



Running Simulations with OpenMM

Peter Eastman

OpenMM Workshop, March 31, 2014



OpenMM is...

- A. An application for running molecular simulations
- B. A library of simulation routines for use by applications
- C. A domain specific language for molecular simulation
- D. All of the above

What is OpenMM?

- A toolkit for high performance molecular simulations
 - A low level computational library (C++)
 - A high level application layer (Python)
- Supports CPUs and GPUs (NVIDIA, AMD, Intel)

Running Simulations

- The “application layer” is really a set of Python libraries
- You write a Python script to run a simulation



*No programming
experience required!*

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Example Script

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

```
pdb = PDBFile('input.pdb')  
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')  
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,  
    nonbondedCutoff=1*nanometer, constraints=HBonds)  
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)  
simulation = Simulation(pdb.topology, system, integrator)  
simulation.context.setPositions(pdb.positions)  
simulation.minimizeEnergy()  
simulation.reporters.append(PDBReporter('output.pdb', 1000))  
simulation.step(10000)
```

Tell Python about the OpenMM libraries we'll be using

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Load the PDB file

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Select the force field to use

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
                                nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Construct the system to simulate

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
                                nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Select the integration method and parameters

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Let's do a simulation!

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Set the initial atom positions

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Better run an energy minimization first

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Save a frame to a PDB file every 1000 steps

Example Script

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Simulate!

Running The Script

> python simulatePdb.py

OR

- Copy it into an IPython notebook

Exercises

- Increase the temperature to 400K
 - Can you see the difference in the output?
- Switch to the SPC/E water model

Constant Pressure

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

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sb.xml', 'tip3p.xml')
system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
    nonbondedCutoff=1*nanometer, constraints=HBonds)
system.addForce(MonteCarloBarostat(1*bar, 300*kelvin))
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(pdb.topology, system, integrator)
simulation.context.setPositions(pdb.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

Starting from AMBER Files

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

prmtop = AmberPrmtopFile('input.prmtop')
inpcrd = AmberInpcrdFile('input.inpcrd')
system = prmtop.createSystem(nonbondedMethod=PME, nonbondedCutoff=1*nanometer,
    constraints=HBonds)
integrator = LangevinIntegrator(300*kelvin, 1/picosecond, 0.002*picoseconds)
simulation = Simulation(prmtop.topology, system, integrator)
simulation.context.setPositions(inpcrd.positions)
simulation.minimizeEnergy()
simulation.reporters.append(PDBReporter('output.pdb', 1000))
simulation.step(10000)
```

http://builder.openmm.org

OpenMM Script Builder [Get Help](#)

General System Integrator Simulation

Input coordinates

Input topology

Forcefield

Water Model

Platform

Precision

Device index

OpenCL platform index

```
#####
# this script was generated by openmm-builder. to customize it further,
# you can save the file to disk and edit it with your favorite editor.
#####

from __future__ import print_function
from simtk.openmm.app import *
from simtk.openmm import *
from simtk.unit import *
from sys import stdout

pdb = PDBFile('input.pdb')
forcefield = ForceField('amber99sbildn.xml', 'tip3p.xml')

system = forcefield.createSystem(pdb.topology, nonbondedMethod=PME,
                                nonbondedCutoff=1.0*nanometers, constraints=HBonds, rigidWater=True,
                                ewaldErrorTolerance=0.0005)
integrator = LangevinIntegrator(300*kelvin, 1.0/picoseconds, 2.0*femtoseconds)
integrator.setConstraintTolerance(0.00001)

platform = Platform.getPlatformByName('CUDA')
properties = {'CudaPrecision': 'mixed'}
simulation = Simulation(pdb.topology, system, integrator, platform, properties)
simulation.context.setPositions(pdb.positions)

print('Minimizing...')
simulation.minimizeEnergy()

simulation.context.setVelocitiesToTemperature(300*kelvin)
print('Equilibrating...')
simulation.step(100)

simulation.reporters.append(DCDReporter('output.dcd', 1000))
simulation.reporters.append(StateDataReporter(stdout, 1000, step=True,
                                              potentialEnergy=True, temperature=True))

print('Running Production...')
simulation.step(1000)
print('Done!')
```

