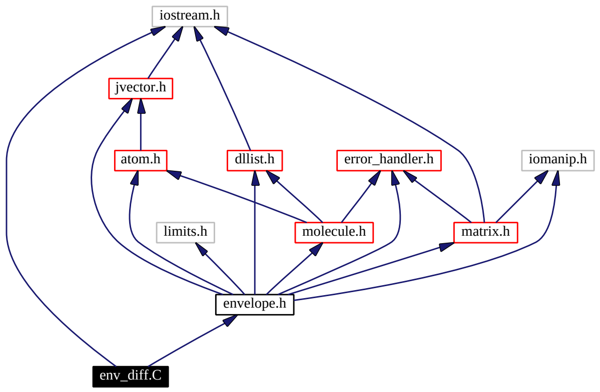
Definition at line 247 of file dynstring.h.

5.40 env_diff.C File Reference

`#include <iostream.h>`

`#include "envelope.h"`

Include dependency graph for env_diff.C:



Functions

- int **main** (int argc, char **argv)

5.40.1 Function Documentation

5.40.1.1 int main (int *argc*, char ** *argv*)

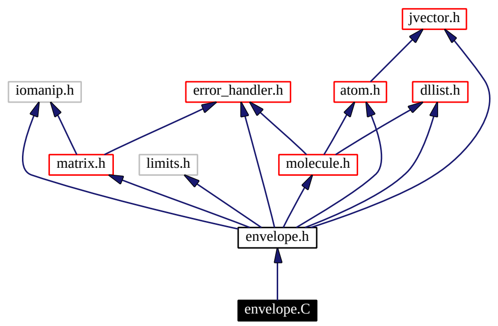
Definition at line 6 of file env_diff.C.

```
00006                                  {
00007
00008   if(argc != 3)
00009     {
00010       cerr << "Usage: env-diff file file\n";
00011       return 0;
00012     }
00013
```

5.41 envelope.C File Reference

`#include "envelope.h"`

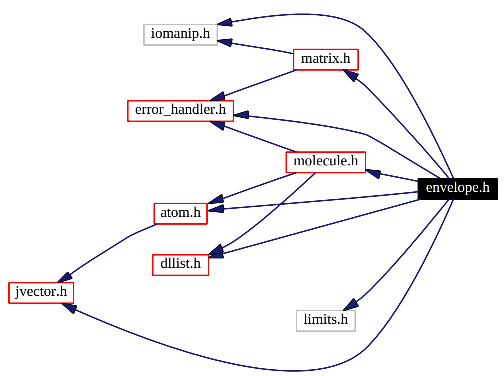
Include dependency graph for envelope.C:



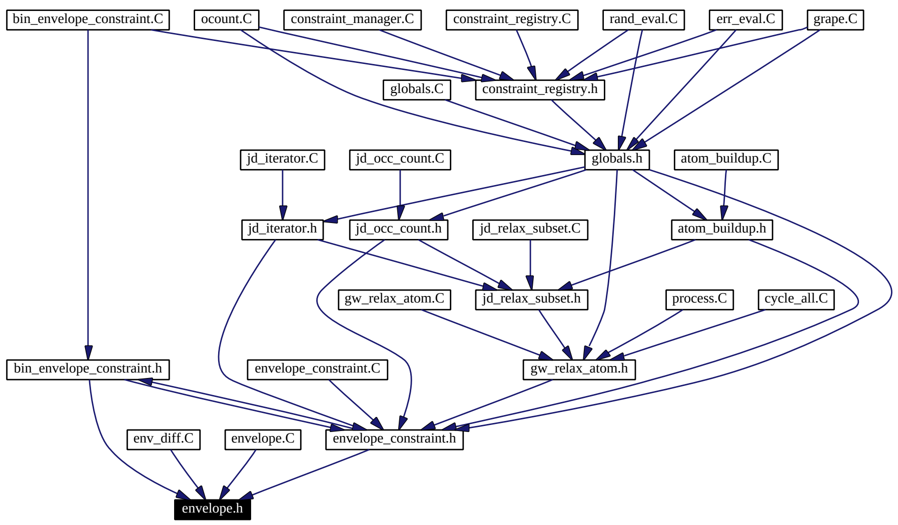
5.42 envelope.h File Reference

```
#include <limits.h>
#include <iomanip.h>
#include "atom.h"
#include "molecule.h"
#include "jvector.h"
#include "dllist.h"
#include "matrix.h"
#include "error_handler.h"
```

Include dependency graph for envelope.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Envelope**

Defines

- #define **MAX_ENV_DEN** 1.
- #define **MIN_ENV_DEN** 0.
- #define **ENV_DEN_RANGE** 1.
- #define **DEFAULT_CUBE_WIDTH** 3.7

5.42.1 Define Documentation

5.42.1.1 #define DEFAULT_CUBE_WIDTH 3.7

Definition at line 29 of file envelope.h.

5.42.1.2 #define ENV_DEN_RANGE 1.

Definition at line 17 of file envelope.h.

5.42.1.3 #define MAX_ENV_DEN 1.

//

envelope.h

This is a surface envelope class. It holds the points and data for a molucular envelope.

//

Definition at line 15 of file envelope.h.

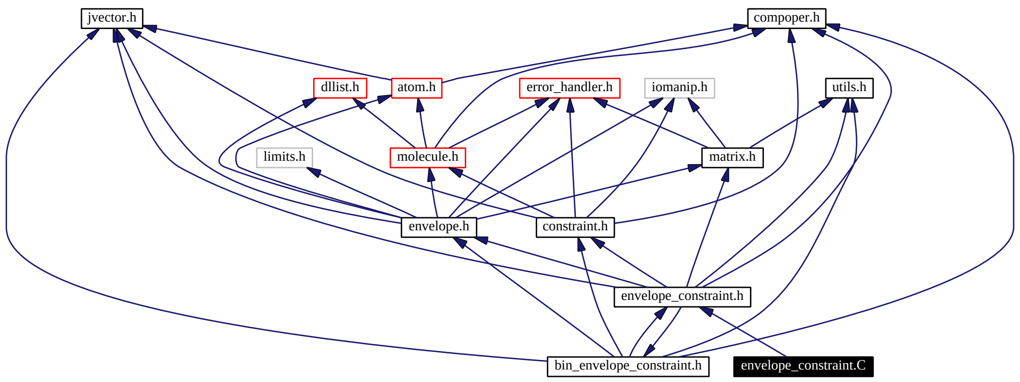
5.42.1.4 #define MIN_ENV_DEN 0.

Definition at line 16 of file envelope.h.

5.43 envelope_constraint.C File Reference

`#include "envelope_constraint.h"`

Include dependency graph for envelope_constraint.C:



5.44 envelope_constraint.h File Reference

`#include "envelope.h"`

`#include "bin_envelope_constraint.h"`

`#include "constraint.h"`

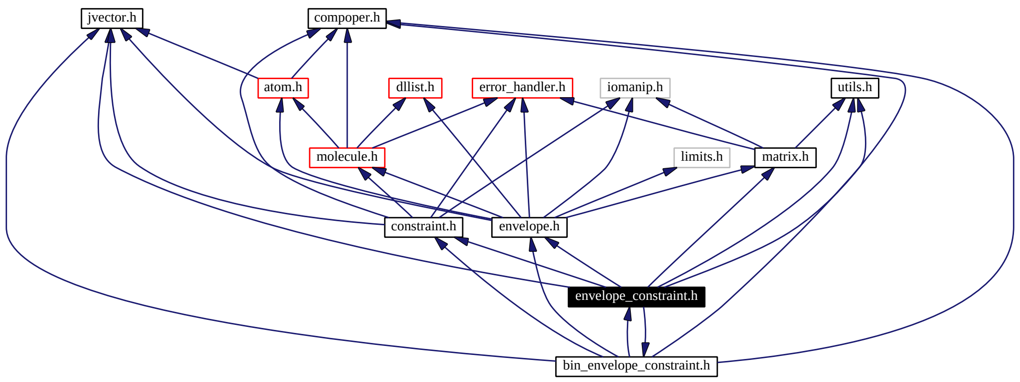
`#include "compoper.h"`

`#include "jvector.h"`

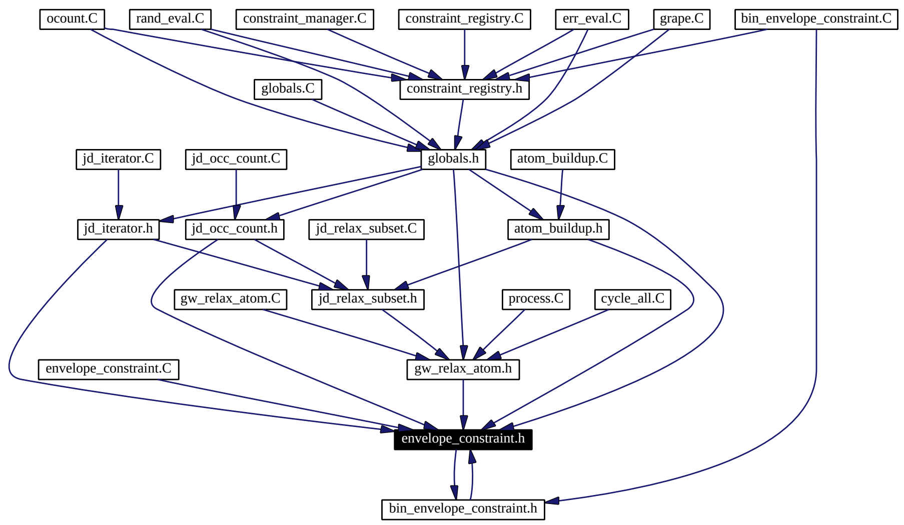
`#include "matrix.h"`

`#include "utils.h"`

Include dependency graph for envelope_constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **EnvelopeConstraint**

5.45 err_eval.C File Reference

`#include <fstream.h>`

`#include "dynstring.h"`

`#include "dllist.h"`

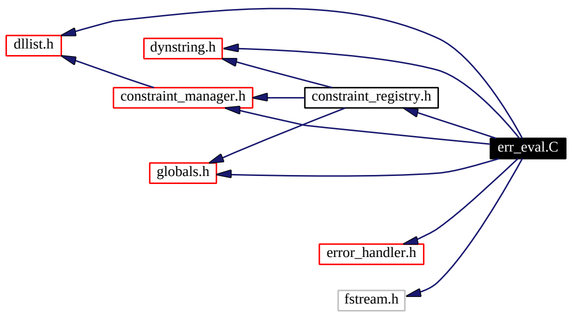
`#include "constraint_manager.h"`

`#include "constraint_registry.h"`

`#include "globals.h"`

`#include "error_handler.h"`

Include dependency graph for err_eval.C:



Functions

- void **print_usage** (ostream &os)
  Prints information about usage.
- void **print_help** (ostream &os)
  Prints full help.
- int **main** (int argc, char **argv)

5.45.1 Function Documentation

5.45.1.1 int main (int *argc*, char ** *argv*)

Definition at line 17 of file err_eval.C.

```
00017                                  {
00018   if(argc < 2)
00019     {
00020       print_usage(cout);
00021       return 1;
00022     }
00023
00024   cerr << "Begin: ";
00025   for (int i =0; i < argc; i++){
00026     cerr << argv[i]<<" ";
00027   }
```

```
cerr << endl;

DLList<DynString> ArgList;

DLList<DynString> ConstraintFileList;

Molecule * predictedMolecule=0;

DynString arg;

DynString positionFile = "";

ifstream * XmolInputFile=0;


ifstream * input=0;

int argStart = 1;

DynString BaseFN = "evalenv-output";

int skip = 1;

// See if the user has specified a parameter file.  It is a
// parameter file if the first argument does *not* start with
// a dash, '-' (i.e. it is not a flag).
//

if(argv[1][0] != '-')
  {
    // Read in the arguments from the parameter file.  This file should
    // contain flags (including their dashes) followed immediately by
    // the values associated with them.  Warning: this processing step
    // ignores white space!
    //
    if(!(input = new ifstream(argv[1])))
      {
        cerr << "Could not open/find parameter file, " << argv[1]
             << ", quitting. Try -help option.\n";
        return 2;
      }
    while(!input->eof() && !input->bad())
      {
        *input >> arg;

        if(arg[0] == '#')
          arg.EatLine(*input);
        // Ignore lines that begin with a pound, '#', sign:
        else
          ArgList.AddToBack(arg);
      }
    ArgList.RemoveFromBack();  // Get rid of the eof character.

    delete input; input=0;
    argStart=2;
  }

// Copy the arguments from the command line onto our argument list:
//
for(int a=argStart; a<argc; ++a)
  ArgList.AddToBack(DynString(argv[a]));

while(ArgList)
  {
    arg = ArgList.RemoveFromFront();
```

```

    if((arg == DynString("-residues")) ||
            (arg == DynString("-res"))) {

      // Check to see if we've already read in a molecule:
        if(predictedMolecule)
          {
            cerr << "Warning: More than one file has been supplied for "
                 << "the initial molecule.\nWill use the molecule "
                 << "specified in the file listed last.\n";
            delete predictedMolecule; predictedMolecule=0;
          }
        arg = ArgList.RemoveFromFront();  // Get number of atoms
        if((input = new ifstream(arg.GetData())))
          {
            predictedMolecule = new Molecule;

            if(predictedMolecule->readFasta(*input)<2)
              {
                cerr << "Error generating molecule from residue sequence.\n";
                delete predictedMolecule; predictedMolecule=0;
              }
            delete input;input = 0;
          } else {
            cerr << "Could not open initial FASTA file, " << arg
                 << ", will try default location.\n";

          }
    }

    // Initial atom positions
    //
    else if((arg == DynString("-initial")) ||
            (arg == DynString("-init")) ||
            (arg == DynString("-i"))) {

      positionFile = ArgList.RemoveFromFront();  // Get the name of the xmol file.

    }


//   while(ArgList)
//     {
//       arg = ArgList.RemoveFromFront();
//
//       // Initial molecule file is being specified.
//       //
//
//       if((arg == DynString("-initial")) ||
//       (arg == DynString("-init")) ||
//       (arg == DynString("-i")))
//      {
//        // Check to see if we've already read in a molecule:
//        if(predictedMolecule)
//          {
//            cerr << "Warning: More than one file has been supplied for "
//                 << "the initial molecule.\nWill use the molecule "
//                 << "specified in the file listed last.\n";
//            delete predictedMolecule; predictedMolecule=0;
//          }
//
//        arg = ArgList.RemoveFromFront();  // Get the name of the xmol file.
//        if((XmolInputFile = new ifstream(arg.GetData())))
//          {
//            predictedMolecule = new Molecule;
```

```
//            if(predictedMolecule->readXmol(*XmolInputFile)<1)
//              {
//                cerr << "Error reading initial molecule.\n";
//                delete predictedMolecule; predictedMolecule=0;
//              }
//          }
//        else
//          {
//            cerr << "Could not open initial molecule file, " << arg
//                 << ", will try default location.\n";
//          }
//      }

      else if(arg == DynString("-f") ||
              arg == DynString("-filebase"))
        {
            arg = ArgList.RemoveFromFront();
            if (arg[0] != '-'){
              BaseFN = arg;
            } else {
              DIE("Confused by filename ");
            }
        }

      else if(arg == DynString("-skip"))
        {
          arg = ArgList.RemoveFromFront();
          if (arg[0] != '-'){
            skip = arg.ToInt();
          } else {
            DIE("Confused by filename ");
          }
        }

      else if((arg == DynString("-constraint")) ||
              (arg == DynString("-constraints")) ||
              (arg == DynString("-c")))
        {
          // Just add this file to the list of files that contain
          // constraints for right now.  We'll read these in once
          // we're sure we have an initial molecule.
          //
          arg = ArgList.RemoveFromFront();
          ConstraintFileList.AddToBack(arg);
        }


      else if(arg == DynString("-help"))
        {
          print_help(cout);
          return 1;

        }

      // Ignore the bogus parameter.
      //
      else
        {
          cerr << "Unknown argument, " << arg << ", given, ignoring.\n";
        }
    }

  // If we havent read in an initial molecule, we'll check the default
  // location for one:
  //
```

```
if(!predictedMolecule)
  {
    cerr << "No initial molecule specified (-residues), quitting. Try -help option.\n";
    return 3;
  }


if (positionFile != "") {

  if((XmolInputFile = new ifstream(positionFile.GetData())))
    {
      if(predictedMolecule->readXmol(*XmolInputFile)<2)
        {
          cerr << "Error reading initial positions from " << positionFile << ".  Check the file.\n";
          delete predictedMolecule; predictedMolecule=0;
          return 4;
        }
    } else    {
      cerr << "Could not open initial molecule position file: " << positionFile << ".\n";
      return 5;
    }

} else {
  cerr << " You must use -i to specify initial positions" << endl;
  return 6;
}

// Now that we've got an actual molecule, we can create a constraint
// manager and read in the constraints:
//

ConstraintManager CM(*predictedMolecule);

//
//  cerr << " WARNING WARNING WARNING -- perturbing inputs \n\n" << endl;
//  CM.RandomPerturb(.2);
//

int num_read = 0;

while(ConstraintFileList)
  {
    // Get the filename and then open the file and then read the
    // constraints in using the constraint manager.
    arg = ConstraintFileList.RemoveFromFront();
    input = new ifstream(arg.GetData());
    if(input->good())
      {
        // Use the registry's routine to read in constraints and
        // append them to the constraint manager.
        //

        num_read += ConstraintRegistry::readConstraints(*input,CM);
        delete input; input=0;
      } else {
        cerr << "\n\nWhile reading file: " << arg.GetData() << endl;
        DIE ("Cannot open file");
      }

  }

if (num_read == 0) {
  WARN ("You have no constraints");
}

```

```
00296
00297   jdErr_Eval myeval(CM, XmolInputFile, skip);
00298   myeval.BaseFN = BaseFN;
00299   myeval.run();
00300
00301   if (predictedMolecule)
00302     delete predictedMolecule;
00303   if (XmolInputFile)
00304     {delete XmolInputFile;XmolInputFile =0;}
00305
```

5.45.1.2 void print_help (ostream & *os*)

Prints full help.

Definition at line 321 of file err_eval.C.

```
00322 {
00323   os    << "\nHELP\n\n";
00324   print_usage(os);
00325   os    << "Additional Options:\n"
00326         << "  \n"
00327         << "  \n";
00328
00329
```

5.45.1.3 void print_usage (ostream & *os*)

Prints information about usage.

Definition at line 310 of file err_eval.C.

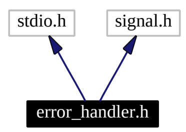
```
00311 {
00312   os    << "\nUsage: evalenv <-help> [optionfilename | options] \n\n"
00313         << "Required options:\n"
00314         << "  {-initial | -init | -i} <filename> }\n"
00315         << "     Initial molecule: xmol file\n"
00316         << "  {-constraints | -c}
00317
```

5.46 error_handler.h File Reference

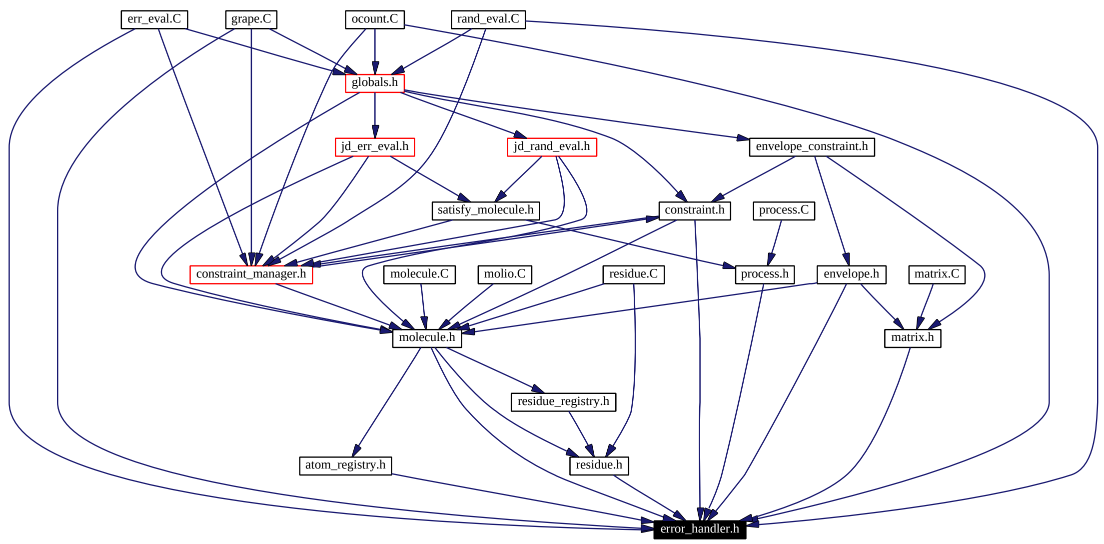
`#include <stdio.h>`

`#include <signal.h>`

Include dependency graph for error_handler.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **DIE**(TEXT)
- #define **WARN**(TEXT)
- #define **WARN_IF**(EXP, TEXT)

5.46.1 Define Documentation

5.46.1.1 #define DIE(TEXT)

Value:

```
do { \
fprintf (stderr, "\n\nERROR: " #TEXT "\n"); \
fprintf (stderr, "+%d %s\n\n", __LINE__, __FILE__); \
raise (SIGQUIT); \
} while (0)
```

Definition at line 7 of file error_handler.h.

Referenced by AnimFrame::AnimFrame(), AtomBuildup::AtomBuildup(), Envelope::Blur(), Envelope::CalcGradient(), EnvelopeConstraint::Compare(), Envelope::Compare(), BinEnvelopeConstraint::Compare(), EnvelopeConstraint::PreSetup(), Envelope::ReadAvsTxt(), Molecule::Reallocate(), RelaxSubset::RelaxSubset(), AnimFrame::Set_Output_Stream(), StopEval::StopEval(), ConstraintManager::TurnAtomOn(), Envelope::Upsize(), ConstraintManager::add(), Process::assertOrder(), Constraint::assertValidity(), PositionConstraint::computeStress(), EnvelopeConstraint::computeStress(), DistanceConstraint::computeStress(), BinEnvelopeConstraint::computeStress(), AtomConstraint::computeStress(), PositionConstraint::dStress(), DistanceConstraint::dStress(), AtomConstraint::dStress(), Matrix::det(), Matrix::findEigenVectors(), ConstraintRegistry::getConstraint(), ConstraintRegistry::getIntType(), ConstraintManager::getParents(), Matrix::identity(), jdErr_Eval::jdErr_Eval(), jdIterator::jdIterator(), main(), Constraint::notifyOff(), AtomConstraint::notifyRemove(), AtomConstraint::notifyValid(), Matrix::operator *(), Matrix::operator()(), Envelope::operator()(), Matrix::operator+(), Matrix::operator-(), ConstraintRegistry::readConstraints(), Molecule::readXmol(), ConstraintManager::registerEnvelope(), ConstraintManager::setMolecule(), ConstraintManager::setPosition(), Molecule::skipXmol(), and Constraint::stress().

5.46.1.2 #define WARN(TEXT)

Value:

```
do { \
fprintf (stderr, "\n\nWarning: " #TEXT "\n +%d %s\n\n", __LINE__, __FILE__ ); \
} while (0)
```

Definition at line 14 of file error_handler.h.

Referenced by EnvelopeConstraint::AlignModeltoCOM(), Envelope::Center(), Molecule::Reallocate(), AtomRegistry::SetType(), and main().

5.46.1.3 #define WARN_IF(EXP, TEXT)

Value:

```
do { if (EXP) \
        fprintf (stderr, "\n\nWarning: " #EXP #TEXT "\n +%d %s\n\n", __LINE__,  __FILE__ ); \
} while (0)
```

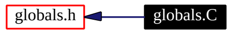
Definition at line 19 of file error_handler.h.

Referenced by Envelope::Build().

5.47 globals.C File Reference

`#include "globals.h"`

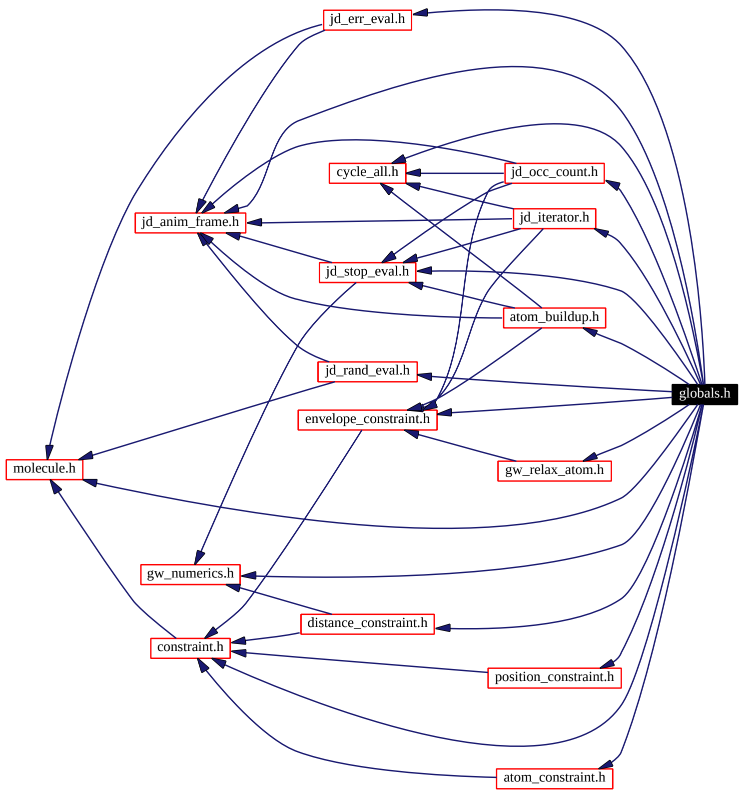
Include dependency graph for globals.C:



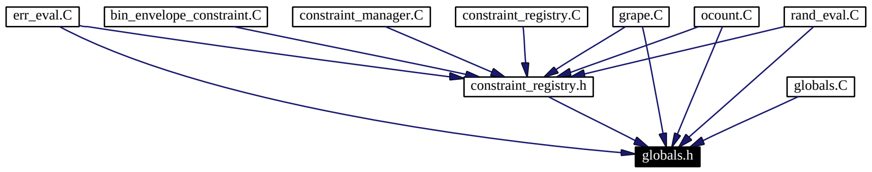
5.48 globals.h File Reference

```
#include "jd_iterator.h"
#include "gw_numerics.h"
#include "gw_relax_atom.h"
#include "cycle_all.h"
#include "jd_stop_eval.h"
#include "jd_anim_frame.h"
#include "jd_occ_count.h"
#include "jd_err_eval.h"
#include "jd_rand_eval.h"
#include "atom_buildup.h"
#include "molecule.h"
#include "constraint.h"
#include "atom_constraint.h"
#include "distance_constraint.h"
#include "envelope_constraint.h"
#include "position_constraint.h"
```

Include dependency graph for globals.h:



This graph shows which files directly or indirectly include this file:



5.49 grape.C File Reference

`#include <fstream.h>`

`#include "dynstring.h"`

`#include "dllist.h"`

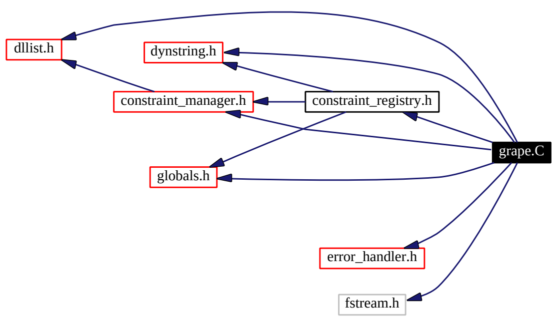
`#include "constraint_manager.h"`

`#include "constraint_registry.h"`

`#include "error_handler.h"`

`#include "globals.h"`

Include dependency graph for grape.C:



Functions

- void **print_usage** (ostream &os)

  Prints information about usage.

- void **print_help** (ostream &os)

  Prints full help.

- int **main** (int argc, char **argv)

5.49.1 Function Documentation

5.49.1.1 int main (int *argc*, char ** *argv*)

Definition at line 17 of file grape.C.

```
00017                                   {
00018   if(argc < 2)
00019     {
00020       print_usage(cout);
00021       return 0;
00022     }
00023 
00024   cerr << "Begin: ";
00025   for (int i =0; i < argc; i++){
00026     cerr << argv[i]<<" ";
00027   }
```

```
cerr << endl;

DLList<DynString> ArgList;

DLList<DynString> ConstraintFileList;

Molecule * predictedMolecule=0;

ofstream *AnimOut = 0;
// default fstream for xmol frames

// Iterate through all the command line arguments:
DynString arg;

// File for initial positions
DynString positionFile = "";

SatisfyMolecule * TOP;

// Command-line argument.
ifstream * input=0;
// Parameter input stream.
int argStart = 1;

// See if the user has specified a parameter file.  It is a
// parameter file if the first argument does *not* start with
// a dash, '-' (i.e. it is not a flag).
//

DynString BaseFN = "grape-output"; // final xmol filename
DynString AnimFN = ""; // anim filename -- defaults to off


if(argv[1][0] != '-')
  {
    // Read in the arguments from the parameter file.  This file should
    // contain flags (including their dashes) followed immediately by
    // the values associated with them.  Warning: this processing step
    // ignores white space!
    //
    if(!(input = new ifstream(argv[1])))
      {
        cerr << "Could not open/find parameter file, " << argv[1]
             << ", quitting. Try -help option.\n";
        return 1;
      }
    while(!input->eof() && !input->bad())
      {
        *input >> arg;

        if(arg[0] == '#')
          arg.EatLine(*input);
        // Ignore lines that begin with a pound, '#', sign:
        else
          ArgList.AddToBack(arg);
      }
    ArgList.RemoveFromBack();  // Get rid of the eof character.

    delete input; input=0;
    argStart=2;
  }

// Copy the arguments from the command line onto our argument list:
//
for(int a=argStart; a<argc; ++a)
```

```
    ArgList.AddToBack(DynString(argv[a]));

  while(ArgList)
    {
      arg = ArgList.RemoveFromFront();

      if((arg == DynString("-residues")) ||
              (arg == DynString("-res"))) {

        // Check to see if we've already read in a molecule:
          if(predictedMolecule)
            {
              cerr << "Warning: More than one file has been supplied for "
                   << "the initial molecule.\nWill use the molecule "
                   << "specified in the file listed last.\n";
              delete predictedMolecule; predictedMolecule=0;
            }
          arg = ArgList.RemoveFromFront();  // Get number of atoms
          if((input = new ifstream(arg.GetData())))
            {
              predictedMolecule = new Molecule;

              if(predictedMolecule->readFasta(*input)<2)
                {
                  cerr << "Error generating molecule from residue sequence.\n";
                  delete predictedMolecule; predictedMolecule=0;
                }
              delete input;input = 0;
            } else {
              cerr << "Could not open initial FASTA file, " << arg
                   << ", will try default location.\n";

            }
      }

      // Initial atom positions
      //
      else if((arg == DynString("-initial")) ||
              (arg == DynString("-init")) ||
              (arg == DynString("-i"))) {

        positionFile = ArgList.RemoveFromFront();  // Get the name of the xmol file.

      }

      else if(arg == DynString("-random")){

        positionFile = "random";

      }


      // Constraint file is being specified.
      //
      else if((arg == DynString("-constraint")) ||
              (arg == DynString("-constraints")) ||
              (arg == DynString("-c")))
        {
          // Just add this file to the list of files that contain
          // constraints for right now.  We'll read these in once
          // we're sure we have an initial molecule.
          //
          arg = ArgList.RemoveFromFront();
          ConstraintFileList.AddToBack(arg);
        }
```

```

//       // Read in van der Waal radius:
//       //
//       else if(arg == DynString("-vdw") ||
//            arg == DynString("-vdwradius"))
//      {
//        double val[4];
//        for(int i=0; i<4; ++i){
//          if (ArgList.IsEmpty()) {
//            DIE("Error while parsing the vdw arguments. Try -help option.");
//          }
//          arg = ArgList.RemoveFromFront();
//          val[i] = arg.ToDbl();
//        }
//        gwNumerics::VDWRadius = val;
//      }
//
//       // Read in local perturbation radius:
//       //
//       else if(arg == DynString("-lpr") ||
//            arg == DynString("-lpradius"))
//      {
//        double val[4];
//        for(int i=0; i<4; ++i){
//          if (ArgList.IsEmpty()) {
//            DIE("Error while  parsing the lpr arguments. Try -help option.");
//          }
//          arg = ArgList.RemoveFromFront();
//          val[i] = arg.ToDbl();
//        }
//        gwNumerics::LPertRadius = val;
//      }

      // Set vdw checking vs all or vs moved
      //
      else if(arg == DynString("-vom"))
        {
          if (ArgList.IsEmpty()) {
            DIE("Error while  parsing the vom argument. Try -help option.");
          }
          arg = ArgList.RemoveFromFront();
          if(arg == DynString("true") || arg == DynString("t") ||
             arg == DynString("True") || arg == DynString("TRUE") ||
             arg == DynString("T") || arg == DynString("1"))
            gwNumerics::UseOnlyMoved = true;
          else if(arg == DynString("false") || arg == DynString("f") ||
                  arg == DynString("False") || arg == DynString("FALSE") ||
                  arg == DynString("F") || arg == DynString("0"))
            gwNumerics::UseOnlyMoved = false;
          else cerr << "Unable do understand vom arg. Not set -- defaults to true\n";
        }



      else if(arg == DynString("-f") ||
              arg == DynString("-filebase"))
        {
            arg = ArgList.RemoveFromFront();
            if (arg[0] != '-'){
              BaseFN = arg;
            } else {
              DIE("Confused by filename ");
            }
        }

```

```

    // Read in flag for local perturbations:
    //
    else if(arg == DynString("-lp") ||
            arg == DynString("-lpert") )
      {
        arg = ArgList.RemoveFromFront();
        if(arg == DynString("true") || arg == DynString("t") ||
           arg == DynString("True") || arg == DynString("TRUE") ||
           arg == DynString("T") || arg == DynString("1"))
          gwNumerics::LocalPerturb = true;
        else if(arg == DynString("false") || arg == DynString("f") ||
                arg == DynString("False") || arg == DynString("FALSE") ||
                arg == DynString("F") || arg == DynString("0"))
          gwNumerics::LocalPerturb = false;
        else cerr << "Unable do understand lp or lpert. Not set.\n";
      }

    // Least-squares success tolerance:
    //
    else if(arg == DynString("-lssucc"))
      {
        arg = ArgList.RemoveFromFront();
        gwNumerics::ls_succ_tolerance = arg.ToDbl();
      }

    // Read in the least-squares derivative tolerance:
    //
    else if(arg == DynString("-lsderiv"))
      {
        arg = ArgList.RemoveFromFront();
        gwNumerics::ls_deriv_tolerance = arg.ToDbl();
      }

    // Read in the least-squares distance tolerance:
    //
    else if(arg == DynString("-lsdist"))
      {
        arg = ArgList.RemoveFromFront();
        gwNumerics::ls_dist_tolerance = arg.ToDbl();
      }

    // Set the maximum number of iterations allowed:
    //
    else if(arg == DynString("-iter"))
      {
        arg = ArgList.RemoveFromFront();
        gwRelaxAtom::max_iterations = arg.ToInt();
      }

    // Set the number of cycles:
    //
    else if(arg == DynString("-cycles"))
      {
        arg = ArgList.RemoveFromFront();
        int n = arg.ToInt();
        jdIterator::MaxIterations = n;
        StopEval::Enable = false;  // turn off auto stop (early) features
      }

    else if((arg == DynString("-buildcycles")) ||
             (arg == DynString("-bc")) )
      {
        arg = ArgList.RemoveFromFront();
        AtomBuildup::MaxIterations = arg.ToInt();
```

```
          arg = ArgList.RemoveFromFront();
          AtomBuildup::MaxFinalIterations = arg.ToInt();
        }


      else if(arg == DynString("-cyclelimit"))
        {
          arg = ArgList.RemoveFromFront();
          jdIterator::MaxIterations = arg.ToInt();
        }

      else if ((arg == DynString("-quiet")) ||
               (arg == DynString("-q")))
        {
          StopEval::Quiet = true;
        }

      else if (arg == DynString("-rtp"))
        {
          StopEval::Quiet = true;
          StopEval::VerboseOn = false;
          StopEval::Rtp  = true;
        }

      else if((arg == DynString("-anim")) ||
               (arg == DynString("-a")))
        {
          arg = ArgList.RemoveFromFront();
          AnimFrame::Enabled = true;
          AnimFN = arg;
        }
      else if(arg == DynString("-help"))
        {
          print_help(cout);
          return 0;

        }

      // Ignore the bogus parameter.
      //
      else
        {
          cerr << "Unknown argument, " << arg << ", given, ignoring.\n";
        }
    }// end processing of arglist


  // If we havent read in a set of residues, quit.
  if(!predictedMolecule)
    {
      cerr << "No initial molecule specified, quitting. Try -help option.\n";
      return 3;
    }

  if (positionFile != "") {
    // no atom buildup, specify positions

    if (positionFile != "random") {
      if((input = new ifstream(positionFile.GetData())))
        {
          if(predictedMolecule->readXmol(*input)<2)
            {
              cerr << "Error reading initial positions from " << positionFile << ".  Check the file.\n";
              delete predictedMolecule; predictedMolecule=0;
              return 4;
```

```
          }
        delete input; input=0;
      }
    else
      {
        cerr << "Could not open initial molecule file, " << positionFile << ".\n";
        return 5;
      }
  } else {
    // set postions randomly
    if(predictedMolecule->setRandom()<2)      {
      cerr << "Error generating random molecule.\n";
      delete predictedMolecule; predictedMolecule=0;
      return 6;
    }
  }

}

// Now that we've got an actual molecule, we can create a constraint
// manager and read in the constraints:
//
ConstraintManager CM(*predictedMolecule);
//  CM.RandomPerturb(.1);

int num_read = 0;

while(ConstraintFileList)
  {
    // Get the filename and then open the file and then read the
    // constraints in using the constraint manager.
    arg = ConstraintFileList.RemoveFromFront();
    input = new ifstream(arg.GetData());
    if(input->good())
      {
        // Use the registry's routine to read in constraints and
        // append them to the constraint manager.
        //

        num_read += ConstraintRegistry::readConstraints(*input,CM);
        delete input; input=0;
      } else {
        cerr << "\n\nWhile reading file: " << arg.GetData() << endl;
        DIE ("Cannot open file");
      }

  }

if (num_read == 0) {
  DIE ("You have no constraints");
}

//
// Begin Sanity Checks
//

//    DLList<Atom> LA;
//     Atom atom;
//     for(int i=0; i<predictedMolecule->Number(); ++i)
//       {
//         LA = CM.getAssocAtoms(i);
//    cout << "atom(" << i << "): ";
//    while(LA)
//      {
//        atom = LA.RemoveFromFront();
```

```
//         cout << atom.ID() << ' ';
//       }
//          cout << '\n';
//        }
//
//      DLList<Constraint*>::ConstIterator it;
//      Constraint * c;
//      for(it = CM.getConstraintList().GetConstIterator(); it; ++it)
//        {
//          c = *it;
//     cout << *c << '\n';
//        }
//
//
//
// End Sanity Checks
//

if (positionFile != "") {
  TOP = new jdIterator(CM);
} else {
  TOP = new AtomBuildup(CM);
}

```

5.49.1.2 void print_help (ostream & *os*)

Prints full help.

Definition at line 484 of file grape.C.

```
{
  os    << "\nHELP for using GRAPE\n\n";
  print_usage(os);
  os    << "Additional Options:\n"
        << "  {-vdw | -vdwradius} n1 n2 n3 n4\n"
        << "     Sets a global Van Der Waals Radius for Objective Function \n"
        << "     Values less than n1 to radius value n2 and to value n4 for \n"
        << "     OFV > n3+n1.\n"
        << "  {-lp | -lpert} {0|1|true|false}\n"
        << "     Enables or disables local pertubation.\n"
        << "  {-lpr | -lpradius} n1 n2 n3 n4\n"
        << "     Sets the global local perturbation radius as described above\n"
        << "  {-cycles n}\n"
        << "     Forces the simluation to run for n cycles.\n"
        << "  {-cyclelimit n}\n"
        << "     Limits the simluation to run for at most n cycles.\n"
        << "  {-a | -anim} cout | cerr | filename \n"
        << "     Write out xmol animation frames to stream or file.\n"
        << "  {-f | -filebase} filename \n"
        << "     Write final atom positions to filename\n"
        << "  {-q | -quiet} \n"
        << "     Run in quiet mode. \n"
        << "  {-rtp}

```

5.49.1.3 void print_usage (ostream & *os*)

Prints information about usage.

Definition at line 471 of file grape.C.

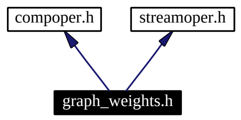
```
{
  os    << "\nUsage: grape <-help> [optionfilename | options] \n\n"
        << "Required options:\n"
        << "  {-initial | -init | -i} <filename>  | -random n }\n"
        << "     Initial molecule: xmol file or random molecule with n atoms\n"
        << "  or {-residue | -res} <filename>}\n"
        << "     Initial molecule: fasta file, implies random start <for now>\n"
        << "  {-constraints | -c}

```

5.50 graph_weights.h File Reference

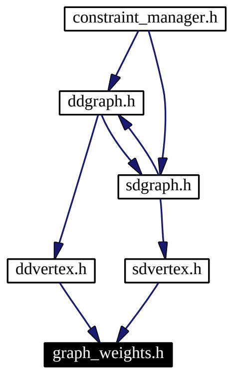
`#include "compoper.h"`

`#include "streamoper.h"`

Include dependency graph for graph_weights.h:



This graph shows which files directly or indirectly include this file:



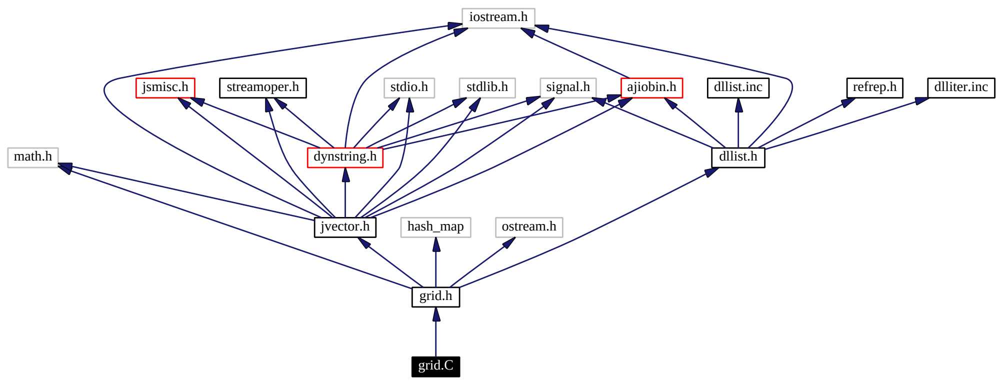
Compounds

- class **UnitWeight**

5.51 grid.C File Reference

`#include "grid.h"`

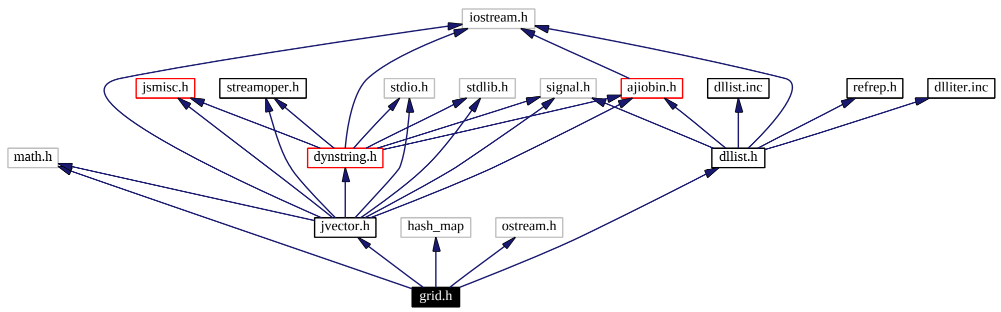
Include dependency graph for grid.C:



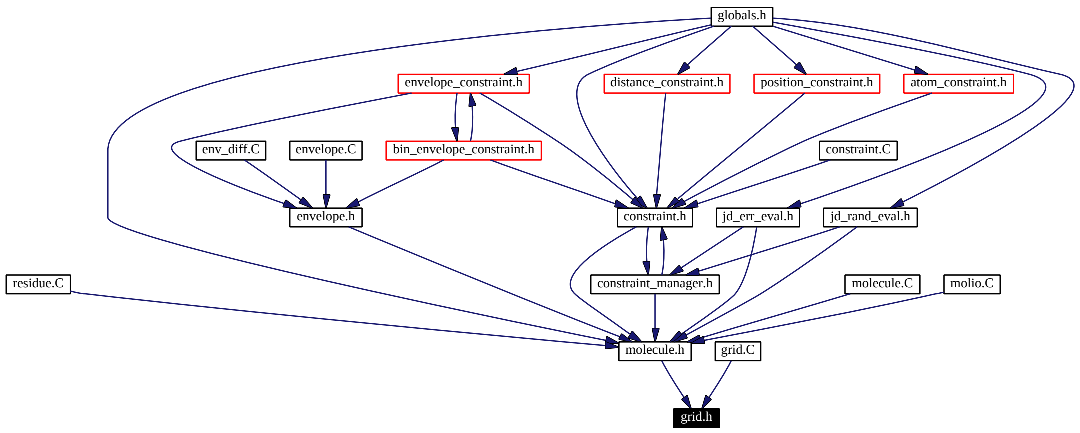
5.52 grid.h File Reference

```
#include <hash_map>
#include <ostream.h>
#include <math.h>
#include "jvector.h"
#include "dllist.h"
```

Include dependency graph for grid.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Grid**

Typedefs

- typedef long **GridNumber**
- typedef hash_multimap<const **GridNumber**, int, hash<long>, equal_to<long> > **hashmap_type**
- typedef **jVector**∗ **pointList**

5.52.1 Typedef Documentation

5.52.1.1 typedef long GridNumber

Definition at line 21 of file grid.h.

5.52.1.2 typedef hash_multimap< const GridNumber, int, hash< long >, equal_to< long > > hashmap_type

Definition at line 22 of file grid.h.

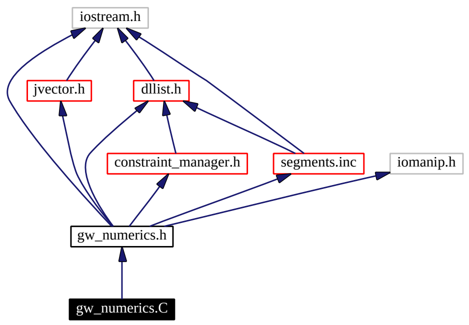
5.52.1.3 typedef jVector * pointList

Definition at line 23 of file grid.h.

5.53 gw_numerics.C File Reference

`#include "gw_numerics.h"`

Include dependency graph for gw_numerics.C:



5.54 gw_numerics.h File Reference

`#include <iostream.h>`

`#include <iomanip.h>`

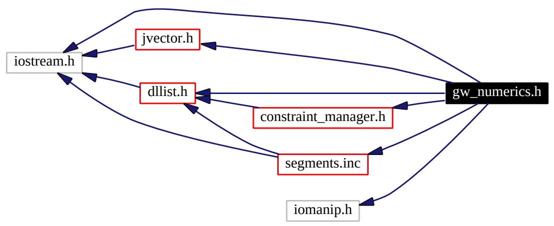
`#include "jvector.h"`

`#include "constraint_manager.h"`

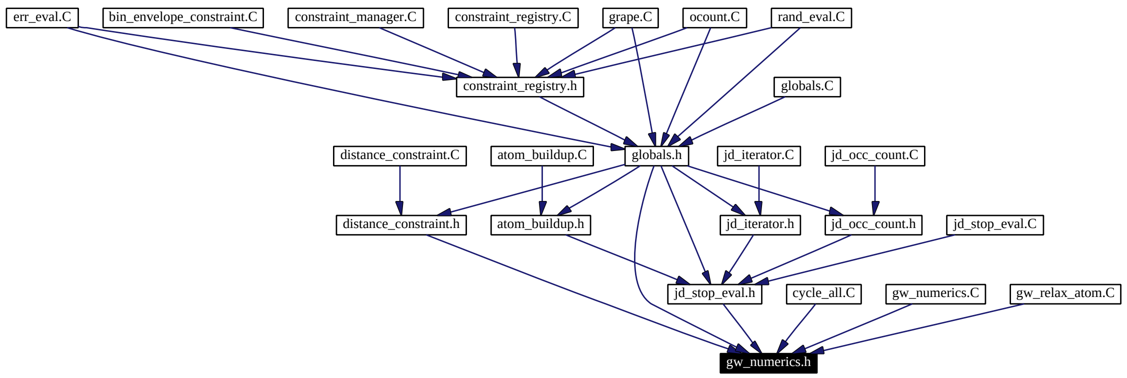
`#include "segments.inc"`

`#include "dllist.h"`

Include dependency graph for gw_numerics.h:



This graph shows which files directly or indirectly include this file:



Compounds

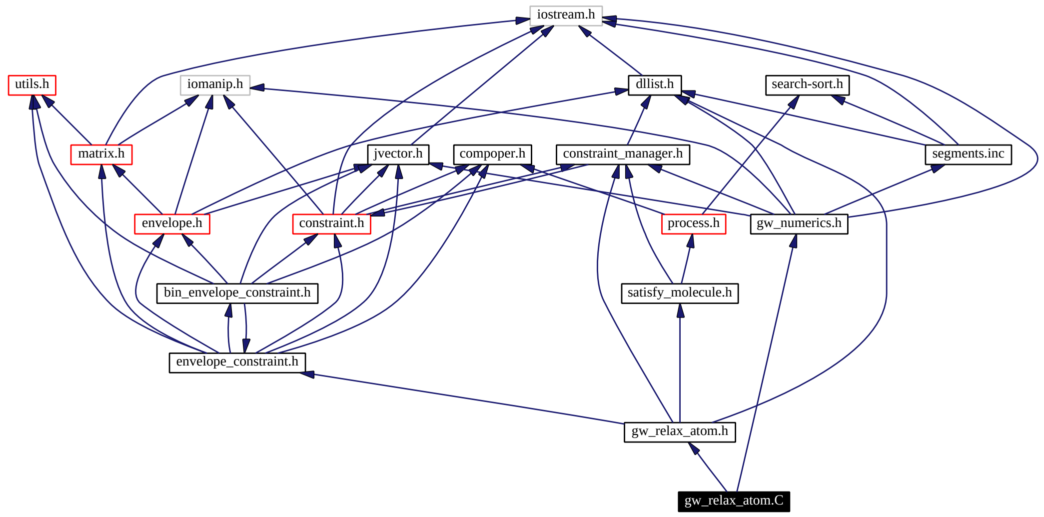
- class **gwNumerics**

5.55 gw_relax_atom.C File Reference

`#include "gw_relax_atom.h"`

`#include "gw_numerics.h"`

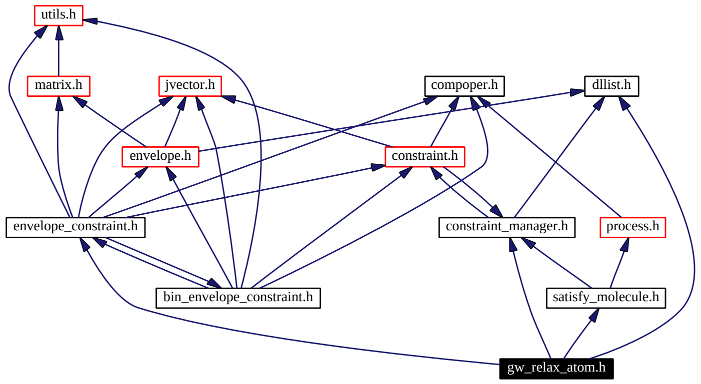
Include dependency graph for gw_relax_atom.C:



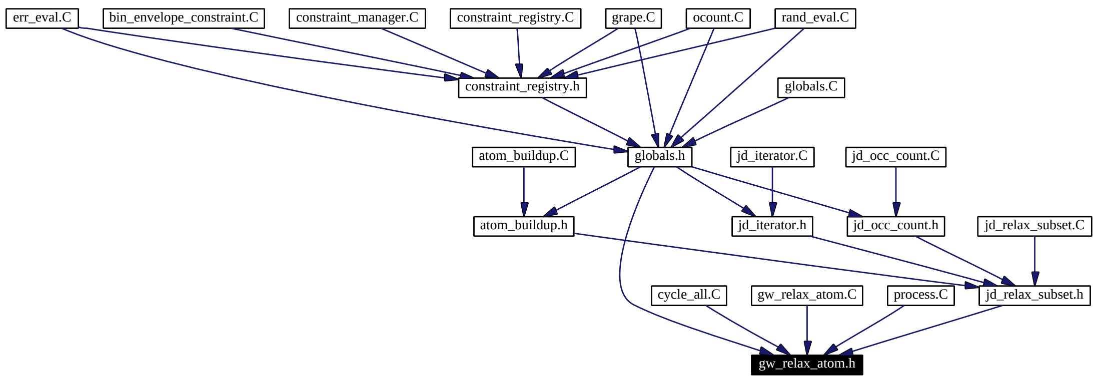
5.56 gw_relax_atom.h File Reference

```
#include "satisfy_molecule.h"
#include "constraint_manager.h"
#include "envelope_constraint.h"
#include "dllist.h"
```

Include dependency graph for gw_relax_atom.h:



This graph shows which files directly or indirectly include this file:



Compounds

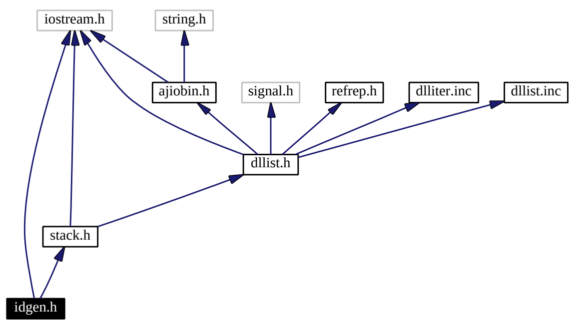
- class **gwRelaxAtom**

5.57 idgen.h File Reference

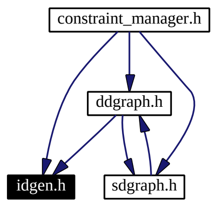
`#include <iostream.h>`

`#include "stack.h"`

Include dependency graph for idgen.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **IdGen**

Functions

- ostream& **operator**<< (ostream &os, const **IdGen** &idg)

5.57.1 Function Documentation

5.57.1.1 ostream & operator<< (ostream & *os*, const IdGen & *idg*) `[inline]`

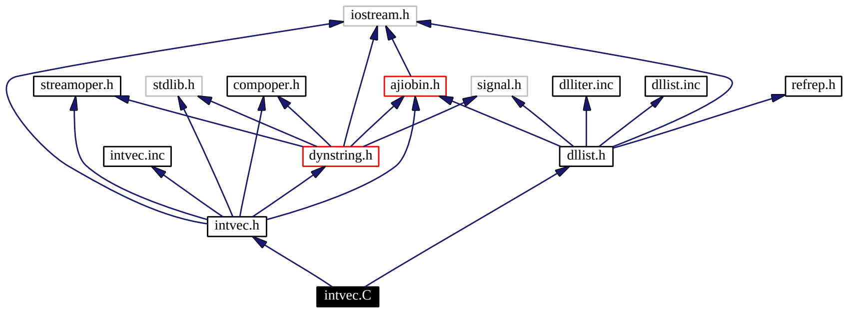
Definition at line 126 of file idgen.h.

```
00127 {
00128   os << '[' << idg.MaxId << "] {" << idg.FreeId << '}
00129
```

5.58 intvec.C File Reference

```
#include "intvec.h"
#include "dllist.h"
```

Include dependency graph for intvec.C:



5.59 intvec.h File Reference

#include <iostream.h>

#include <stdlib.h>

#include "dynstring.h"

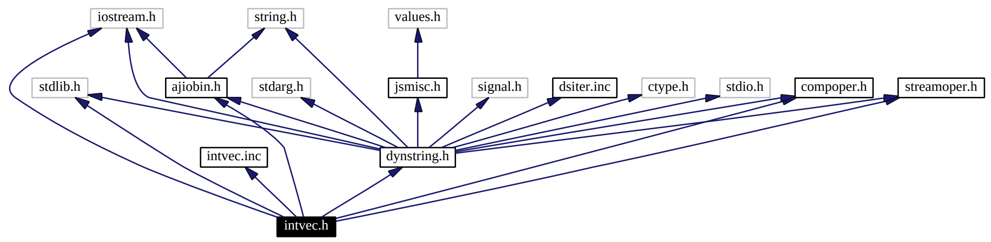
#include "ajiobin.h"

#include "compoper.h"

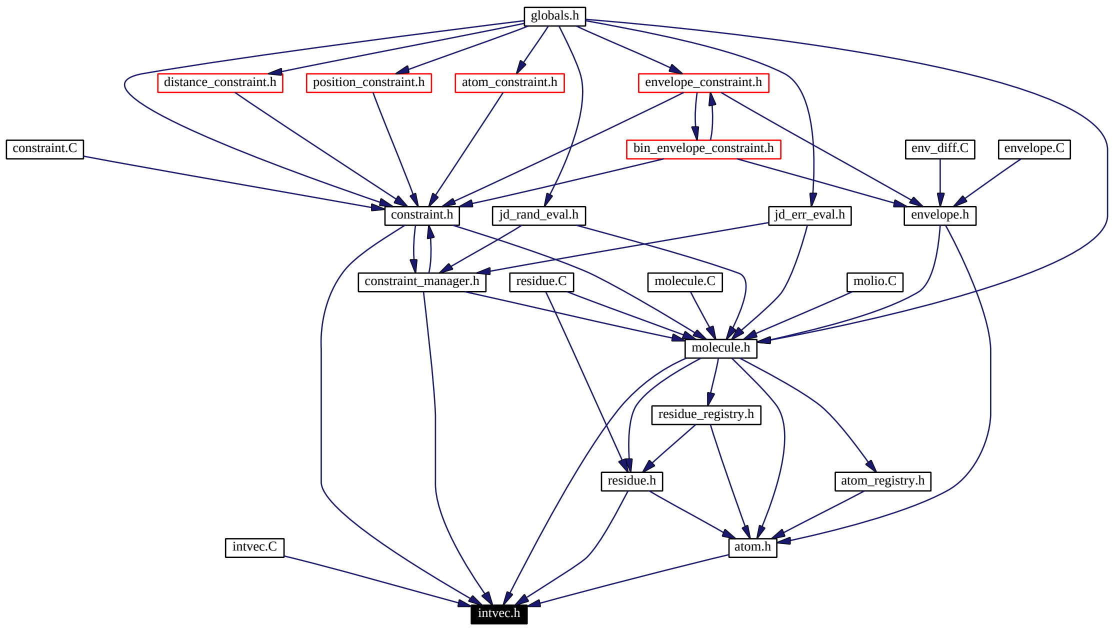
#include "streamoper.h"

#include <intvec.inc>

Include dependency graph for intvec.h:



This graph shows which files directly or indirectly include this file:

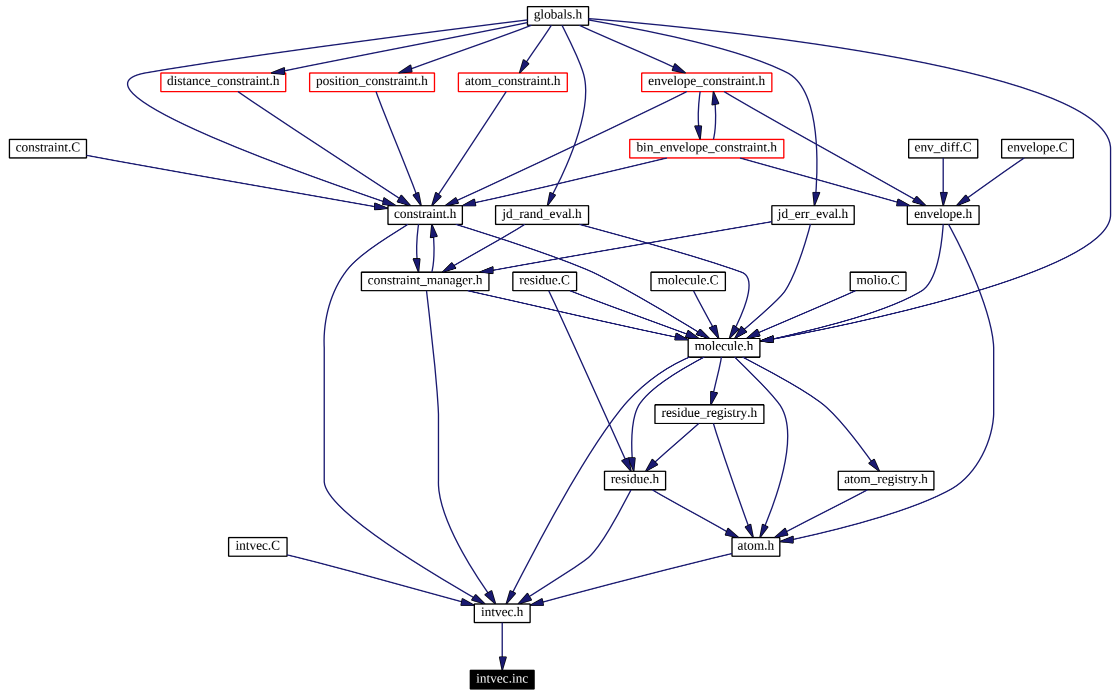


Compounds

- class **IntVec**

5.60 intvec.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- int **operator==** (const **IntVec** &a, const **IntVec** &b)
- int **operator!=** (const **IntVec** &a, const **IntVec** &b)
- int **operator<=** (const **IntVec** &a, const **IntVec** &b)
- int **operator>=** (const **IntVec** &a, const **IntVec** &b)
- int **operator<** (const **IntVec** &a, const **IntVec** &b)
- int **operator>** (const **IntVec** &a, const **IntVec** &b)
- **IntVec cat** (const **IntVec** &a, const **IntVec** &b)

5.60.1 Function Documentation

5.60.1.1 IntVec cat (const IntVec & *a*, const IntVec & *b*) `[inline]`

Definition at line 233 of file intvec.inc.

Referenced by ConstraintManager::add().

```
{
  IntVec c(a.L+b.L);
  for(int i=0; i<a.L; ++i) { c.I[i] = a.I[i]; }
  for(int i=0; i<b.L; ++i) { c.I[a.L+i] = b.I[i]; }

```

5.60.1.2 int operator!= (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 127 of file intvec.inc.

```
{ return !(a == b); }

```

5.60.1.3 int operator< (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 154 of file intvec.inc.

```
{
  if(a.L > b.L) return 0;
  if(a.L < b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] > b.I[i]) return 0;
      if(a.I[i] < b.I[i]) return 1;
    }

```

5.60.1.4 int operator<= (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 130 of file intvec.inc.

```
{
  if(a.L > b.L) return 0;
  if(a.L < b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] > b.I[i]) return 0;
      if(a.I[i] < b.I[i]) return 1;
    }

```

5.60.1.5 int operator== (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 119 of file intvec.inc.

```
{
  if(a.L != b.L) return 0;
  for(int i=0; i<a.L; ++i)
      if(a.I[i] != b.I[i]) return 0;
  return 1;

```

5.60.1.6 int operator> (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 166 of file intvec.inc.

```
{
  if(a.L < b.L) return 0;
  if(a.L > b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
```

```
      if(a.I[i] < b.I[i]) return 0;
      if(a.I[i] > b.I[i]) return 1;
    }

```

5.60.1.7 int operator>= (const IntVec & *a*, const IntVec & *b*) [inline]

Definition at line 142 of file intvec.inc.

```
{
  if(a.L < b.L) return 0;
  if(a.L > b.L) return 1;
  for(int i=0; i<a.L; ++i)
    {
      if(a.I[i] < b.I[i]) return 0;
      if(a.I[i] > b.I[i]) return 1;
    }

```

5.61 ipqueue.h File Reference

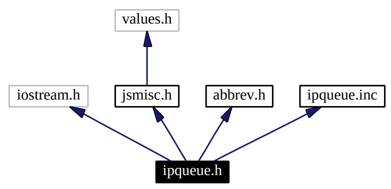
`#include <iostream.h>`

`#include "jsmisc.h"`

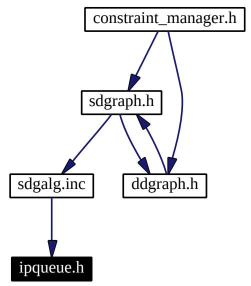
`#include "abbrev.h"`

`#include <ipqueue.inc>`

Include dependency graph for ipqueue.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **IntPriorityQueue**

Variables

- const int **IPQDEFLEN** = 1000

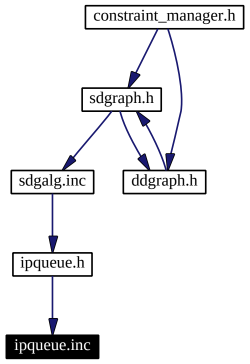
5.61.1 Variable Documentation

5.61.1.1 const int IPQDEFLEN = 1000

Definition at line 9 of file ipqueue.h.

5.62 ipqueue.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- template<class PT> ostream& **operator**<< (ostream &os, const **IntPriorityQueue**< PT > &Q)

5.62.1 Function Documentation

5.62.1.1 template<class PT> ostream & operator<< (ostream & *os*, const IntPriorityQueue< PT > & *Q*) `[inline]`

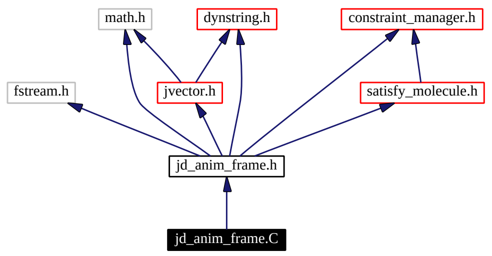
Definition at line 453 of file ipqueue.inc.

```
{
  if(Q.IsBad())
    {
      os << "<<BADIPQUEUE>>";
      return os;
    }

```

5.63 jd_anim_frame.C File Reference

`#include "jd_anim_frame.h"`

Include dependency graph for jd_anim_frame.C:



5.64 jd_anim_frame.h File Reference

`#include <fstream.h>`

`#include "jvector.h"`

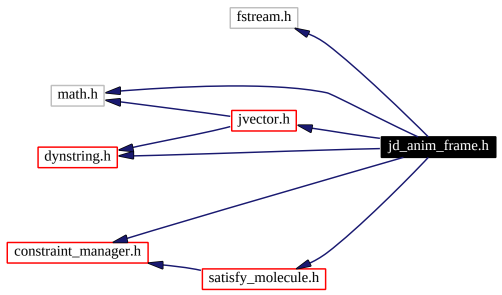
`#include "math.h"`

`#include "constraint_manager.h"`

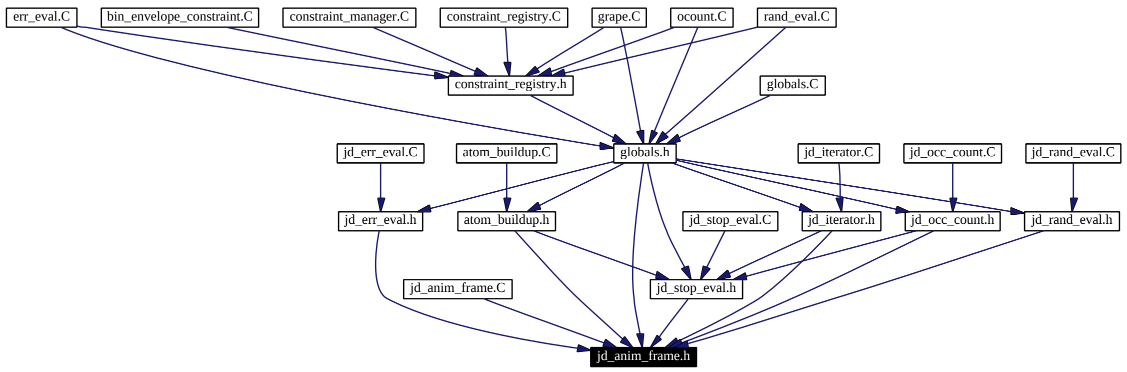
`#include "satisfy_molecule.h"`

`#include "dynstring.h"`

Include dependency graph for jd_anim_frame.h:



This graph shows which files directly or indirectly include this file:



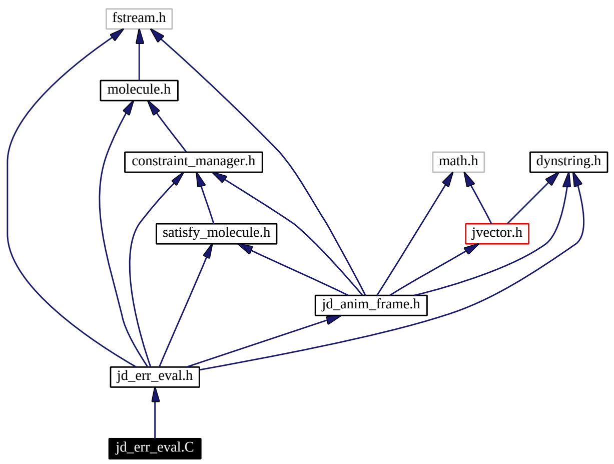
Compounds

- class **AnimFrame**

5.65 jd_err_eval.C File Reference

```
#include "jd_err_eval.h"
```

Include dependency graph for jd_err_eval.C:



5.66 jd_err_eval.h File Reference

`#include <fstream.h>`

`#include "jd_anim_frame.h"`

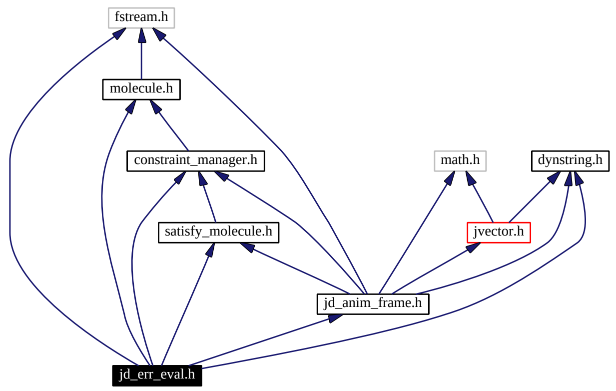
`#include "constraint_manager.h"`

`#include "satisfy_molecule.h"`

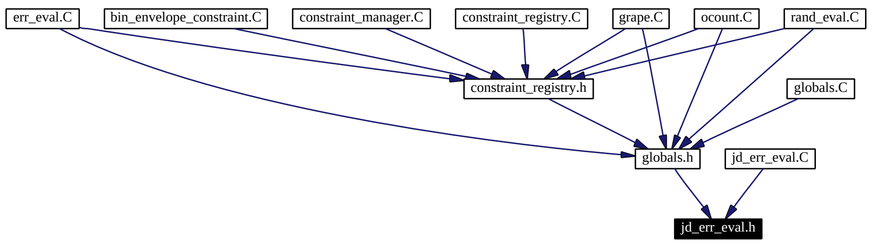
`#include "dynstring.h"`

`#include "molecule.h"`

Include dependency graph for jd_err_eval.h:



This graph shows which files directly or indirectly include this file:



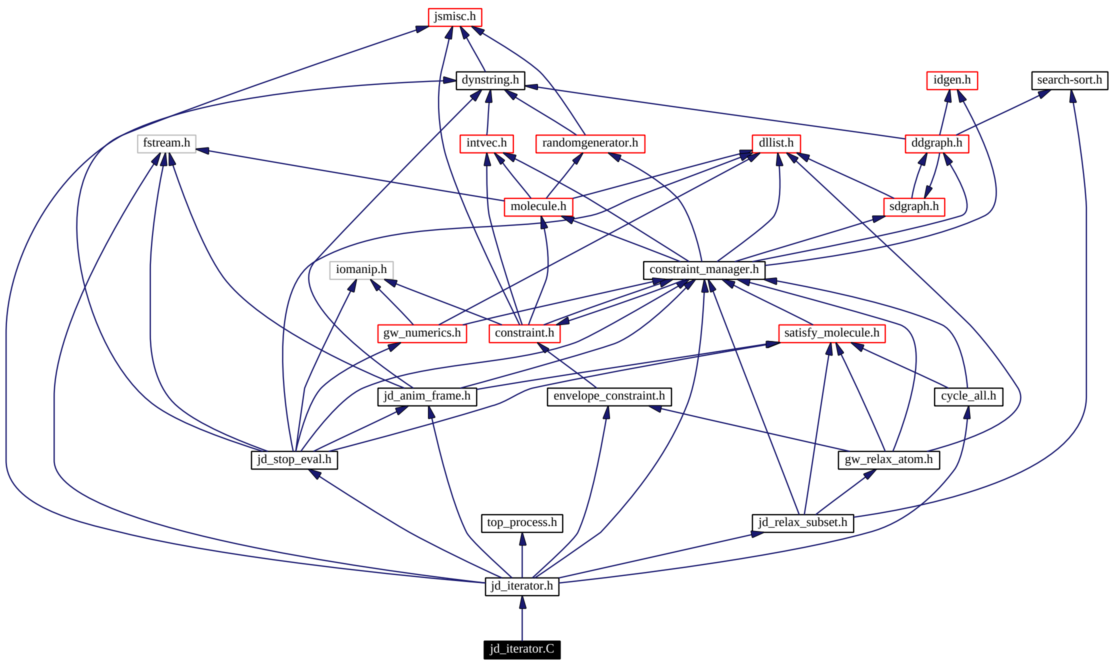
Compounds

- class **jdErr_Eval**

5.67 jd_iterator.C File Reference

`#include "jd_iterator.h"`

Include dependency graph for jd_iterator.C:



5.68 jd_iterator.h File Reference

#include <fstream.h>

#include "constraint_manager.h"

#include "top_process.h"

#include "dynstring.h"

#include "cycle_all.h"

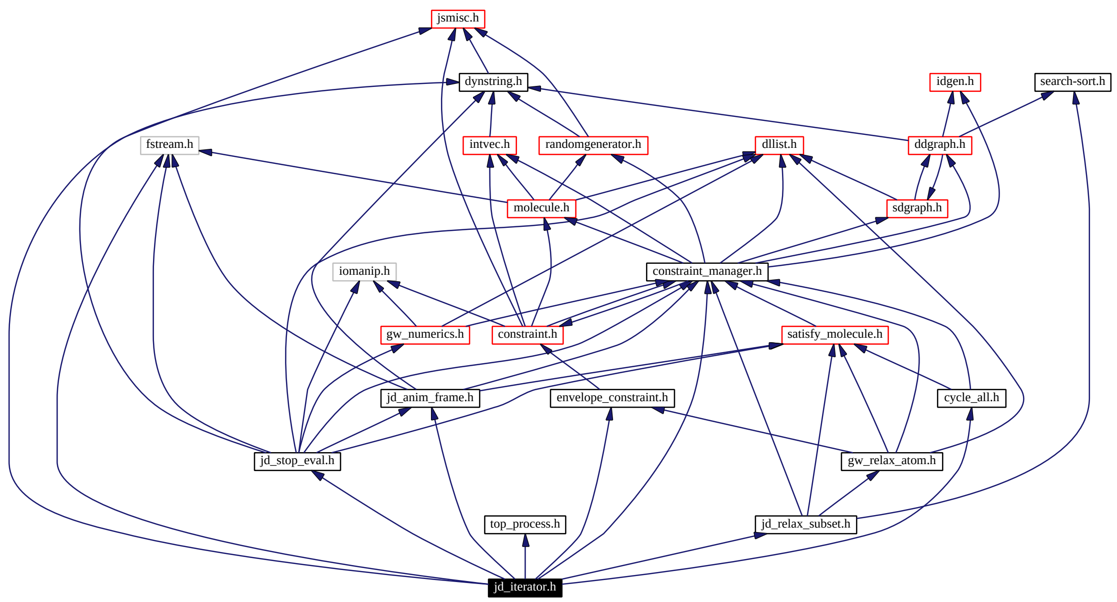
#include "jd_stop_eval.h"

#include "jd_anim_frame.h"

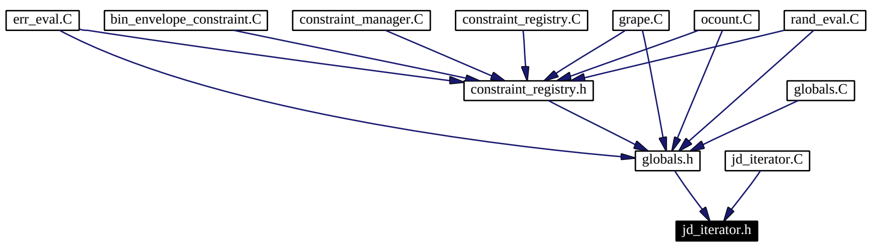
#include "jd_relax_subset.h"

#include "envelope_constraint.h"

Include dependency graph for jd_iterator.h:



This graph shows which files directly or indirectly include this file:



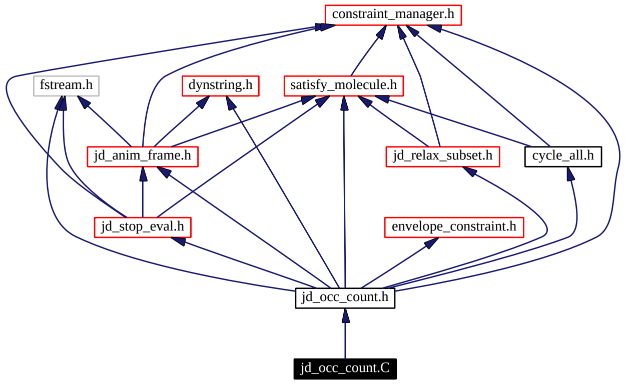
Compounds

- class **jdIterator**

5.69 jd_occ_count.C File Reference

`#include "jd_occ_count.h"`

Include dependency graph for jd_occ_count.C:



5.70 jd_occ_count.h File Reference

#include <fstream.h>

#include "constraint_manager.h"

#include "satisfy_molecule.h"

#include "dynstring.h"

#include "cycle_all.h"

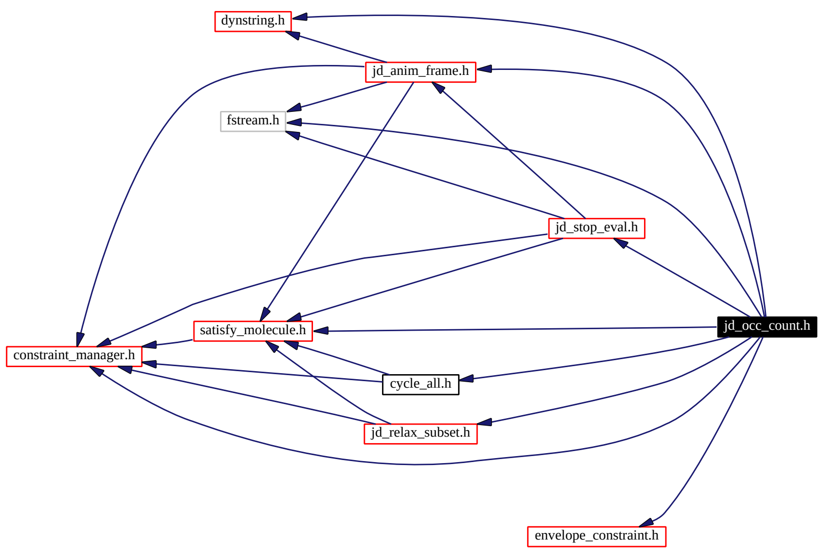
#include "jd_stop_eval.h"

#include "jd_anim_frame.h"

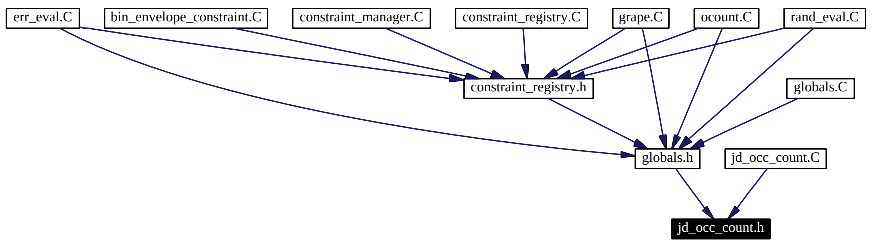
#include "jd_relax_subset.h"

#include "envelope_constraint.h"

Include dependency graph for jd_occ_count.h:



This graph shows which files directly or indirectly include this file:



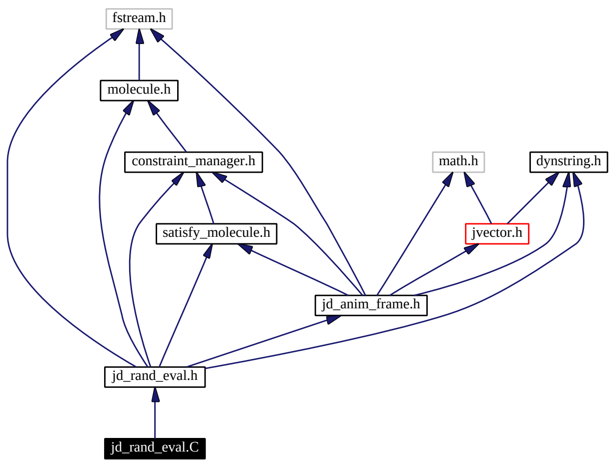
Compounds

- class **jdOcc_Count**

5.71 jd_rand_eval.C File Reference

`#include "jd_rand_eval.h"`

Include dependency graph for jd_rand_eval.C:



5.72 jd_rand_eval.h File Reference

`#include <fstream.h>`

`#include "jd_anim_frame.h"`

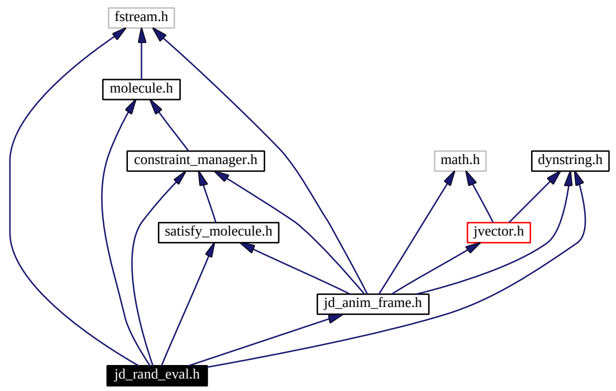
`#include "constraint_manager.h"`

`#include "satisfy_molecule.h"`

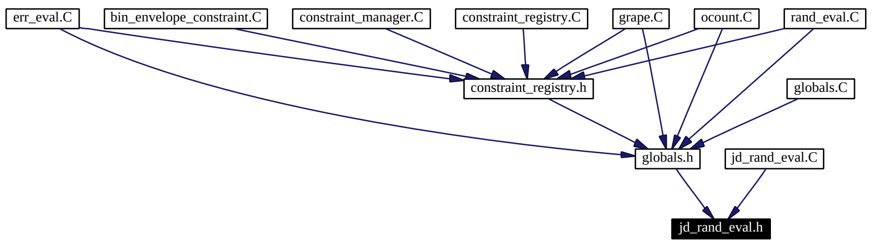
`#include "dynstring.h"`

`#include "molecule.h"`

Include dependency graph for jd_rand_eval.h:



This graph shows which files directly or indirectly include this file:



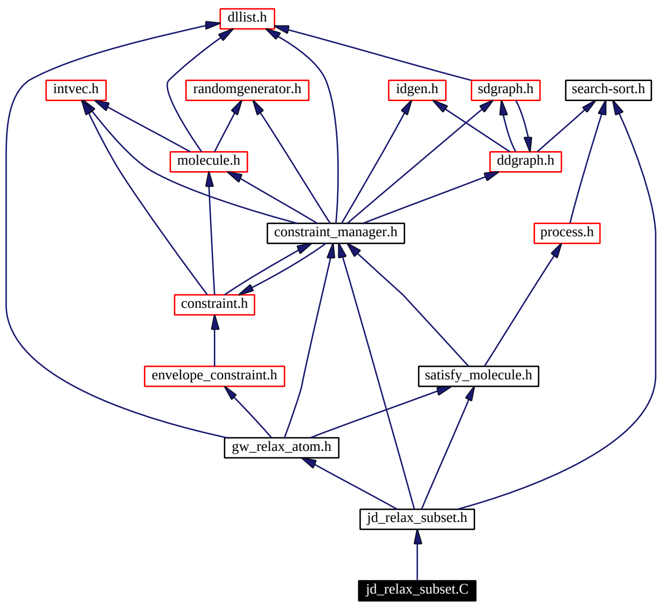
Compounds

- class **jdRand_Eval**

5.73 jd_relax_subset.C File Reference

`#include "jd_relax_subset.h"`

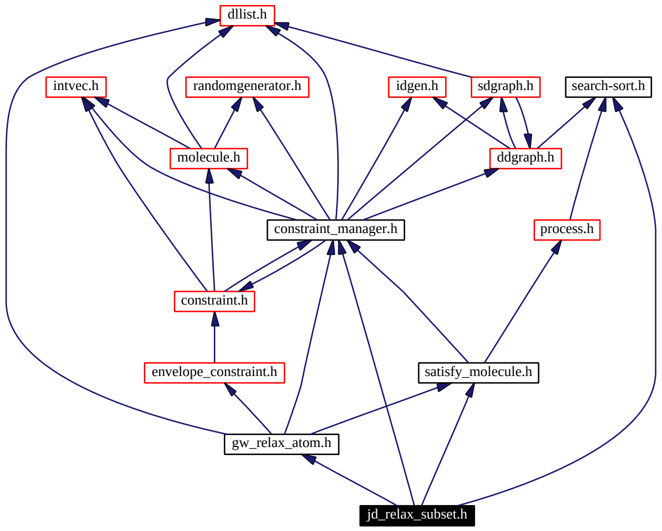
Include dependency graph for jd_relax_subset.C:



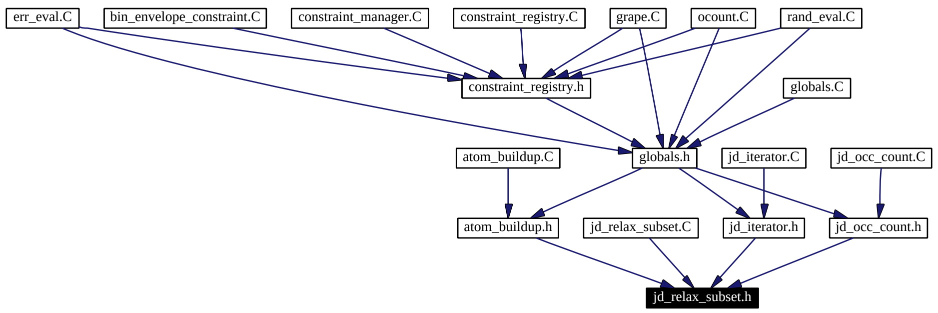
5.74 jd_relax_subset.h File Reference

```
#include "constraint_manager.h"
#include "search-sort.h"
#include "satisfy_molecule.h"
#include "gw_relax_atom.h"
```

Include dependency graph for jd_relax_subset.h:



This graph shows which files directly or indirectly include this file:



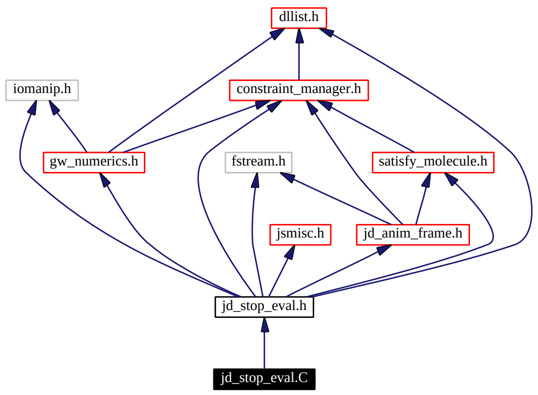
Compounds

- class **RelaxSubset**

5.75 jd_stop_eval.C File Reference

`#include "jd_stop_eval.h"`

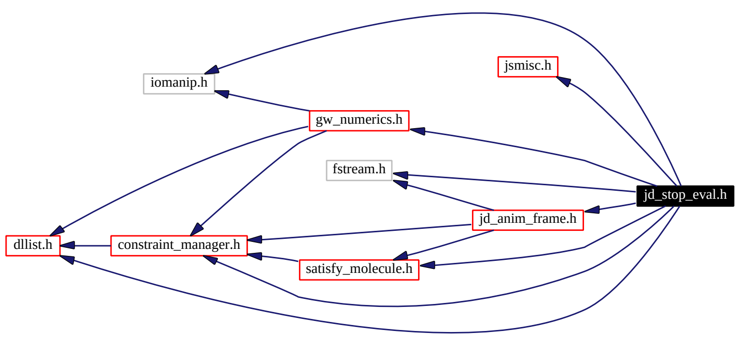
Include dependency graph for jd_stop_eval.C:



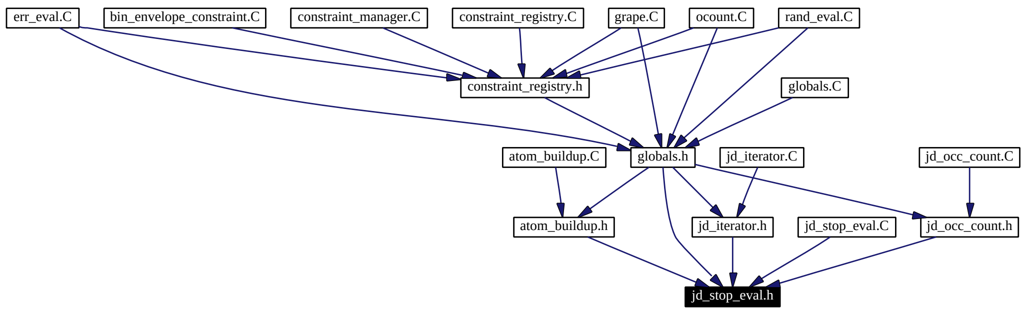
5.76 jd_stop_eval.h File Reference

```
#include <fstream.h>
#include <iomanip.h>
#include "constraint_manager.h"
#include "dllist.h"
#include "gw_numerics.h"
#include "satisfy_molecule.h"
#include "jsmisc.h"
#include "jd_anim_frame.h"
```

Include dependency graph for jd_stop_eval.h:



This graph shows which files directly or indirectly include this file:



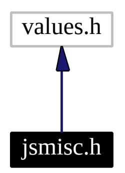
Compounds

- class **StopEval**

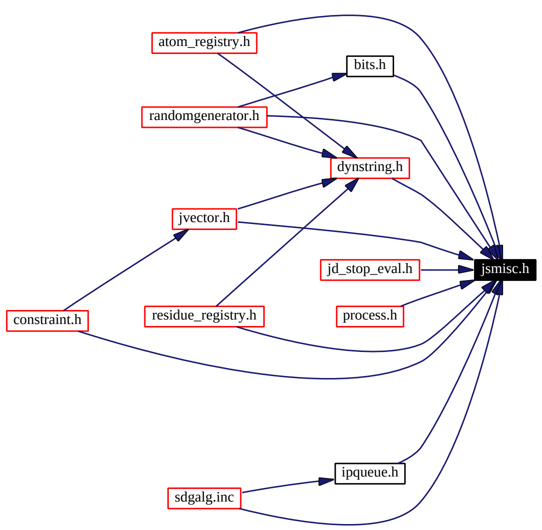
5.77 jsmisc.h File Reference

`#include <values.h>`

Include dependency graph for jsmisc.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **PI** 3.141592653589793116
- #define **INF** MAXDOUBLE∗MAXDOUBLE

Typedefs

- typedef unsigned int **uint**
- typedef short int **sint**
- typedef long int **lint**
- typedef unsigned short int **usint**
- typedef unsigned long int **ulint**
- typedef unsigned char **uchar**
- typedef class **ABitSet ABS**

5.77.1 Define Documentation

5.77.1.1 #define INF MAXDOUBLE*MAXDOUBLE

Definition at line 28 of file jsmisc.h.

5.77.1.2 #define PI 3.141592653589793116

Definition at line 24 of file jsmisc.h.

5.77.2 Typedef Documentation

5.77.2.1 typedef class ABitSet ABS

Definition at line 21 of file jsmisc.h.

Referenced by bf_iterator::bf_iterator(), df_iterator::df_iterator(), bf_iterator::operator=(), and df_iterator::operator=().

5.77.2.2 typedef long int lint

Definition at line 17 of file jsmisc.h.

5.77.2.3 typedef short int sint

Definition at line 16 of file jsmisc.h.

5.77.2.4 typedef unsigned char uchar

Definition at line 20 of file jsmisc.h.

5.77.2.5 typedef unsigned int uint

Definition at line 15 of file jsmisc.h.

5.77.2.6 typedef unsigned long int ulint

Definition at line 19 of file jsmisc.h.

5.77.2.7 typedef unsigned short int usint

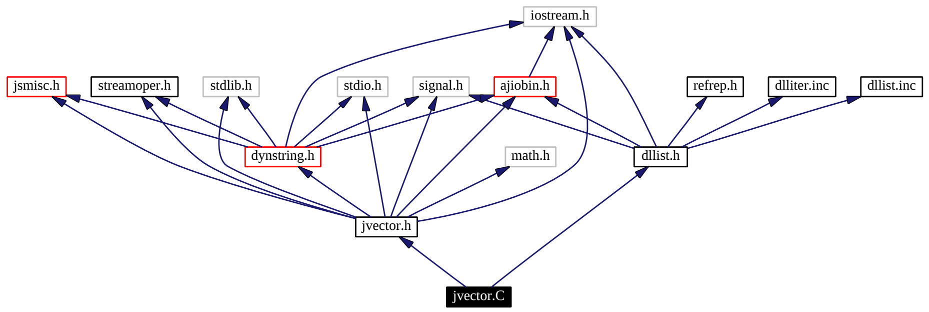
Definition at line 18 of file jsmisc.h.

5.78 jvector.C File Reference

`#include "jvector.h"`

`#include "dllist.h"`

Include dependency graph for jvector.C:



Functions

- **jVector operator-** (**jVector** x, const **jVector** &y)
- **jVector operator+** (**jVector** x, const **jVector** &y)
- double **operator** ∗ (**jVector** x, const **jVector** &y)
- **jVector operator%** (**jVector** x, const **jVector** &y)
- **jVector operator** ∗ (**jVector** y, double d)
- **jVector operator** ∗ (double d, **jVector** y)
- **jVector operator/** (**jVector** y, double d)
- **jVector operator%** (**jVector** x, double d)
- double **JVAngle** (const **jVector** &x, const **jVector** &y)
- double **euclid** (const **jVector** &x, const **jVector** &y)
- ostream& **VBPrintLine** (ostream &os, const **jVector** &x, const **jVector** &y, const **DynString** &color)
- ostream& **VBPrintPoint** (ostream &os, const **jVector** &x, const **DynString** &color)
- ostream& **VBPrintCircle** (ostream &os, const **jVector** &x, const double &radius, const **DynString** &color)
- double **Covariance** (const **jVector** &x, const **jVector** &y)

5.78.1 Function Documentation

5.78.1.1 double Covariance (const jVector & *x*, const jVector & *y*)

Definition at line 326 of file jvector.C.

```
00327 {
00328   if(x.L != y.L || x.L == 0)
00329       return INF;
00330
00331   double x_mean = x.mean();
00332   double y_mean = y.mean();
00333   double cov = 0.0;
00334
00335   for(int i=0; i<x.L; ++i)
```

```
00336       cov += (x[i] - x_mean)*(y[i] - y_mean);
00337   cov /= (x.L-1);
00338
00339   return cov;
00340
```

5.78.1.2 double JVAngle (const jVector & *x*, const jVector & *y*)

Definition at line 152 of file jvector.C.

```
00153 { return acos( (x*y) / x.mag() / y.mag() ); }
00154
```

5.78.1.3 ostream & VBPrintCircle (ostream & *os*, const jVector & *x*, const double & *radius*, const DynString & *color*)

Definition at line 264 of file jvector.C.

```
00266 {
00267
00268   if((x.L>3) || (x.L<2)) return os;
00269   char old_delim = jVector::IODelimiter;
00270   jVector::IODelimiter = ',';
00271
00272   if(x.L==3)
00273     os << "circle (" << x << ") " << radius << " :" << color;
00274   else if(x.L==2)
00275     os << "circle (" << x << ",0) " << radius << " :" << color;
00276   os << '\n';
00277   jVector::IODelimiter = old_delim;
00278   return os;
00279
```

5.78.1.4 ostream & VBPrintLine (ostream & *os*, const jVector & *x*, const jVector & *y*, const DynString & *color*)

Definition at line 226 of file jvector.C.

```
00228 {
00229   if((x.L != y.L) || (x.L>3) || (y.L>3) || (x.L<1) || (y.L<1)) return os;
00230
00231   char old_delim = jVector::IODelimiter;
00232   jVector::IODelimiter = ',';
00233
00234   if(x.L==3)
00235     os << "line (" << x << ") (" << y << ") :" << color;
00236   else if(x.L==2)
00237     os << "line (" << x << ",0) (" << y << ",0) :" << color;
00238   else if(x.L==1)
00239     os << "line (" << x << "0,0) (" << y << "0,0) :" << color;
00240   os << '\n';
00241   jVector::IODelimiter = old_delim;
00242   return os;
00243
```

5.78.1.5 ostream & VBPrintPoint (ostream & *os*, const jVector & *x*, const DynString & *color*)

Definition at line 245 of file jvector.C.

```
00246 {
00247
00248   if((x.L>3) || (x.L<1)) return os;
00249   char old_delim = jVector::IODelimiter;
00250   jVector::IODelimiter = ',';
00251
00252   if(x.L==3)
00253     os << "point " << x << " :" << color;
00254   else if(x.L==2)
00255     os << "point " << x << ",0 :" << color;
00256   else if(x.L==1)
00257     os << "point " << x << ",0,0 :" << color;
00258   os << '\n';
00259   jVector::IODelimiter = old_delim;
00260   return os;
00261
```

5.78.1.6 double euclid (const jVector & *x*, const jVector & *y*)

Definition at line 155 of file jvector.C.

Referenced by gwNumerics::addVDWSegments(), DistanceConstraint::computeAbsRes(), DistanceConstraint::computeStress(), PositionConstraint::dStress(), DistanceConstraint::dStress(), main(), and jdErr_Eval::run().

```
00156 {
00157   if(x.L != y.L) return -1.0;
00158   double sumproducts = 0.0;
00159   for(int i=0; i<x.L; ++i) sumproducts += pow(x.V[i]-y.V[i], 2.0);
00160   return sqrt(sumproducts);
00161
```

5.78.1.7 jVector operator ∗ (double *d*, jVector *y*)

Definition at line 141 of file jvector.C.

```
00141                                         { return y*=d; }
00142
```

5.78.1.8 jVector operator ∗ (jVector *y*, double *d*)

Definition at line 140 of file jvector.C.

```
00140                                         { return y*=d; }
00141
```

5.78.1.9 double operator ∗ (jVector *x*, const jVector & *y*)

Definition at line 124 of file jvector.C.

```
00125 {
00126   double product_sum=0.0;
00127   if(x.L == y.L) for(int i=0; i<x.L; ++i) product_sum += x.V[i]*y.V[i];
00128   return product_sum;
00129
```

5.78.1.10 jVector operator% (jVector *x*, double *d*)

Definition at line 149 of file jvector.C.

```
00150   { return x%=d; }
00151
```

5.78.1.11 jVector operator% (jVector *x*, const jVector & *y*)

Definition at line 134 of file jvector.C.

```
00134                                                    { return x %= y; }
00135
```

5.78.1.12 jVector operator+ (jVector *x*, const jVector & *y*)

Definition at line 121 of file jvector.C.

```
00121                                                   { return (x += y); }
00122
```

5.78.1.13 jVector operator- (jVector *x*, const jVector & *y*)

Definition at line 117 of file jvector.C.

```
00117                                                   { return (x -= y); }
00118
```

5.78.1.14 jVector operator/ (jVector *y*, double *d*)

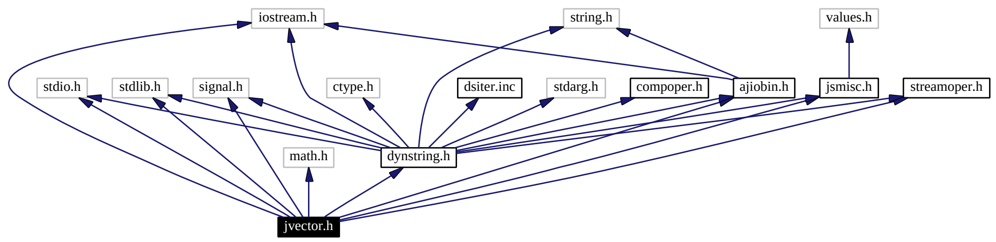
Definition at line 145 of file jvector.C.

```
00145                                           { return y/=d; }
00146
```

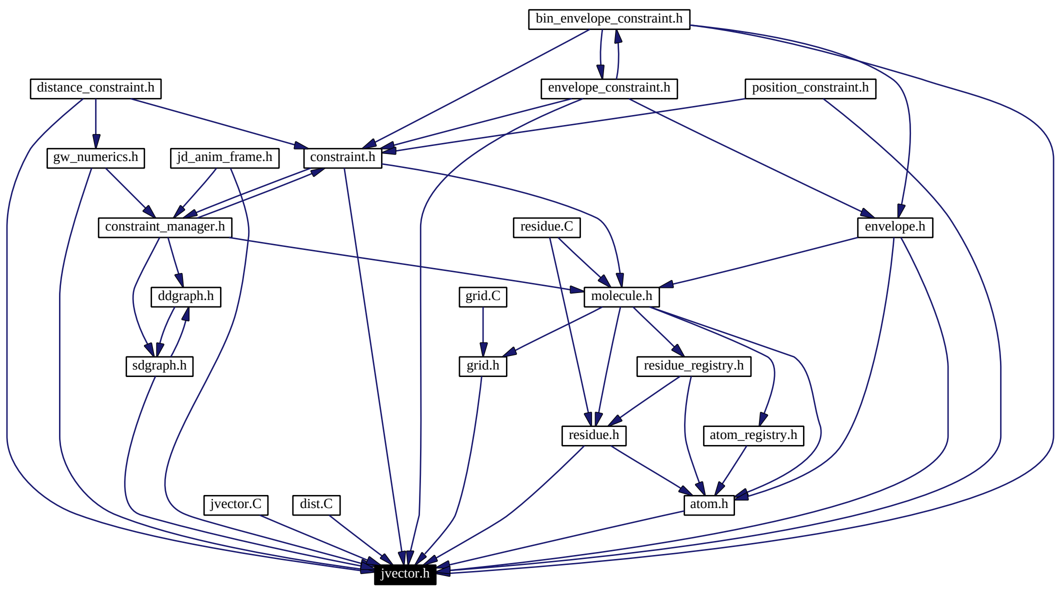
5.79 jvector.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <iostream.h>
#include <math.h>
#include "jsmisc.h"
#include "dynstring.h"
#include "ajiobin.h"
#include "streamoper.h"
```

Include dependency graph for jvector.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **jVector**

Functions

- int **doublecompare** (const void ∗x, const void ∗y)
- int **binsearch** (const **jVector** &v, double d)
- int **dcomp** (const void ∗x, const void ∗y)
- void **JVqsort** (**jVector** &v)

Variables

- const int **DEFAULTLEN** = 3

5.79.1 Function Documentation

5.79.1.1 void JVqsort (jVector & *v*) [inline]

Definition at line 329 of file jvector.h.

```
00329                          { qsort((void *) v.V, v.L, sizeof(double), dcomp); }
00330
```

5.79.1.2 int binsearch (const jVector & *v*, double *d*) [inline]

Definition at line 299 of file jvector.h.

```
00300 {
00301   // left index, right index, pivot, respectively.
00302   //
00303   int i=0;
00304   int j=v.L-1;
00305   int p;
00306
00307   while((i<=j) && (j>=0))
00308     {
00309       p = (j+i)/2;
00310       if(d < v.V[p]) j = p-1;
00311       else if(d > v.V[p]) i = p+1;
00312       else return p;
00313     }
00314
```

5.79.1.3 int dcomp (const void ∗ *x*, const void ∗ *y*) [inline]

Definition at line 319 of file jvector.h.

```
00320 {
00321   const double * X = (const double *) x;
00322   const double * Y = (const double *) y;
00323   if(*X < *Y) return -1;
00324   if(*X > *Y) return  1;
00325   return 0;
00326
```

5.79.1.4 int doublecompare (const void * x, const void * y)

5.79.2 Variable Documentation

5.79.2.1 const int DEFAULTLEN = 3

Definition at line 25 of file jvector.h.

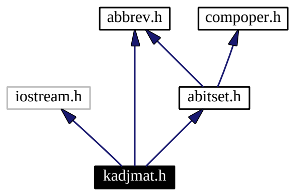
5.80 kadjmat.h File Reference

`#include <iostream.h>`

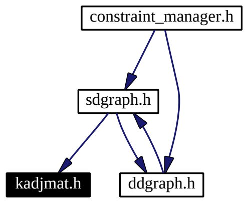
`#include "abitset.h"`

`#include "abbrev.h"`

Include dependency graph for kadjmat.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **KAdjMat**

5.81 Makefile File Reference

5.82 Makefile File Reference

5.83 Makefile File Reference

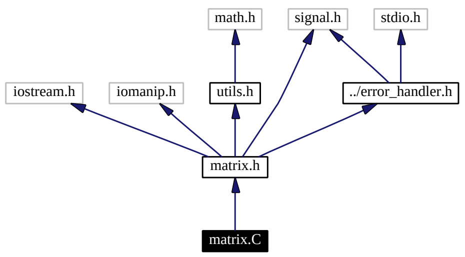
5.84 Makefile File Reference

5.85 Makefile File Reference

5.86 matrix.C File Reference

`#include "matrix.h"`

Include dependency graph for matrix.C:



Functions

- **Matrix operator** ∗ (double d, const **Matrix** &m)

5.86.1 Function Documentation

5.86.1.1 Matrix operator ∗ (double *d*, const Matrix & *m*)

Definition at line 407 of file matrix.C.

```
00407                                              {
00408   int rows = m.getRows(), cols = m.getCols();
00409   Matrix result(rows,cols);
00410   for (int row = 1; row <= rows; row++)
00411     for (int col = 1; col <= cols; col++)
00412       result(row,col) = d * m(row,col);
00413   return result;
00414
```

5.87 matrix.h File Reference

`#include <iostream.h>`

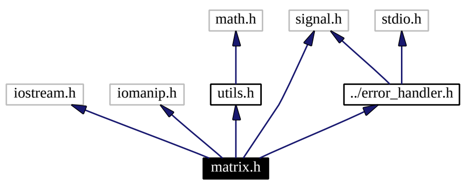
`#include <iomanip.h>`

`#include "utils.h"`

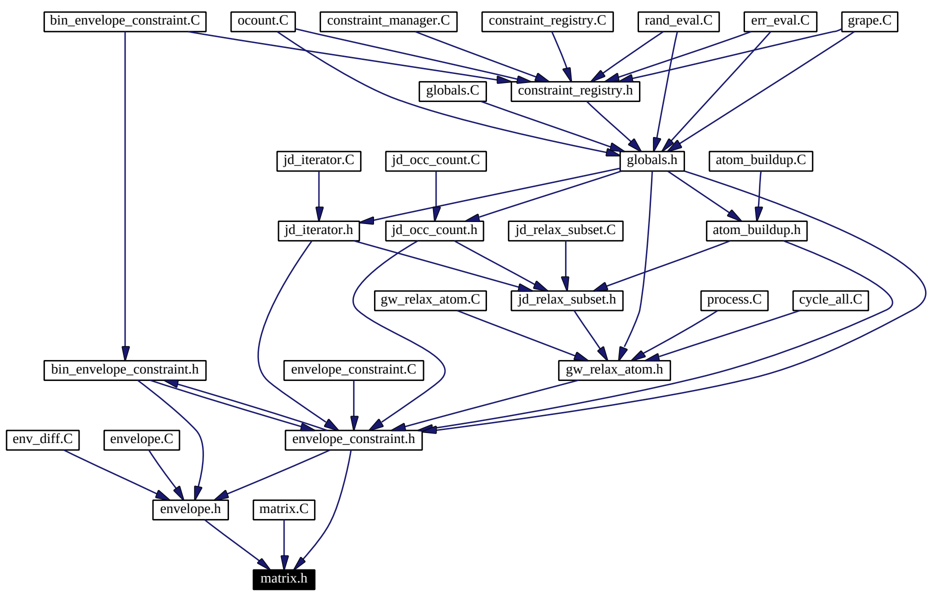
`#include "signal.h"`

`#include "../error_handler.h"`

Include dependency graph for matrix.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Matrix**

Functions

- **Matrix operator** ∗ (double d, const **Matrix** &m)
- ostream& **operator**<< (ostream &os, const **Matrix** &m)

5.87.1 Function Documentation

5.87.1.1 Matrix operator * (double *d*, const Matrix & *m*)

Definition at line 407 of file matrix.C.

```
00407                                         {
00408   int rows = m.getRows(), cols = m.getCols();
00409   Matrix result(rows,cols);
00410   for (int row = 1; row <= rows; row++)
00411     for (int col = 1; col <= cols; col++)
00412       result(row,col) = d * m(row,col);
00413   return result;
00414
```

5.87.1.2 ostream & operator<< (ostream & *os*, const Matrix & *m*) [inline]

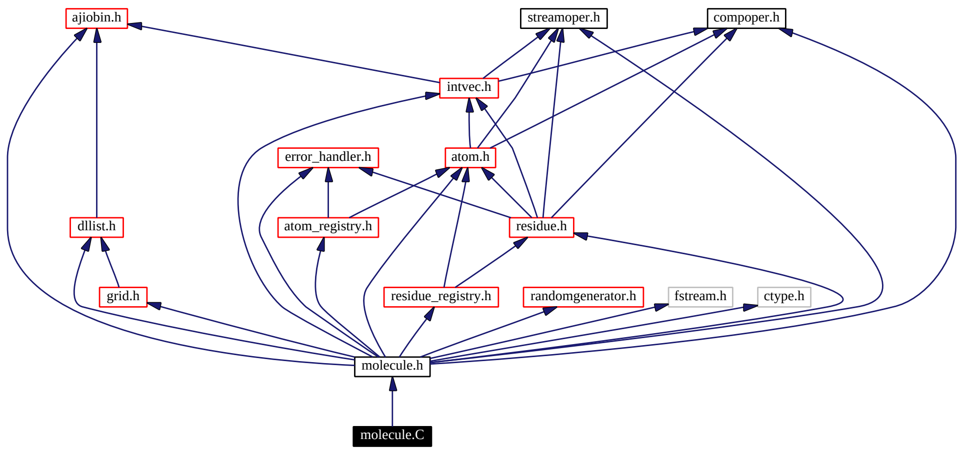
Definition at line 94 of file matrix.h.

```
00094                                                      {
00095   for (int row = 1; row <= m.getRows(); row++) {
00096     os << "[ ";
00097     for (int col = 1; col <= m.getCols(); col++)
00098         os << m(row,col) << " ";
00099     os << "]" << endl;
00100   }
00101
```

5.88 molecule.C File Reference

`#include "molecule.h"`

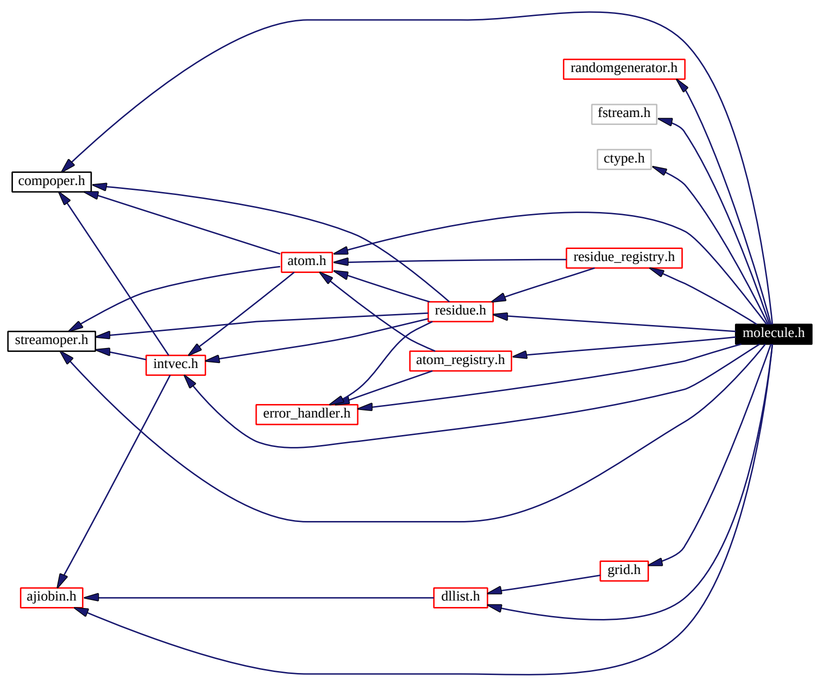
Include dependency graph for molecule.C:



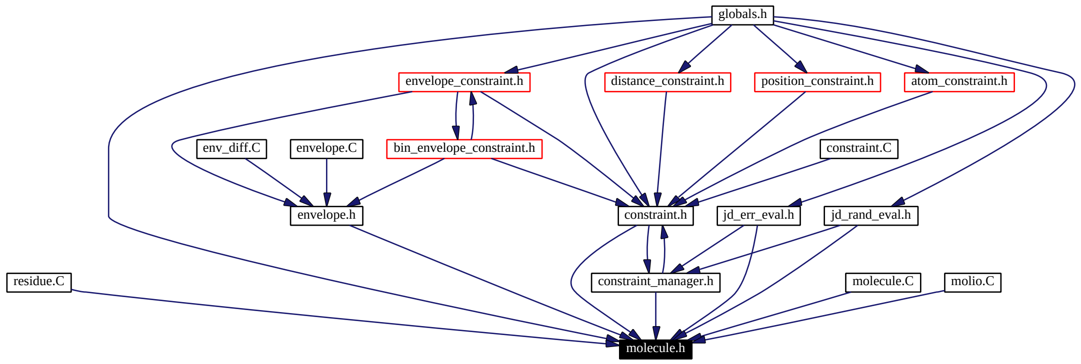
5.89 molecule.h File Reference

```
#include <fstream.h>
#include <ctype.h>
#include "compoper.h"
#include "streamoper.h"
#include "ajiobin.h"
#include "dllist.h"
#include "atom_registry.h"
#include "residue_registry.h"
#include "atom.h"
#include "residue.h"
#include "grid.h"
#include "randomgenerator.h"
#include "error_handler.h"
#include "intvec.h"
```

Include dependency graph for molecule.h:



This graph shows which files directly or indirectly include this file:



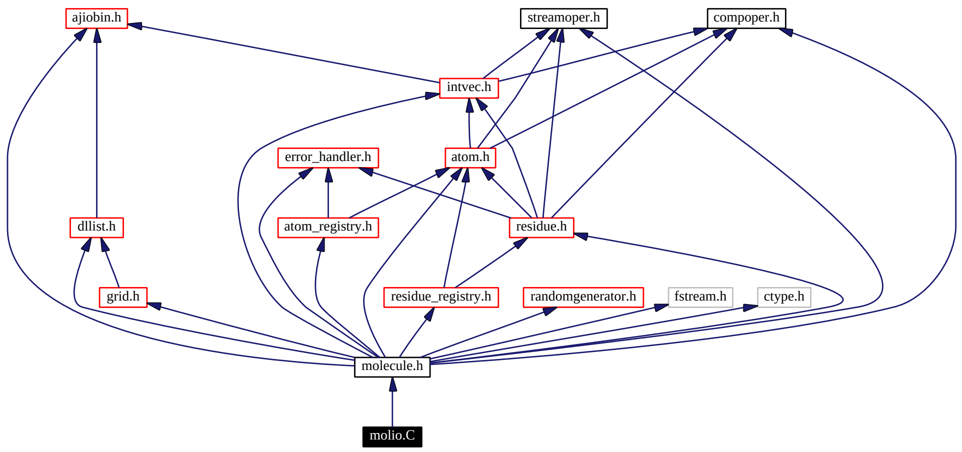
Compounds

- class **Molecule**

5.90 molio.C File Reference

`#include "molecule.h"`

Include dependency graph for molio.C:



Functions

- **main** ()

5.90.1 Function Documentation

5.90.1.1 main ()

Definition at line 4 of file molio.C.

```
{
  Molecule M;
  while(!cin.eof() && !cin.bad())
    {
      M.readXmol(cin);
      M.writeXmol(cout);
    }
```

5.91 ocount.C File Reference

`#include <fstream.h>`

`#include "dynstring.h"`

`#include "dllist.h"`

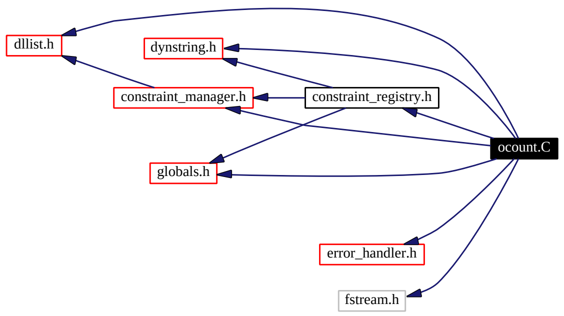
`#include "constraint_manager.h"`

`#include "constraint_registry.h"`

`#include "globals.h"`

`#include "error_handler.h"`

Include dependency graph for ocount.C:



Functions

- void **print_usage** (ostream &os)
  Prints information about usage.
- void **print_help** (ostream &os)
  Prints full help.
- int **main** (int argc, char ∗∗argv)

5.91.1 Function Documentation

5.91.1.1 int main (int *argc*, char ∗∗ *argv*)

Definition at line 17 of file ocount.C.

```
00017                                   {
00018   if(argc < 2)
00019     {
00020       print_usage(cout);
00021       return 0;
00022     }
00023 
00024   cerr << "Begin: ";
00025   for (int i =0; i < argc; i++){
00026     cerr << argv[i]<<" ";
00027   }
```

```
cerr << endl;

DLList<DynString> ArgList;
// List of command line (and parameter file originating) arguments.

Molecule * predictedMolecule=0;
// now the molecule is a private member of CM'

// Iterate through all the command line arguments:
DynString arg;
// Command-line argument.
ifstream * input=0;
// Parameter input stream.
int argStart = 1;

// default output filenamestart string
DynString BaseFN = "ocount-output";


// See if the user has specified a parameter file.  It is a
// parameter file if the first argument does *not* start with
// a dash, '-' (i.e. it is not a flag).
//

if(argv[1][0] != '-')
  {
    // Read in the arguments from the parameter file.  This file should
    // contain flags (including their dashes) followed immediately by
    // the values associated with them.  Warning: this processing step
    // ignores white space!
    //
    input = new ifstream(argv[1]);
    if(!(input->good()))
      {
        cerr << "Could not open/find parameter file, " << argv[1]
             << ", quitting. Try -help option.\n";
        return 1;
      }
    while(input->good())
      {
        *input >> arg;
        cerr << arg << " ";
        if(arg[0] == '#')
          arg.EatLine(*input);
        // Ignore lines that begin with a pound, '#', sign:
        else
          ArgList.AddToBack(arg);
      }
    cerr << endl;
    ArgList.RemoveFromBack();  // Get rid of the eof character.
    delete input; input=0;
    argStart=2;
  }

// Copy the arguments from the command line onto our argument list:
//
for(int a=argStart; a<argc; ++a)
  ArgList.AddToBack(DynString(argv[a]));

while(ArgList)
  {
    arg = ArgList.RemoveFromFront();

    // Initial molecule file is being specified.
    //
```

```
if((arg == DynString("-initial")) ||
   (arg == DynString("-init")) ||
   (arg == DynString("-i")))
  {
    // Check to see if we've already read in a molecule:
    if(predictedMolecule)
      {
        cerr << "Warning: More than one file has been supplied for "
             << "the initial molecule.\nWill use the molecule "
             << "specified in the file listed last.\n";
        delete predictedMolecule; predictedMolecule=0;
      }

    arg = ArgList.RemoveFromFront();  // Get the name of the xmol file.
    if((input = new ifstream(arg.GetData())))
      {
        predictedMolecule = new Molecule;
        if(predictedMolecule->readXmol(*input)<1)
          {
            cerr << "Error reading initial molecule.\n";
            delete predictedMolecule; predictedMolecule=0;
          }
        delete input; input=0;
      }
    else
      {
        cerr << "Could not open initial molecule file, " << arg
             << ", will try default location.\n";
      }
  }

else if(arg == DynString("-f") ||
        arg == DynString("-filebase"))
  {
      arg = ArgList.RemoveFromFront();
      if (arg[0] != '-'){
        BaseFN = arg;
      } else {
        DIE("Confused by filename ");
      }
  }


else if(arg == DynString("-w") ||
        arg == DynString("-width"))
  {
    double width;
    if (ArgList.IsEmpty()) {
      DIE("Error while parsing the vdw arguments. Try -help option.");
    }
    arg = ArgList.RemoveFromFront();
    width = arg.ToDbl();
    jdOcc_Count::cube_width = width;
  }

else if(arg == DynString("-help"))
  {
    print_help(cout);
    return 0;

  }

// Ignore the bogus parameter.
//
```

```
      else
        {
          cerr << "Unknown argument, " << arg << ", given, ignoring.\n";
        }
    }

  // If we havent read in an initial molecule, we'll check the default
  // location for one:
  //
  if(!predictedMolecule)
    {
      cerr << "No initial molecule specified, quitting. Try -help option.\n";
      return 3;
    }

```

5.91.1.2 void print_help (ostream & *os*)

Prints full help.

Definition at line 197 of file ocount.C.

```
{
  os    << "\nHELP for using GRAPE\n\n";
  print_usage(os);
  os    << "Additional Options:\n"
        << "  {-w | -width n}

```

5.91.1.3 void print_usage (ostream & *os*)

Prints information about usage.

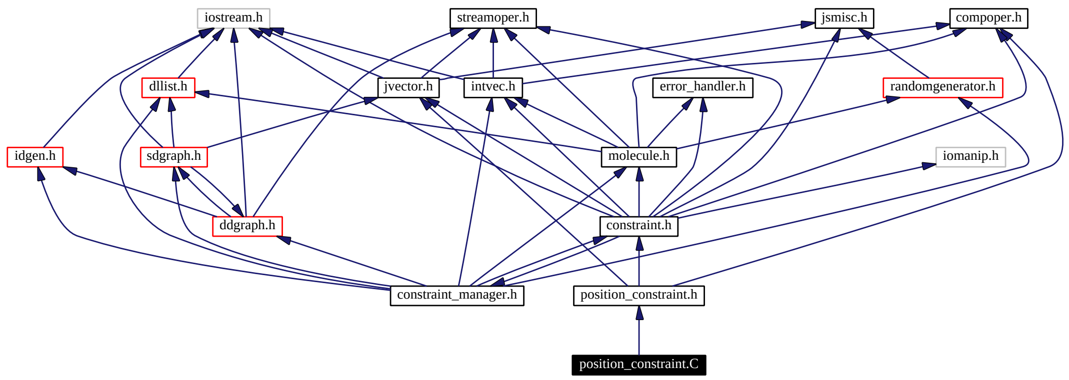
Definition at line 188 of file ocount.C.

```
{
  os    << "\nUsage: ocount <-help> [optionfilename | options] \n\n"
        << "Required options:\n"
        << "  {-initial | -init | -i} <filename> }

```

5.92 position_constraint.C File Reference

`#include "position_constraint.h"`

Include dependency graph for position_constraint.C:



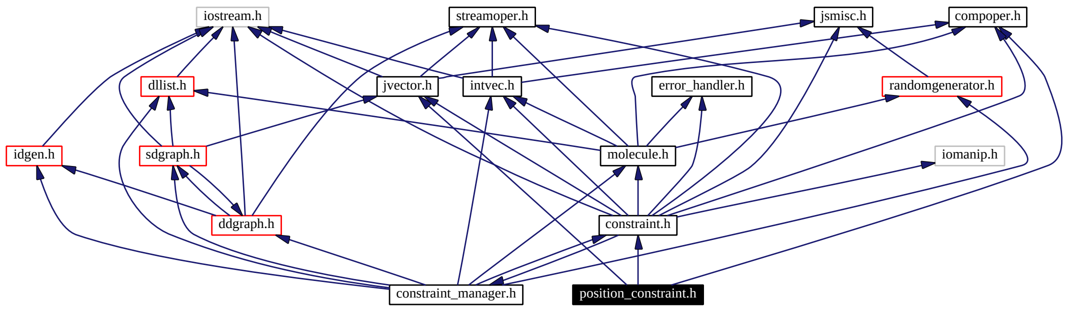
5.93 position_constraint.h File Reference

`#include "constraint.h"`

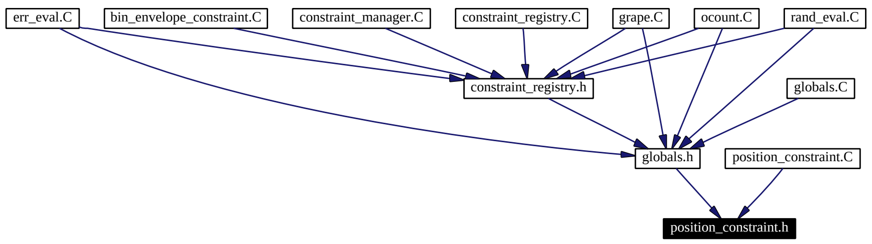
`#include "compoper.h"`

`#include "jvector.h"`

Include dependency graph for position_constraint.h:



This graph shows which files directly or indirectly include this file:



Compounds

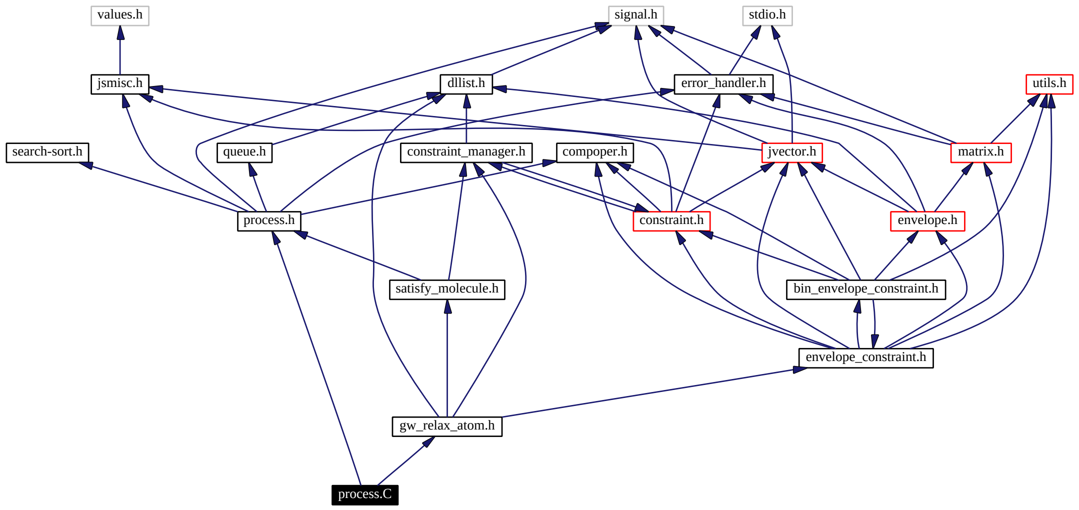
- class **PositionConstraint**

5.94 process.C File Reference

`#include "process.h"`

`#include "gw_relax_atom.h"`

Include dependency graph for process.C:



5.95 process.h File Reference

#include "search-sort.h"

#include "compoper.h"

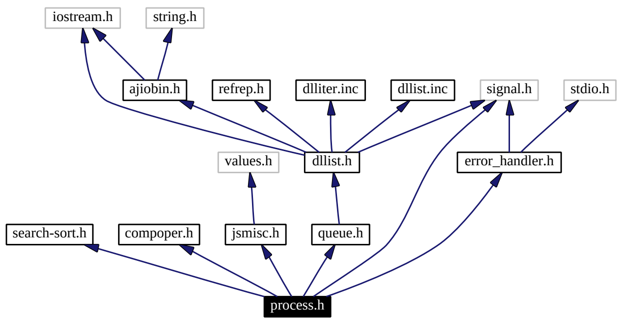
#include "jsmisc.h"

#include "queue.h"

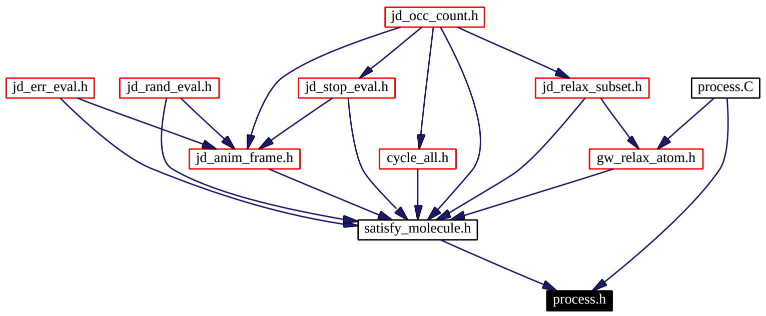
#include "signal.h"

#include "error_handler.h"

Include dependency graph for process.h:



This graph shows which files directly or indirectly include this file:



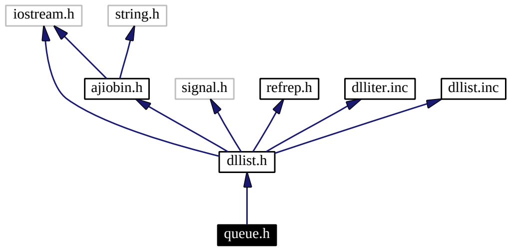
Compounds

- class **Process**

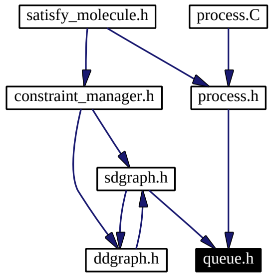
5.96 queue.h File Reference

`#include "dllist.h"`

Include dependency graph for queue.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Queue**

5.97 rand_eval.C File Reference

`#include <fstream.h>`

`#include "dynstring.h"`

`#include "dllist.h"`

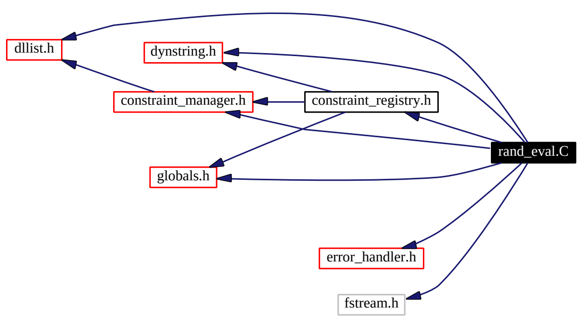
`#include "constraint_manager.h"`

`#include "constraint_registry.h"`

`#include "globals.h"`

`#include "error_handler.h"`

Include dependency graph for rand_eval.C:



Functions

- void **print_usage** (ostream &os)

  Prints information about usage.

- void **print_help** (ostream &os)

  Prints full help.

- int **main** (int argc, char **argv)

5.97.1 Function Documentation

5.97.1.1 int main (int *argc*, char ** *argv*)

Definition at line 17 of file rand_eval.C.

```
00017                                  {
00018   if(argc < 2)
00019     {
00020       print_usage(cout);
00021       return 0;
00022     }
00023
00024   cerr << "Begin: ";
00025   for (int i =0; i < argc; i++){
00026     cerr << argv[i]<<" ";
00027   }
```

```
cerr << endl;

DLList<DynString> ArgList;

DLList<DynString> ConstraintFileList;

Molecule * predictedMolecule=0;

DynString arg;

ifstream * XmolInputFile=0;


ifstream * input=0;

int argStart = 1;

DynString BaseFN = "evalenv-output";

int cycle = 10;
double pert = 0.05;

// See if the user has specified a parameter file.  It is a
// parameter file if the first argument does *not* start with
// a dash, '-' (i.e. it is not a flag).
//

if(argv[1][0] != '-')
  {
    // Read in the arguments from the parameter file.  This file should
    // contain flags (including their dashes) followed immediately by
    // the values associated with them.  Warning: this processing step
    // ignores white space!
    //
    if(!(input = new ifstream(argv[1])))
      {
        cerr << "Could not open/find parameter file, " << argv[1]
             << ", quitting. Try -help option.\n";
        return 1;
      }
    while(!input->eof() && !input->bad())
      {
        *input >> arg;

        if(arg[0] == '#')
          arg.EatLine(*input);
        // Ignore lines that begin with a pound, '#', sign:
        else
          ArgList.AddToBack(arg);
      }
    ArgList.RemoveFromBack();  // Get rid of the eof character.

    delete input; input=0;
    argStart=2;
  }

// Copy the arguments from the command line onto our argument list:
//
for(int a=argStart; a<argc; ++a)
  ArgList.AddToBack(DynString(argv[a]));

while(ArgList)
  {
    arg = ArgList.RemoveFromFront();

```

```
// Initial molecule file is being specified.
//

if((arg == DynString("-initial")) ||
   (arg == DynString("-init")) ||
   (arg == DynString("-i")))
  {
    // Check to see if we've already read in a molecule:
    if(predictedMolecule)
      {
        cerr << "Warning: More than one file has been supplied for "
             << "the initial molecule.\nWill use the molecule "
             << "specified in the file listed last.\n";
        delete predictedMolecule; predictedMolecule=0;
      }

    arg = ArgList.RemoveFromFront();  // Get the name of the xmol file.
    if((XmolInputFile = new ifstream(arg.GetData())))
      {
        predictedMolecule = new Molecule;
        if(predictedMolecule->readXmol(*XmolInputFile)<1)
          {
            cerr << "Error reading initial molecule.\n";
            delete predictedMolecule; predictedMolecule=0;
          }
      }
    else
      {
        cerr << "Could not open initial molecule file, " << arg
             << ", will try default location.\n";
      }
  }

else if((arg == DynString("-constraint")) ||
        (arg == DynString("-constraints")) ||
        (arg == DynString("-c")))
  {
    // Just add this file to the list of files that contain
    // constraints for right now.  We'll read these in once
    // we're sure we have an initial molecule.
    //
    arg = ArgList.RemoveFromFront();
    ConstraintFileList.AddToBack(arg);
  }

else if(arg == DynString("-f") ||
        arg == DynString("-filebase"))
  {
      arg = ArgList.RemoveFromFront();
      if (arg[0] != '-'){
        BaseFN = arg;
      } else {
        DIE("Confused by filename ");
      }
  }

else if(arg == DynString("-cycle"))
  {
    arg = ArgList.RemoveFromFront();
    if (arg[0] != '-'){
      cycle = arg.ToInt();
    } else {
      DIE("Confused by cycle arg ");
    }
  }
```

```

      else if(arg == DynString("-pert"))
        {
          arg = ArgList.RemoveFromFront();
          if (arg[0] != '-'){
            pert = arg.ToDbl();
          } else {
            DIE("Confused by pert arg ");
          }
        }

      else if(arg == DynString("-help"))
        {
          print_help(cout);
          return 0;

        }

      // Ignore the bogus parameter.
      //
      else
        {
          cerr << "Unknown argument, " << arg << ", given, ignoring.\n";
        }
    }

  // If we havent read in an initial molecule, we'll check the default
  // location for one:
  //
  if(!predictedMolecule)
    {
      cerr << "No initial molecule specified, quitting. Try -help option.\n";
      return 3;
    }

  // Now that we've got an actual molecule, we can create a constraint
  // manager and read in the constraints:
  //

  ConstraintManager CM(*predictedMolecule);

  int num_read = 0;

  while(ConstraintFileList)
    {
      // Get the filename and then open the file and then read the
      // constraints in using the constraint manager.
      arg = ConstraintFileList.RemoveFromFront();
      input = new ifstream(arg.GetData());
      if(input->good())
        {
          // Use the registry's routine to read in constraints and
          // append them to the constraint manager.
          //

          num_read += ConstraintRegistry::readConstraints(*input,CM);
          delete input; input=0;
        } else {
          cerr << "\n\nWhile reading file: " << arg.GetData() << endl;
          DIE ("Cannot open file");
        }

    }

  if (num_read == 0) {
```

```
    DIE ("You have no constraints");
  }

  jdRand_Eval myeval(CM, XmolInputFile, cycle, pert);
  myeval.BaseFN = BaseFN;
  myeval.run();

  delete predictedMolecule;
  if (XmolInputFile)
    {delete XmolInputFile;XmolInputFile =0;}

```

5.97.1.2 void print_help (ostream & *os*)

Prints full help.

Definition at line 260 of file rand_eval.C.

Referenced by main().

```
{
  os    << "\nHELP\n\n";
  print_usage(os);
  os    << "Additional Options:\n"
        << "  -cycle n   # perturb for n cycles\n"
        << "  -pert  n    # move each atom up to +/- n Angstoms in each direction\n";


```

5.97.1.3 void print_usage (ostream & *os*)

Prints information about usage.

Definition at line 246 of file rand_eval.C.

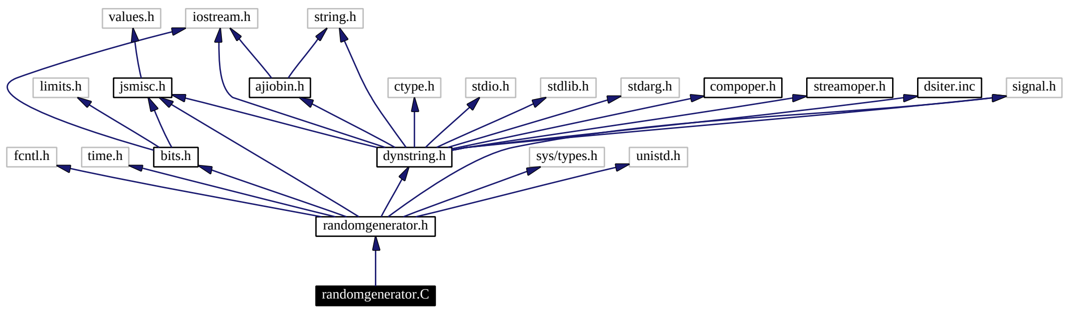
Referenced by main(), and print_help().

```
{
  os    << "\nUsage: rand_eval <-help> [optionfilename | options] \n\n"
        << "Required options:\n"
        << "  {-initial | -init | -i} <filename> }\n"
        << "     Initial molecule: xmol file\n"
        << "  {-constraints | -c}

```

5.98 randomgenerator.C File Reference

`#include "randomgenerator.h"`

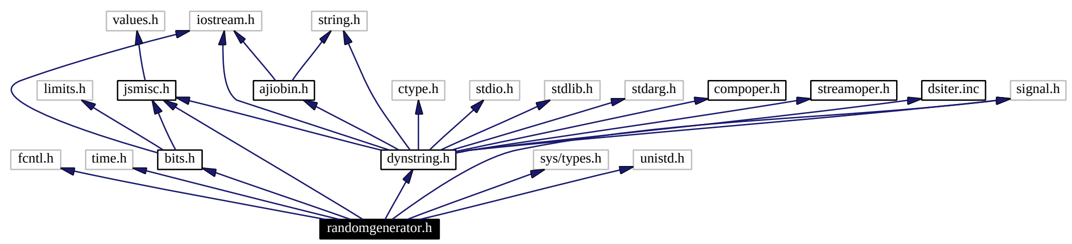
Include dependency graph for randomgenerator.C:



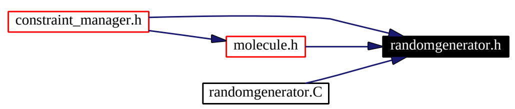
5.99 randomgenerator.h File Reference

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <time.h>
#include <bits.h>
#include <signal.h>
#include "jsmisc.h"
#include "dynstring.h"
```

Include dependency graph for randomgenerator.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RanBaseClass**
- class **RanChar**
- class **RanDbl**
- class **RanInt**
- class **RanLetter**
- class **RanWord**
- class **z**

Variables

- class z **z**

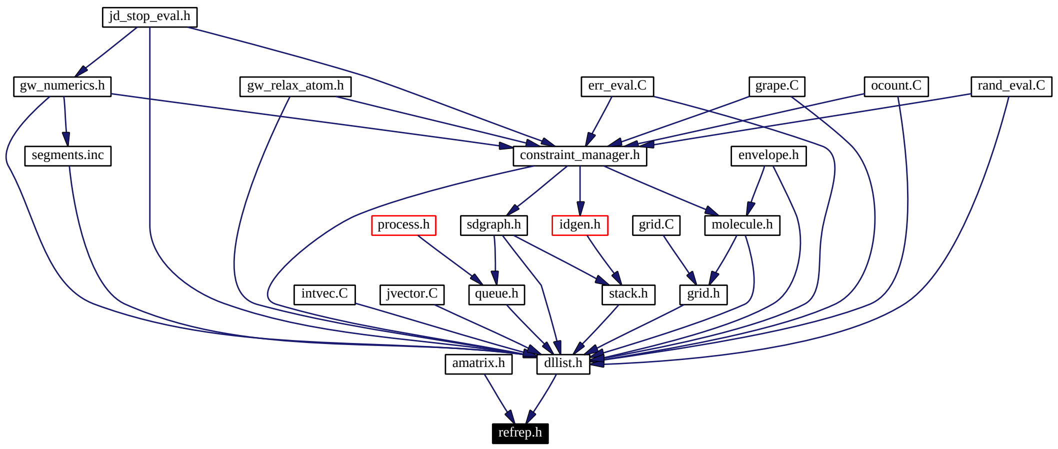
5.99.1 Variable Documentation

5.99.1.1 class z z [static]

Referenced by Matrix::findEigenVectors().

5.100 refrep.h File Reference

This graph shows which files directly or indirectly include this file:



Compounds

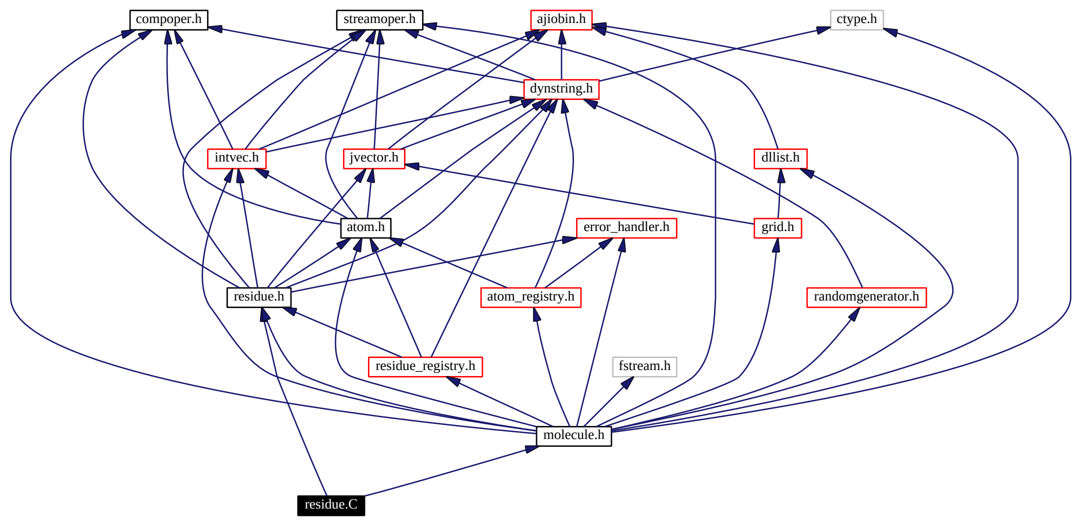
- class **RefRep**

5.101 residue.C File Reference

`#include "residue.h"`

`#include "molecule.h"`

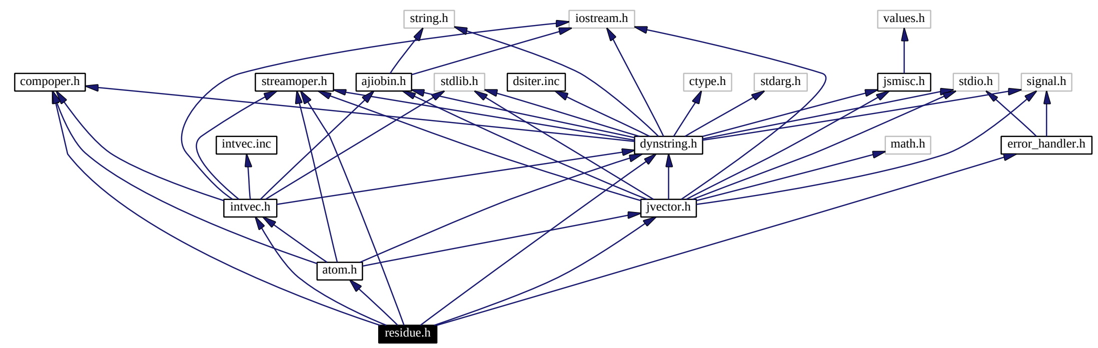
Include dependency graph for residue.C:



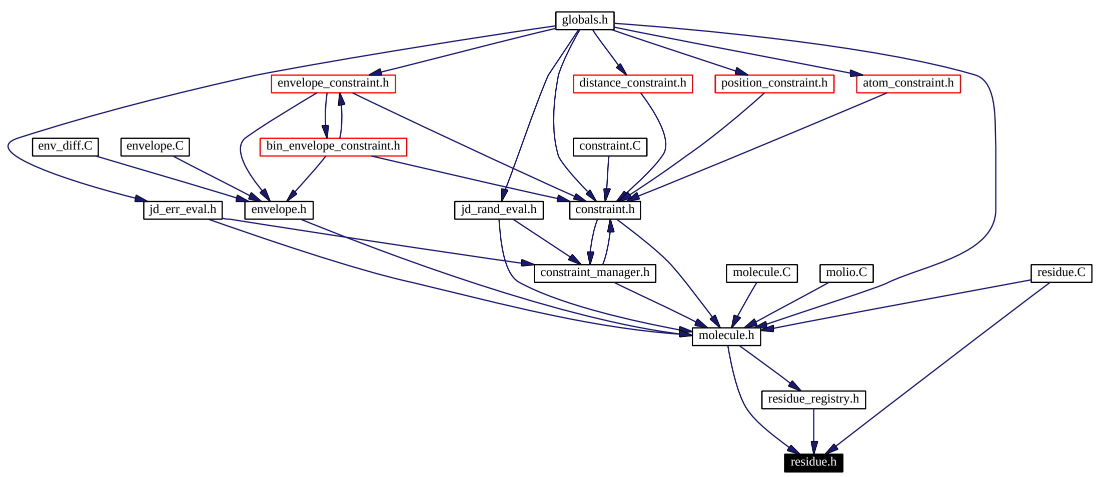
5.102 residue.h File Reference

```
#include "intvec.h"
#include "dynstring.h"
#include "atom.h"
#include "jvector.h"
#include "compoper.h"
#include "streamoper.h"
#include "error_handler.h"
```

Include dependency graph for residue.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Residue**

Defines

- #define **UNINITIALIZED_RES_ID** -1

5.102.1 Define Documentation

5.102.1.1 #define UNINITIALIZED_RES_ID -1

//

File: **residue.h**

Used to save information about one residue.

//

Definition at line 23 of file residue.h.

5.103 residue_registry.h File Reference

`#include <signal.h>`

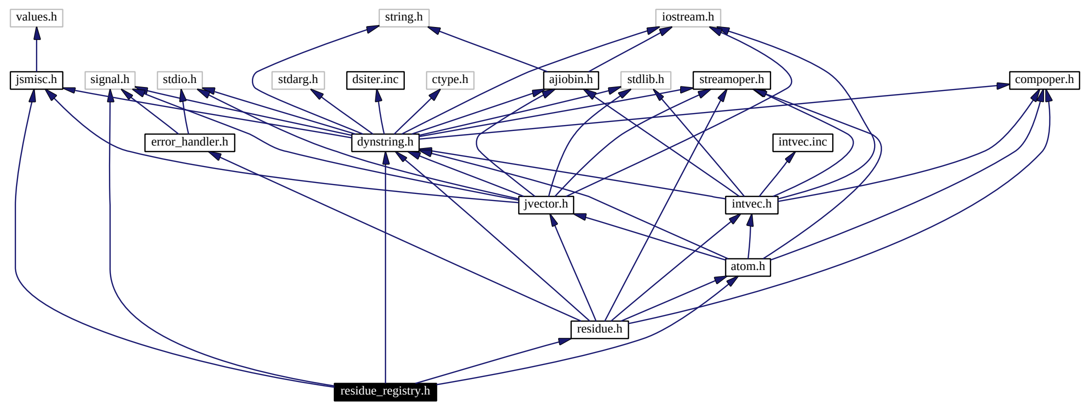
`#include "dynstring.h"`

`#include "residue.h"`

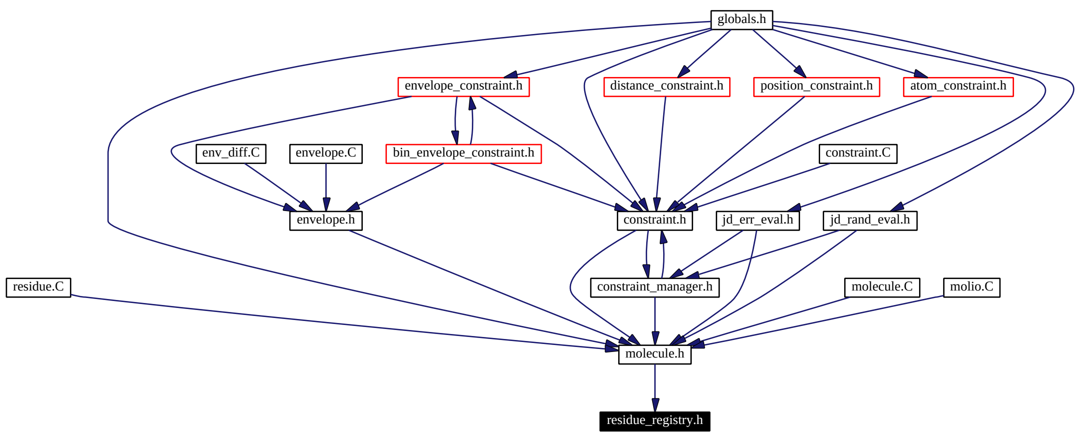
`#include "atom.h"`

`#include "jsmisc.h"`

Include dependency graph for residue_registry.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ResidueRegistry**

Defines

- #define **num_known_residue_species** 21

5.103.1 Define Documentation

5.103.1.1 #define num_known_residue_species 21

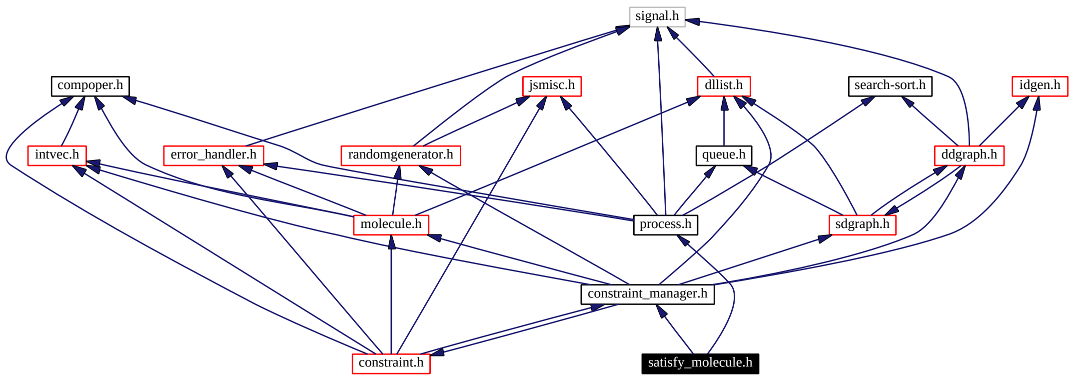
Definition at line 17 of file residue_registry.h.

5.104 satisfy_molecule.h File Reference

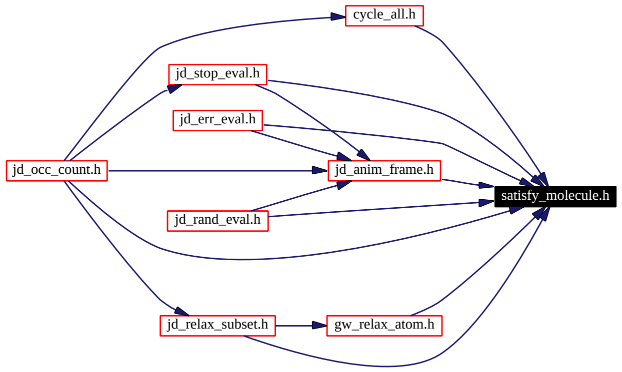
`#include "process.h"`

`#include "constraint_manager.h"`

Include dependency graph for satisfy_molecule.h:



This graph shows which files directly or indirectly include this file:



Compounds

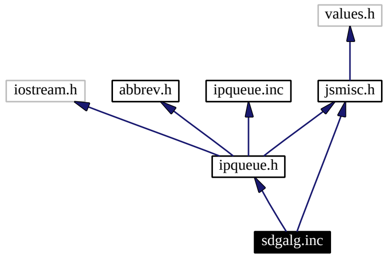
- class **SatisfyMolecule**

5.105 sdgalg.inc File Reference

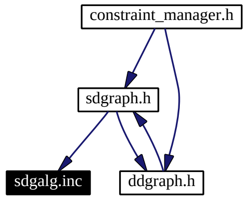
`#include "ipqueue.h"`

`#include "jsmisc.h"`

Include dependency graph for sdgalg.inc:

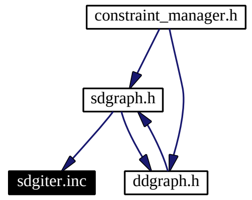


This graph shows which files directly or indirectly include this file:



5.106 sdgiter.inc File Reference

This graph shows which files directly or indirectly include this file:



Compounds

- class **base_iterator**
- class **bf_iterator**
- class **df_iterator**
- class **neighbor_iterator**
- class **path_iterator**

5.107 sdgraph.h File Reference

`#include <iostream.h>`

`#include "sdvertex.h"`

`#include "ddgraph.h"`

`#include "dllist.h"`

`#include "stack.h"`

`#include "queue.h"`

`#include "kadjmat.h"`

`#include "abbrev.h"`

`#include "abitset.h"`

`#include "ajiobin.h"`

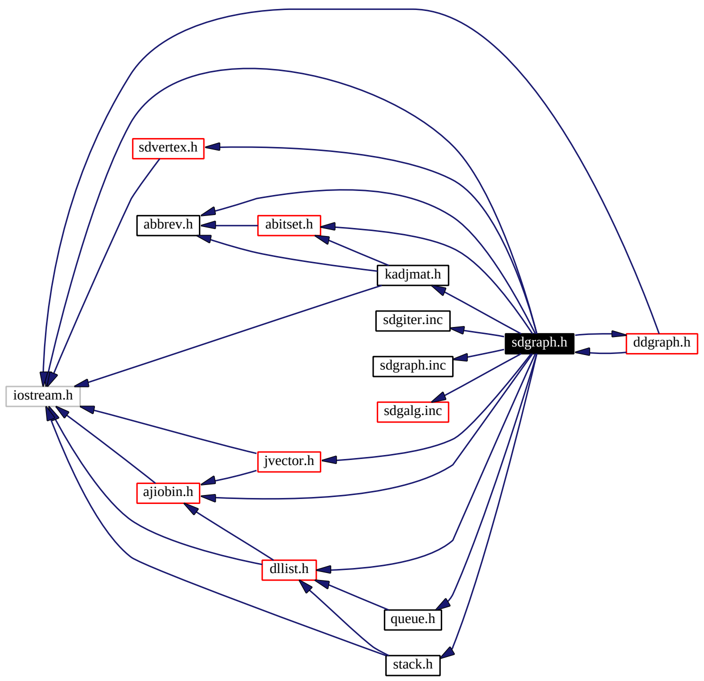
`#include "jvector.h"`

`#include <sdgiter.inc>`

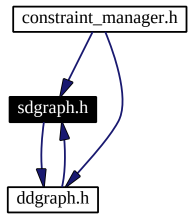
`#include <sdgraph.inc>`

`#include <sdgalg.inc>`

Include dependency graph for sdgraph.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **sdGraph**

Typedefs

- typedef set<int> **SDG_SetInt**
- typedef set<**SDG_SetInt**> **SDG_SetSetInt**
- typedef **DLList**<int> **SDG_IntList**
- typedef **sdGraph**<**SDG_SetInt**, class WT> **SDG_GraphSetInt**

Enumerations

- enum **sdg_color** { **sdg_white** = 0, **sdg_gray** = 1, **sdg_black** = 2 }

5.107.1 Typedef Documentation

5.107.1.1 typedef sdGraph< SDG_SetInt, class WT > SDG_GraphSetInt

Definition at line 63 of file sdgraph.h.

5.107.1.2 typedef DLList< int > SDG_IntList

Definition at line 41 of file sdgraph.h.

5.107.1.3 typedef set< int > SDG_SetInt

Definition at line 38 of file sdgraph.h.

5.107.1.4 typedef set< SDG_SetInt > SDG_SetSetInt

Definition at line 40 of file sdgraph.h.

5.107.2 Enumeration Type Documentation

5.107.2.1 enum sdg_color

Enumeration values:
sdg_white

sdg_gray

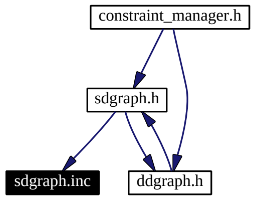
sdg_black

Definition at line 65 of file sdgraph.h.

```
00066 {
00067   sdg_white     = 0,
00068   sdg_gray      = 1,
00069   sdg_black     = 2
00070
```

5.108 sdgraph.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- template<class DT, class WT> ostream& **operator<<** (ostream &os, const **sdGraph**< DT, WT > &graph)
- istream& **operator>=** (istream &is, **sdGraph**< DT, WT > &sdg)
- template<class DT, class WT> ostream& **operator<=** (ostream &os, const **sdGraph**< DT, WT > &sdg)

5.108.1 Function Documentation

5.108.1.1 template<class DT, class WT> ostream & operator<< (ostream & *os*, const sdGraph< DT, WT > & *graph*) `[inline]`

Definition at line 661 of file sdgraph.inc.

```
{ graph.Output(os); return os; }
```

5.108.1.2 template<class DT, class WT> ostream & operator<= (ostream & *os*, const sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 948 of file sdgraph.inc.

```
{ sdg.OutputBinary(os); return os; }
```

5.108.1.3 istream & operator>= (istream & *is*, sdGraph< DT, WT > & *sdg*) `[inline]`

Definition at line 944 of file sdgraph.inc.

```
{ sdg.InputBinary(is); return is; }
```

5.109 sdvertex.h File Reference

#include <iostream.h>

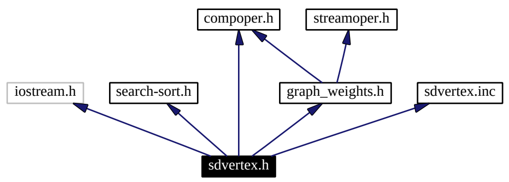
#include <search-sort.h>

#include <compoper.h>

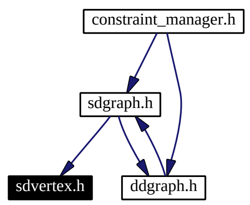
#include <graph_weights.h>

#include <sdvertex.inc>

Include dependency graph for sdvertex.h:



This graph shows which files directly or indirectly include this file:

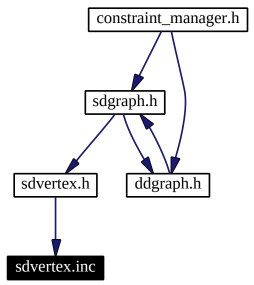


Compounds

- class **sdEdge**
- class **sdVertex**

5.110 sdvertex.inc File Reference

This graph shows which files directly or indirectly include this file:



Functions

- template<class WT> ostream& **operator**<< (ostream &os, const **sdVertex**< WT > &vert)
- int **Connected** (const **sdVertex**< WT > &v1, const **sdVertex**< WT > &v2)

5.110.1 Function Documentation

5.110.1.1 int Connected (const sdVertex< WT > & *v1*, const sdVertex< WT > & *v2*) `[inline]`

Definition at line 244 of file sdvertex.inc.

```
00245   { return (v1.HasEdgeTo(v2) || v1.HasEdgeFrom(v2)); }
00246
```

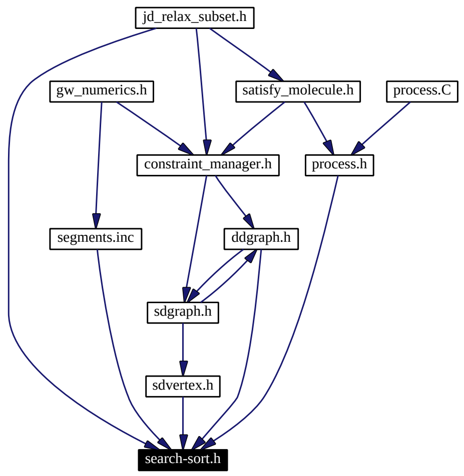
5.110.1.2 template<class WT> ostream & operator<< (ostream & *os*, const sdVertex< WT > & *vert*) `[inline]`

Definition at line 122 of file sdvertex.inc.

```
00123   { os << vert.ID; return os; }
00124
```

5.111 search-sort.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void **search_swap** (T &t1, T &t2)
- void **BubbleSort** (T ∗A, int num)
- void **BubbleSortPtrs** (T ∗∗A, int num)
- void **BubbleSortR** (T ∗A, int num)
- void **BubbleSortPtrsR** (T ∗∗A, int num)
- int **Partition** (T ∗A, int p, int r)
- int **PartitionPtrs** (T ∗∗A, int p, int r)
- void **RecQuickSort** (T ∗A, int p, int r)
- void **RecQuickSortPtrs** (T ∗∗A, int p, int r)
- void **QuickSort** (T ∗A, int Num)
- void **QuickSortPtrs** (T ∗∗A, int Num)
- int **BinSearch** (const T ∗A, int Num, const T &x)
- int **BinSearchPtr** (T ∗∗A, int Num, const T &x)
- int **SelectPartition** (T ∗A, int p, int r, T x)
- double **LinearSelect** (double ∗L, int k, int n)

5.111.1 Function Documentation

5.111.1.1 int BinSearch (const T ∗ *A*, int *Num*, const T & *x*) `[inline]`

Definition at line 192 of file search-sort.h.

Referenced by ddGraph::Copy(), sdVertex::GetInEdgeIndex(), sdVertex::GetOutEdgeIndex(), and sdGraph::GetVertexIndex().

```
00193 {
00194   // left index, right index, pivot, respectively.
00195   //
00196   int i=0;
00197   int j=Num-1;
```

```
  int p;

  while((i<=j) && (j>=0))
    {
      p = (j+i)/2;
      if(x < A[p]) j = p-1;
      else if(x > A[p]) i = p+1;
      else return p;
    }

```

5.111.1.2 int BinSearchPtr (T ** *A*, int *Num*, const T & *x*) [inline]

Definition at line 211 of file search-sort.h.

```
{
  // left index, right index, pivot, respectively.
  //
  int i=0;
  int j=Num-1;
  int p;

  while((i<=j) && (j>=0))
    {
      p = (j+i)/2;
      if(x < *(A[p])) j = p-1;
      else if(x > *(A[p])) i = p+1;
      else return p;
    }

```

5.111.1.3 void BubbleSort (T * *A*, int *num*) [inline]

Definition at line 51 of file search-sort.h.

```
                                      {
bool stillGoing;
do {
  stillGoing = false;
  for( int i=0; i<num-1; i++ ) {
    if( A[i] > A[i+1] ) {
      search_swap( A[i], A[i+1] );
      stillGoing = true;
    }
  }
}

```

5.111.1.4 void BubbleSortPtrs (T ** *A*, int *num*) [inline]

Definition at line 65 of file search-sort.h.

```
                                           {
bool stillGoing;
do {
  stillGoing = false;
  for( int i=0; i<num-1; i++ ) {
    if( *(A[i]) > *(A[i+1]) ) {
```

```
00071         search_swap( A[i], A[i+1] );
00072         stillGoing = true;
00073       }
00074     }
00075   }
00076
```

5.111.1.5 void BubbleSortPtrsR (T ** *A*, int *num*) [inline]

Definition at line 93 of file search-sort.h.

```
00093                                           {
00094   bool stillGoing;
00095   do {
00096     stillGoing = false;
00097     for( int i=num-1;i>0; i-- ) {
00098       if( *(A[i]) < *(A[i-1]) ) {
00099         search_swap( A[i], A[i-1] );
00100         stillGoing = true;
00101       }
00102     }
00103   }
00104
```

5.111.1.6 void BubbleSortR (T * *A*, int *num*) [inline]

Definition at line 79 of file search-sort.h.

```
00079                                      {
00080   bool stillGoing;
00081   do {
00082     stillGoing = false;
00083     for( int i=num-1; i>0; i-- ) {
00084       if( A[i] < A[i-1] ) {
00085         search_swap( A[i], A[i-1] );
00086         stillGoing = true;
00087       }
00088     }
00089   }
00090
```

5.111.1.7 double LinearSelect (double * *L*, int *k*, int *n*) [inline]

Definition at line 263 of file search-sort.h.

```
00263                                                  {
00264   // returns the k'th smallest element in L.
00265   // !! k is in the set {1 -> n} for an n element list. !! (not 0->n-1)
00266   // n represents the number of elements in L.
00267
00268   int i;
00269   if (n <=10) {
00270     RecQuickSort(L, 0, n-1);
00271     return L[k-1];
00272   }
00273   int N = (int) ((n+4)/5);
00274   double ** S = new double * [N];
00275   int j = 0;
```

```
for (i = 0; i < (N-1); i++){
  S[i] = new double [5];
  for (int u =0; u<5; u++) S[i][u] = L[j++];
}
S[N-1] = new double [5];
int rem = n-j;
for (i = 0; i < rem; i++){
  S[N-1][i] = L[j+i];
}
double * X = new double [N];
for (i = 0;i<N-1;i++){
  RecQuickSort(S[i], 0, 4);
  X[i] = S[i][2];
}
if (rem == 1) X[N-1] = S[i][0];
else {
  RecQuickSort(S[i], 0, rem-1);
  if (rem == 4) X[N-1] = S[i][2];
  else X[N-1] = S[i][1];
}
for (i = 0;i<N;i++){
  delete [] S[i];
}

```

5.111.1.8 int Partition (T * *A*, int *p*, int *r*) [inline]

Definition at line 107 of file search-sort.h.

Referenced by RecQuickSort().

```
{
  T tmp;
  T x = A[p];
  int i = p-1;
  int j = r+1;

  while(1)
    {
      --j;
      while(A[j]>x) { --j; }
      ++i;
      while(A[i]<x) { ++i; }

      //
      // Swap the values at i and j.
      if(i<j)
        {
          tmp  = A[i];
          A[i] = A[j];
          A[j] = tmp;
        }
      else return j;
    }

```

5.111.1.9 int PartitionPtrs (T ** *A*, int *p*, int *r*) [inline]

Definition at line 134 of file search-sort.h.

Referenced by RecQuickSortPtrs().

```
00135 {
00136   T * tmp;
00137   T * x = A[p];
00138   int i = p-1;
00139   int j = r+1;
00140 
00141   while(1)
00142     {
00143       --j;
00144       while(*(A[j])>*x) { --j; }
00145       ++i;
00146       while(*(A[i])<*x) { ++i; }
00147 
00148       if(i<j)
00149         {
00150           tmp  = A[i];
00151           A[i] = A[j];
00152           A[j] = tmp;
00153         }
00154       else return j;
00155     }
00156 
```

5.111.1.10 void QuickSort (T * *A*, int *Num*) [inline]

Definition at line 181 of file search-sort.h.

Referenced by sdGraph::Copy(), sdVertex::SortEdges(), and sdGraph::SubGraph().

```
00182 {
00183   if(Num<2) return;
00184   RecQuickSort(A, 0, Num-1);
00185 
```

5.111.1.11 void QuickSortPtrs (T ** *A*, int *Num*) [inline]

Definition at line 188 of file search-sort.h.

Referenced by Process::assertOrder().

```
00189 { if(Num<2) return; RecQuickSortPtrs(A,0,Num-1); }
00190 
```

5.111.1.12 void RecQuickSort (T * *A*, int *p*, int *r*) [inline]

Definition at line 159 of file search-sort.h.

Referenced by LinearSelect(), Segments::QSortReduce(), and QuickSort().

```
00160 {
00161   if(p<r)
00162     {
00163       int q = Partition(A,p,r);
00164       RecQuickSort(A,p,q);
00165       RecQuickSort(A,q+1,r);
00166     }
00167 
```

5.111.1.13 void RecQuickSortPtrs (T ** *A*, int *p*, int *r*) [inline]

Definition at line 170 of file search-sort.h.

Referenced by QuickSortPtrs().

```
{
  if(p<r)
    {
      int q = PartitionPtrs(A,p,r);
      RecQuickSortPtrs(A,p,q);
      RecQuickSortPtrs(A,q+1,r);
    }

```

5.111.1.14 int SelectPartition (T * *A*, int *p*, int *r*, T *x*) [inline]

Definition at line 230 of file search-sort.h.

Referenced by LinearSelect().

```
{
  // as above, but able to specify partion element
  T tmp;
  int i = p-1;
  int j = r+1;

  while(1)
    {
      --j;
      while(A[j]>x) { --j; }
      ++i;
      while(A[i]<x) { ++i; }

      //
      // Swap the values at i and j.
      if(i<j)
        {
          tmp  = A[i];
          A[i] = A[j];
          A[j] = tmp;
        }
      else return j;
    }

```

5.111.1.15 void search_swap (T & *t1*, T & *t2*) [static]

Definition at line 42 of file search-sort.h.

Referenced by BubbleSort(), BubbleSortPtrs(), BubbleSortPtrsR(), and BubbleSortR().

```
{
      T t = t1;
      t1 = t2;
      t2 = t;

```

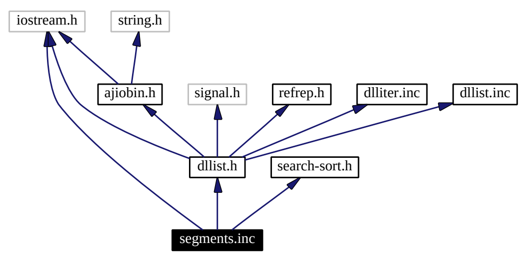
5.112 segments.inc File Reference

`#include <iostream.h>`

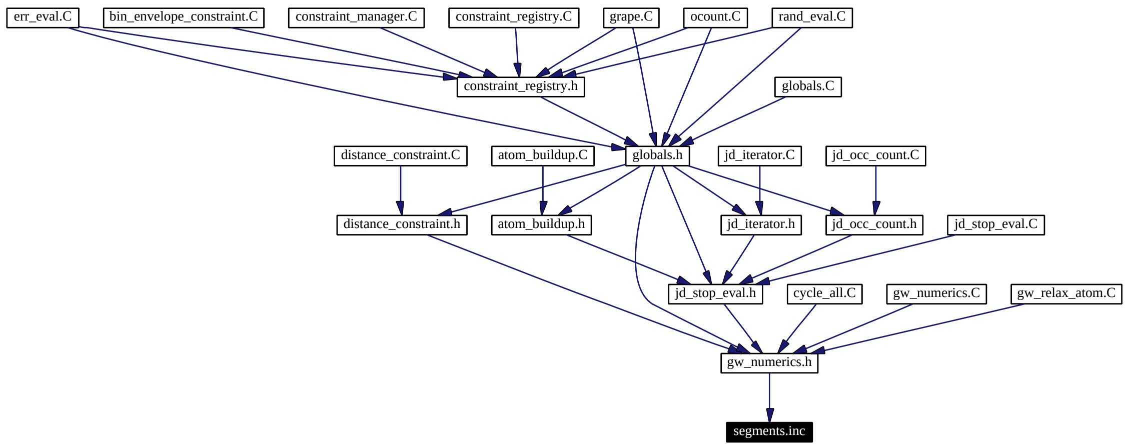
`#include "dllist.h"`

`#include "search-sort.h"`

Include dependency graph for segments.inc:



This graph shows which files directly or indirectly include this file:



Compounds

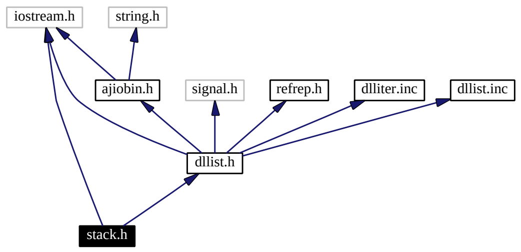
- class **Segment**
- class **Segments**

5.113 stack.h File Reference

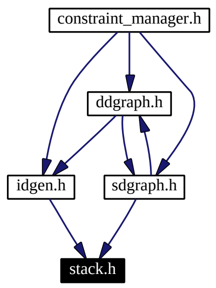
`#include <iostream.h>`

`#include "dllist.h"`

Include dependency graph for stack.h:



This graph shows which files directly or indirectly include this file:

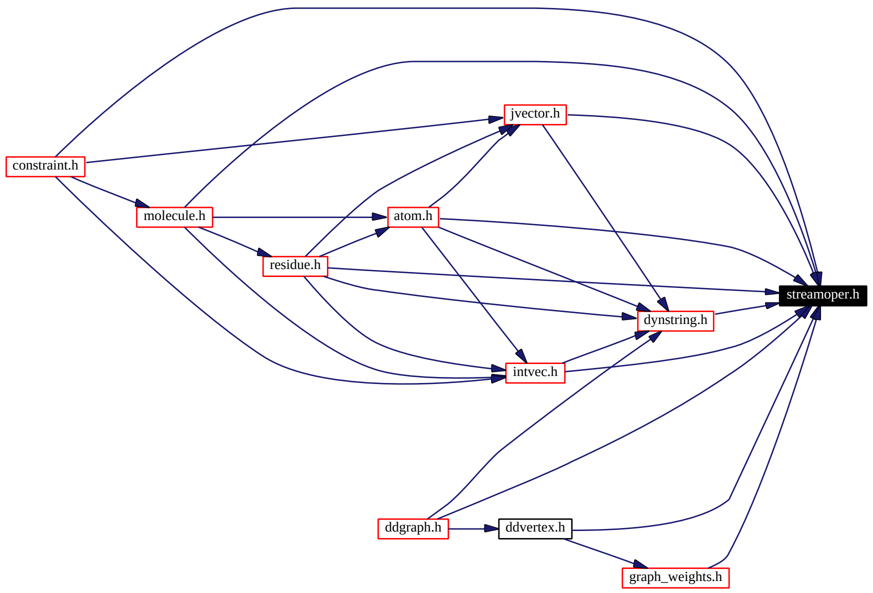


Compounds

- class **Stack**

5.114 streamoper.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **MAKE_IO_STREAM_OPERATORS**(CLASSTYPE_XX)
- #define **MAKE_BIN_STREAM_OPERATORS**(CLASSTYPE_XX)

5.114.1 Define Documentation

5.114.1.1 #define MAKE_BIN_STREAM_OPERATORS(CLASSTYPE_XX)

Value:

```
friend ostream& operator <= (ostream& os, const CLASSTYPE_XX& x) \
        { x.OutputBinary(os); return os; } \
friend istream& operator >= (istream& is, CLASSTYPE_XX& x) \
        { x.InputBinary(is); return is; }
```

Definition at line 16 of file streamoper.h.

5.114.1.2 #define MAKE_IO_STREAM_OPERATORS(CLASSTYPE_XX)

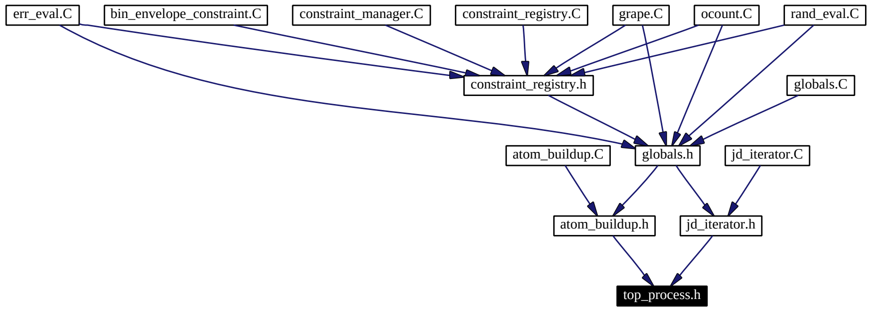
Value:

```
friend ostream& operator << (ostream& os, const CLASSTYPE_XX& x) \
        { x.Output(os); return os; } \
friend istream& operator >> (istream& is, CLASSTYPE_XX& x) \
        { x.Input(is); return is; }
```

Definition at line 10 of file streamoper.h.

5.115 top_process.h File Reference

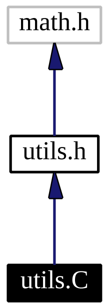
This graph shows which files directly or indirectly include this file:



5.116 utils.C File Reference

```
#include "utils.h"
```

Include dependency graph for utils.C:



Variables

- const char **sccs_ident** [] = "@(#)utils.cpp 2.3 (02/23/98)"
- const int **bit_count_array** [256]

5.116.1 Variable Documentation

5.116.1.1 const int bit_count_array[256]

Initial value:

```
{
0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8,
}
```

Definition at line 5 of file utils.C.

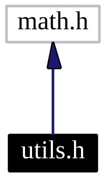
5.116.1.2 const char sccs_ident[] = "@(#)utils.cpp 2.3 (02/23/98)" [static]

Definition at line 3 of file utils.C.

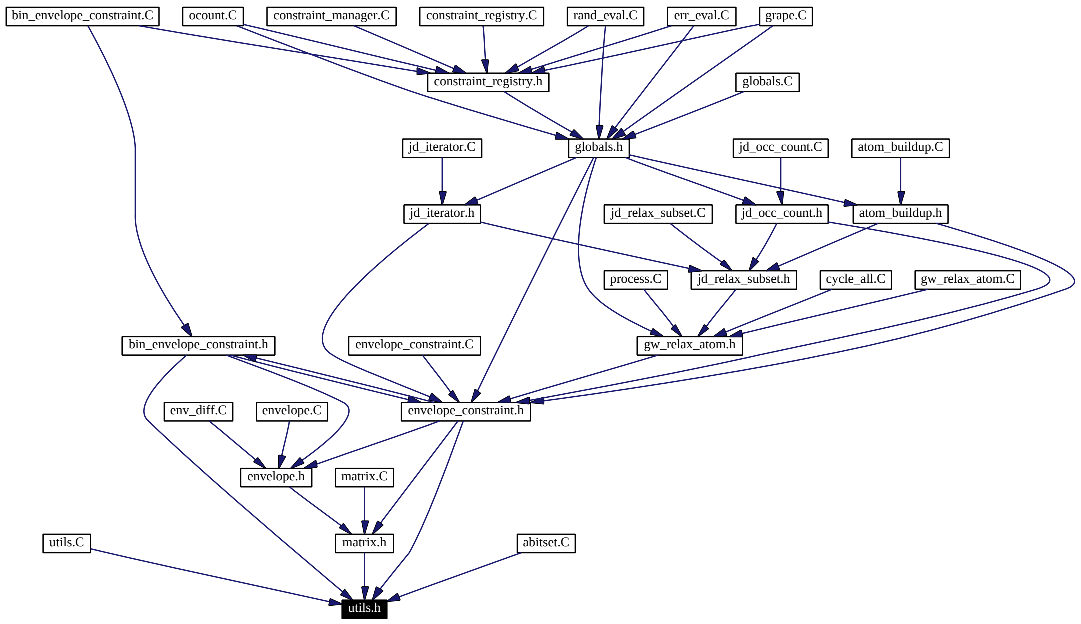
5.117 utils.h File Reference

#include <math.h>

Include dependency graph for utils.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **ERROR**(X)

Functions

- void **util_swap** (T &t1, T &t2)
- T **util_max** (const T &t1, const T &t2)
- T **util_min** (const T &t1, const T &t2)
- T **bound** (const T &t, const T &t1, const T &t2)
- bool **bounded** (const T &t, const T &t1, const T &t2)
- bool **bounds_compare** (const T &t, const T &t1, const T &t2)
- double **mod_dist** (double x, double y, double m)
- int **bit_count** (unsigned char uc)
- int **odd** (int n)
- T **sq** (T &t1)

Variables

- const int **bit_count_array** [256]
- const double **sqrt2**
- const double **sqrt2_2**
- const double **sqrt3**
- const double **sqrt3_2**
- const double **sqrt5**
- const double **sqrt5_2**
- const double **pi**
- const double **pi2**
- const double **pi_2**
- const double **o_sqrt2**
- const double **o_sqrt2_2**
- const double **o_sqrt3**
- const double **o_sqrt3_2**
- const double **o_pi**
- const double **o_pi2**
- const double **o_pi_2**
- const double **rad_to_deg**
- const double **deg_to_rad**

5.117.1 Define Documentation

5.117.1.1 #define ERROR(X)

Value:

```
cerr << "Error: " << __FILE__ << " (" << __LINE__ << ')' \
     << "\t" #X " = " << (X) \
     << endl;
```

Definition at line 133 of file utils.h.

5.117.2 Function Documentation

5.117.2.1 int bit_count (unsigned char *uc*) `[inline]`

Definition at line 88 of file utils.h.

Referenced by ABitSet::Count().

```
{
      return bit_count_array[uc];

```

5.117.2.2 T bound (const T & *t*, const T & *t1*, const T & *t2*) `[static]`

Definition at line 45 of file utils.h.

```
{
      return util_min(util_max(t1,t2), util_max(util_min(t1,t2),t));

```

5.117.2.3 bool bounded (const T & *t*, const T & *t1*, const T & *t2*) [static]

Definition at line 52 of file utils.h.

```
{
        return (t <= util_max(t1,t2)) && (t >= util_min(t1,t2));
```

5.117.2.4 bool bounds_compare (const T & *t*, const T & *t1*, const T & *t2*) [static]

Definition at line 59 of file utils.h.

```
{
        return
                (t > util_max(t1,t2))
                        ? +1
                        : ((t < min(t1,t2))
                                ? -1
                                : 0);
```

5.117.2.5 double mod_dist (double *x*, double *y*, double *m*) [inline]

Definition at line 70 of file utils.h.

```
{
        double  d = x-y;
        if (m == 0.0)
                return d<0.0 ? -d : d;

        if (d<0.0)
                d = -d;

        const double  p = fmod(d, m);
        const double  n = fmod(-p+m, m);
        return util_min(p,n);
```

5.117.2.6 int odd (int *n*) [inline]

Definition at line 139 of file utils.h.

Referenced by Matrix::det().

```
{
  return (n % 2);
```

5.117.2.7 T sq (T & *t1*) [static]

Definition at line 147 of file utils.h.

Referenced by Matrix::findEigenVectors().

```
{
  return (t1 * t1);
```

5.117.2.8 T util_max (const T & *t1*, const T & *t2*) [static]

Definition at line 31 of file utils.h.

Referenced by bound(), bounds_compare(), and BinEnvelopeConstraint::computeStress().

```
{
        return t1>t2 ? t1 : t2;

```

5.117.2.9 T util_min (const T & *t1*, const T & *t2*) [static]

Definition at line 38 of file utils.h.

Referenced by ABitSet::Resize(), bound(), bounded(), mod_dist(), ABitSet::operator &=(), ABitSet::operator-=(), ABitSet::operator^=(), and ABitSet::operator|=().

```
{
        return t1<t2 ? t1 : t2;

```

5.117.2.10 void util_swap (T & *t1*, T & *t2*) [static]

Definition at line 22 of file utils.h.

Referenced by ABitSet::Clear(), ABitSet::Invert(), ABitSet::Set(), Matrix::swapCols(), and Matrix::swapRows().

```
{
        T t = t1;
        t1 = t2;
        t2 = t;

```

5.117.3 Variable Documentation

5.117.3.1 const int bit_count_array[256]

Initial value:

```
{
0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8,
}
```

Definition at line 5 of file utils.C.

5.117.3.2 const double deg_to_rad

Initial value:

```
0.017453292519943295769236907684886127134428718
```

Definition at line 129 of file utils.h.

5.117.3.3 const double o_pi

Initial value:

```
0.318309886183790671537767526745028724068919291
```

Definition at line 120 of file utils.h.

5.117.3.4 const double o_pi2

Initial value:

```
0.15915494309189533576888376337251436203445964
```

Definition at line 122 of file utils.h.

5.117.3.5 const double o_pi_2

Initial value:

```
0.636619772367581343075535053490057448137838582
```

Definition at line 124 of file utils.h.

5.117.3.6 const double o_sqrt2

Initial value:

```
0.707106781186547524400844362104849039284835937
```

Definition at line 112 of file utils.h.

5.117.3.7 const double o_sqrt2_2

Initial value:

```
1.414213562373095048801688724209698078569671875
```

Definition at line 114 of file utils.h.

5.117.3.8 const double o_sqrt3

Initial value:

```
0.577350269189625764509148780501957455647601751
```

Definition at line 116 of file utils.h.

5.117.3.9 const double o_sqrt3_2

Initial value:

```
1.1547005383792515290182975610039149112952035025
```

Definition at line 118 of file utils.h.

5.117.3.10 const double pi

Initial value:

```
3.14159265358979323846264338327950288419716939937
```

Definition at line 105 of file utils.h.

5.117.3.11 const double pi2

Initial value:

```
6.28318530717958647692528676655900576839433879875
```

Definition at line 107 of file utils.h.

5.117.3.12 const double pi_2

Initial value:

```
1.57079632679489661923132169163975144209858469968
```

Definition at line 109 of file utils.h.

5.117.3.13 const double rad_to_deg

Initial value:

```
57.2957795130823208767981548141051703324054724665
```

Definition at line 127 of file utils.h.

5.117.3.14 const double sqrt2

Initial value:

```
1.41421356237309504880168872420969807856967187537
```

Definition at line 93 of file utils.h.

5.117.3.15 const double sqrt2_2

Initial value:

```
0.70710678118654752440084436210484903928483593768
```

Definition at line 95 of file utils.h.

5.117.3.16 const double sqrt3

Initial value:

```
1.732050807568877293527446341505872366942805253
```

Definition at line 97 of file utils.h.

5.117.3.17 const double sqrt3_2

Initial value:

```
0.866025403784438646763723170752936183471402626
```

Definition at line 99 of file utils.h.

5.117.3.18 const double sqrt5

Initial value:

```
2.236067977499789696409173668731276235440618359
```

Definition at line 101 of file utils.h.

5.117.3.19 const double sqrt5_2

Initial value:

```
1.118033988749894848204586834365638117720309179
```

Definition at line 103 of file utils.h.