Custom Forces in OpenMM

Peter Eastman

OpenMM Workshop, Sept. 6, 2012
Forces in OpenMM

• “Standard” forces cover the most widely used force fields
  – HarmonicBondForce, HarmonicAngleForce, NonbondedForce, etc.

• But what if you need something that isn’t provided?
  – Could write a plugin, but that’s hard and a lot of work
Custom Forces

• Combine some of the flexibility of a plugin with most of the simplicity of standard forces
• You specify the energy function, it does the rest

CustomNonbondedForce("A*exp(-B*r)-C/r^6")
Custom Forces Classes

• CustomBondForce
  – Bonded force between two particles
  – Energy is a function of the distance

• CustomAngleForce
  – Bonded force between three particles
  – Energy is a function of the angle

• CustomTorsionForce
  – Bonded force between four particles
  – Energy is a function of the dihedral angle
Custom Forces Classes (continued)

- **CustomExternalForce**
  - Force applied to each particle independently
  - Energy is a function of the particle position

- **CustomNonbondedForce**
  - Nonbonded force between pairs of particles
  - Energy is a function of the distance
Custom Forces Classes (continued)

- **CustomGBForce**
  - Supports various implicit solvent models

- **CustomHbondForce**
  - Supports various hydrogen bonding models
Example: Harmonic Restraints

• Restrain particular atoms from moving

\[ E(x,y,z) = 10 \cdot \left( (x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 \right) \]

# Create the force
force = CustomExternalForce("10*((x-x0)^2+(y-y0)^2+(z-z0)^2)")
force.addPerParticleParameter("x0")
force.addPerParticleParameter("y0")
force.addPerParticleParameter("z0")

# Bind particle 5 to the location (0, 1, -0.5) nm
force.addParticle(5, (0, 1, -0.5)*nanometers)
Lennard-Jones Combining Rules

• Lennard-Jones potential represents Van der Waals interactions

\[ E(r) = 4\epsilon \left( \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^{6} \right) \]

• The combining rule determines \( \epsilon \) and \( \sigma \) from the particle parameters

• NonbondedForce uses Lorentz-Bertelot combining rule

\[ \epsilon = \sqrt{\epsilon_1 \epsilon_2} \quad \sigma = 0.5(\sigma_1 + \sigma_2) \]
Jorgensen Combining Rules

- Some force fields (e.g. OPLS) use Jorgensen combining rules

$$\varepsilon = \sqrt{\varepsilon_1 \varepsilon_2} \quad \sigma = \sqrt{\sigma_1 \sigma_2}$$

```python
force = CustomNonbondedForce(
    "4*eps*((sig/r)^12-(sig/r)^6); eps=sqrt(eps1*eps2);
    sig=sqrt(sig1*sig2)"
)
force.addPerParticleParameter("eps")
force.addPerParticleParameter("sig")
force.addParticle([0.1, 0.3])
```
Other Features

• Global parameters
  – Have a single value for all bonds/particles
  – Can change during a simulation

• Tabulated functions
  – Use tabulated values to define a function, then use it in expressions
  – Only supported by CustomNonbondedForce, CustomGBForce, and CustomHbondForce
Performance of Custom Forces

• Reference and CUDA platforms use an interpreter to evaluate expressions
  – Much slower than standard forces
  – Not a big deal for bonded forces, but very slow for nonbonded forces

• OpenCL platform generates a new kernel at runtime to evaluate the expression
  – Little or no performance difference from standard forces
Example: A Spherical Potential

```python
from simtk.openmm.app import *
from simtk.openmm import *
from simtk.unit import *

pdb = PDBFile('waterSphere.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology,
    nonbondedMethod=NoCutoff)
force = CustomExternalForce('10*max(0, r-1)^2; r=sqrt(x*x+y*y+z*z)')
for i in range(system.getNumParticles()):
    force.addParticle(i, ()
force = CustomExternalForce('10*max(0, r-1)^2; r=sqrt(x*x+y*y+z*z)')
for i in range(system.getNumParticles()):
    force.addParticle(i, ()
simulation = Simulation(pdb.topology, system, integrator) 
simulation.context.setPositions(pdb.positions)
simulation.reporters.append(PDBReporter('output.pdb', 100))
simulation.step(5000)
```
Example: A Spherical Potential

```python
from simtk.openmm.app import *
from simtk.openmm import *
from simtk.unit import *

pdb = PDBFile('waterSphere.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology,
    nonbondedMethod=NoCutoff)
force = CustomExternalForce('10*max(0, r-1)^2; r=sqrt(x*x+y*y+z*z)')
for i in range(system.getNumParticles()):
    force.addParticle(i, ())
system.addForce(force)
integrator = LangevinIntegrator(1000*kelvin, 1/picosecond,
    0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.reporters.append(PDBReporter('output.pdb', 100))
simulation.step(5000)
```
Exercises

• Run the script and view the results in VMD
• Increase the sphere radius to 2 nm. What happens?
• Reduce the temperature to 300K. What happens? Why?
Custom Integrators

• Define integration algorithm as an arbitrary series of computations
• Supports many types of integrators
  – Deterministic
  – Stochastic
  – Metropolized
  – Generalized Langevin
  – Multiple time step
  – ...

Example: Velocity Verlet

```python
integrator = CustomIntegrator(0.001)
integrator.addComputePerDof("v", "v+0.5*dt*f/m")
integrator.addComputePerDof("x", "x+dt*v")
integrator.addComputePerDof("v", "v+0.5*dt*f/m")
```

• CustomIntegrator automatically recalculates forces when necessary
• Supports arbitrary global and per-DOF variables