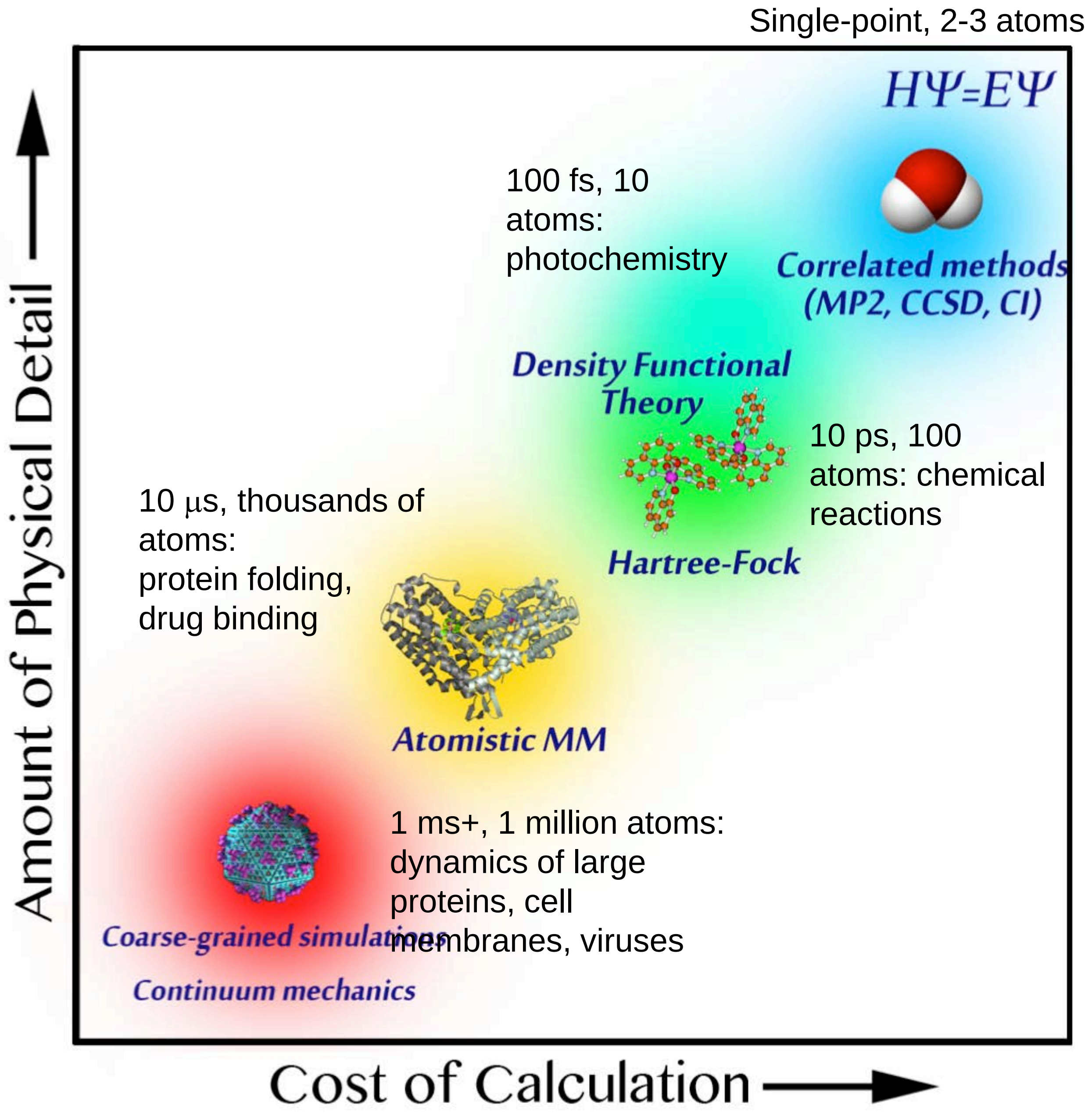
Creating and customizing force fields in OpenMM

Lee-Ping Wang
Stanford Department of Chemistry

OpenMM Workshop, Stanford University

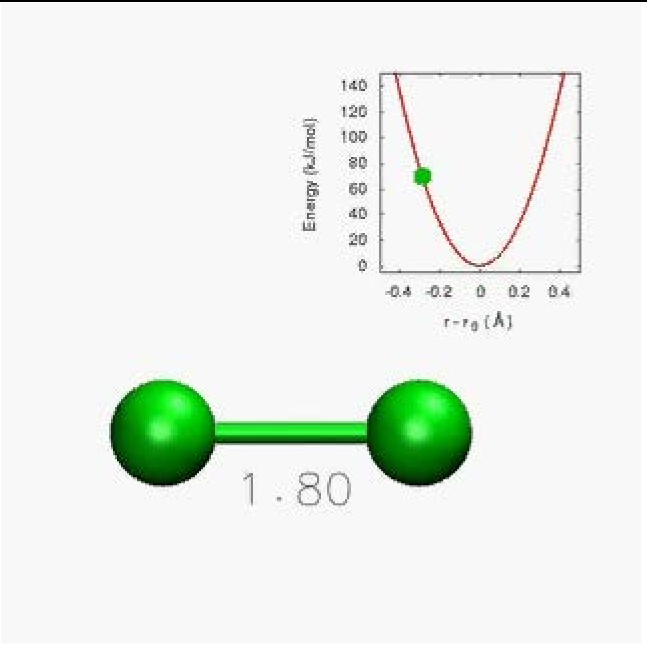
March 2013

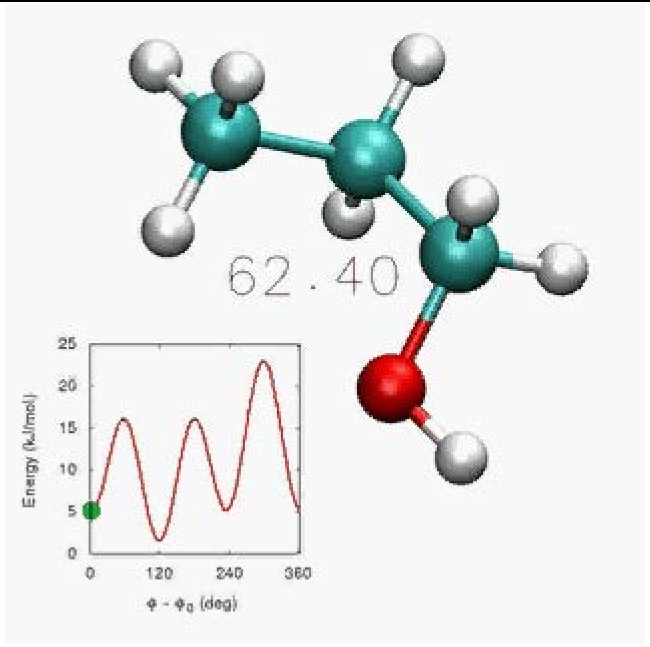
Introduction: A wide range of simulation domains

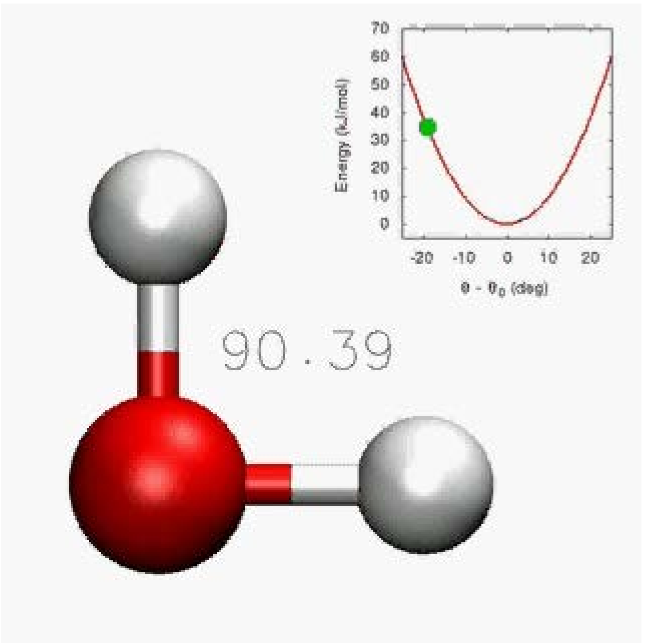

Single-point, 2-3 atoms
$H\Psi = E\Psi$
100 fs, 10 atoms: photochemistry
Correlated methods (MP2, CCSD, CI)
Density Functional Theory
10 ps, 100 atoms: chemical reactions
Hartree-Fock
10 μs, thousands of atoms: protein folding, drug binding
Atomistic MM
1 ms+, 1 million atoms: dynamics of large proteins, cell membranes, viruses
Coarse-grained simulations
Continuum mechanics
Amount of Physical Detail
Cost of Calculation

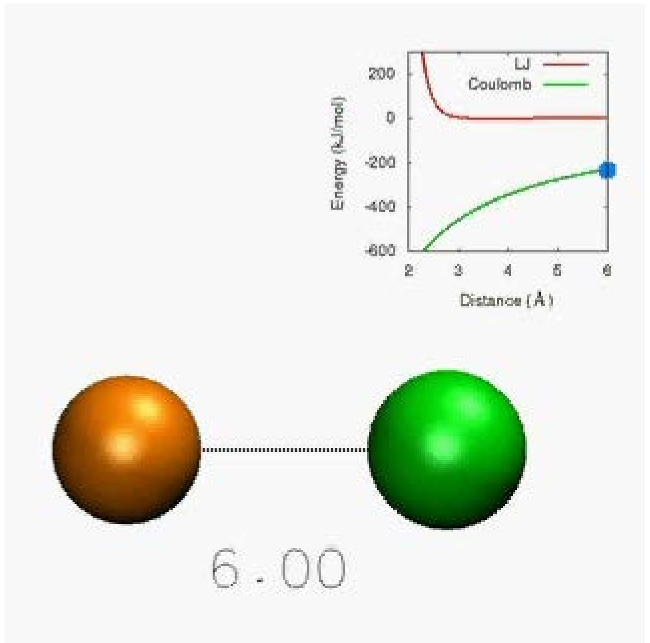
- Computer simulations of atoms and molecules span a vast range of detail
- More detailed theories can describe complex phenomena and offer higher accuracy
- Less detailed theories allow for simulation of larger systems / longer timescales
- In molecular mechanics simulation, the potential energy of molecules is

Introduction: Force Fields









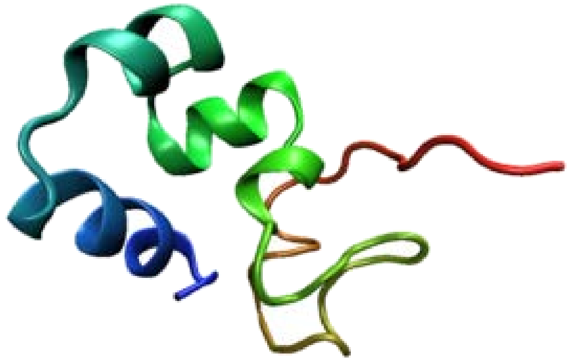
- **Force fields** are built from *functional forms* and empirical *parameters*
- Interactions include bonded pairwise, 3-body, and 4-body interactions…
- … as well as non-bonded pairwise interactions
- Simulation accuracy depends critically on choice of parameters

Creating a Force Field

Your project may require you to build a force field or obtain parameters from the literature.

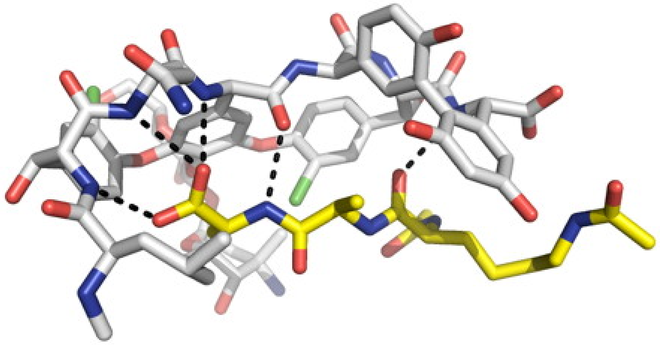
Common:
Biomolecules (e.g. villin)

Ships with OpenMM and most MD codes



Uncommon:
Organic molecules (e.g. vancomycin)

General force field procedures (GAFF, CGenFF) available



Rare:
Inorganic molecules (e.g. Photosystem II OEC)

Consult the literature

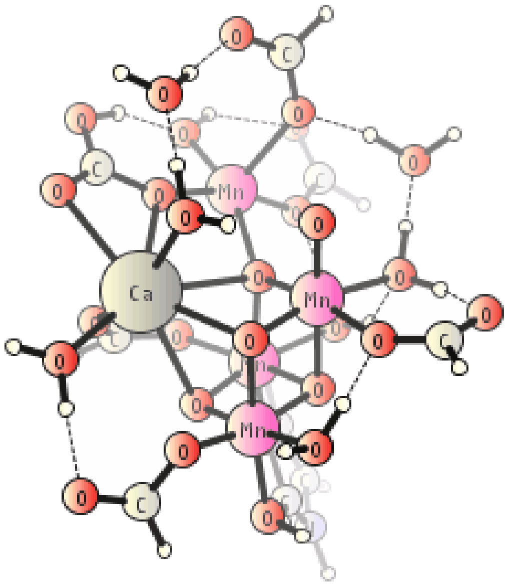


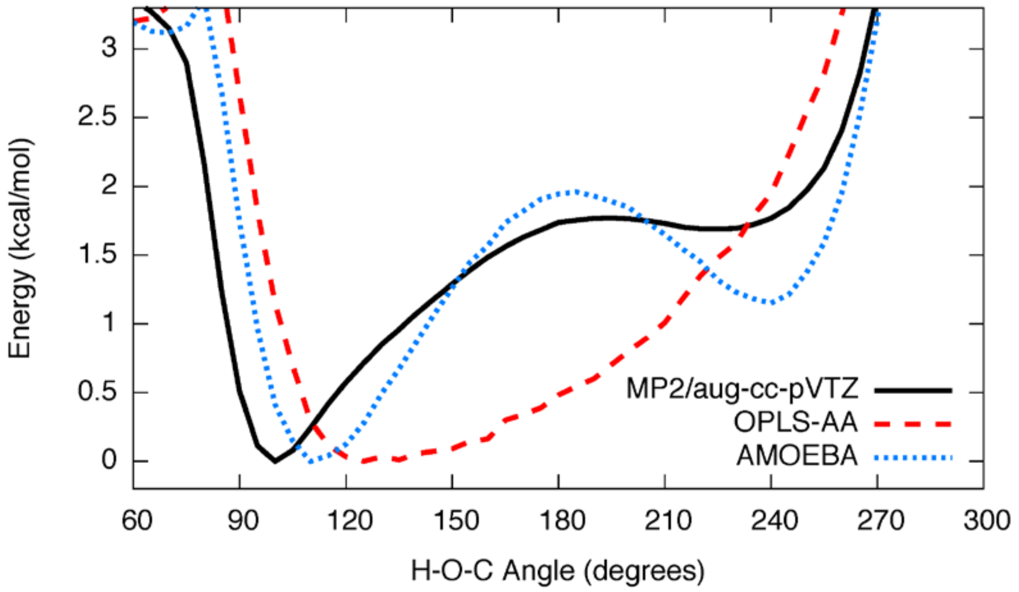
Image sources:
http://en.wikipedia.org/wiki/Villin
http://www.georgian.edu/chemistry/
http://qc.physto.se/~ps/biochem.html

Overview

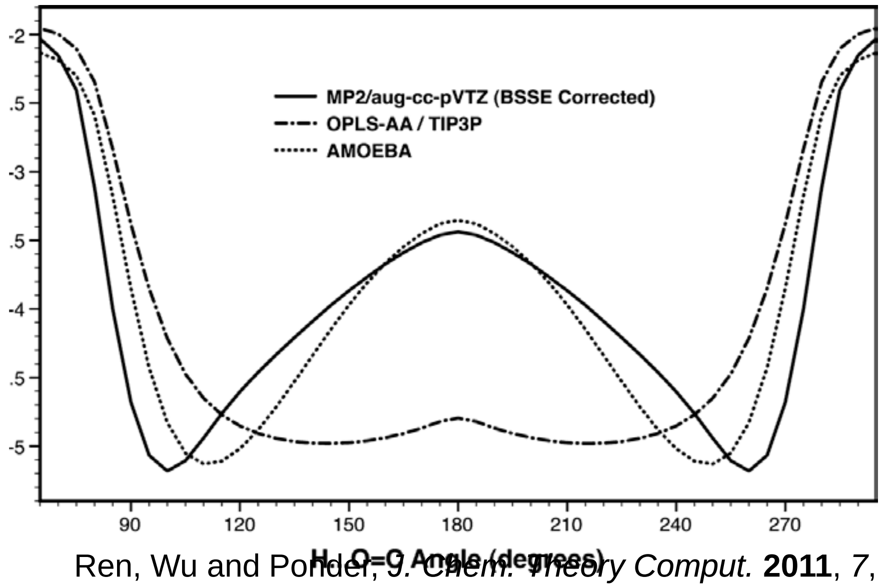
We will build a force field for formaldehyde and reproduce a literature result.

Our graph matches the left side of the literature plot and also reveals a previously hidden asymmetry

OpenMM result

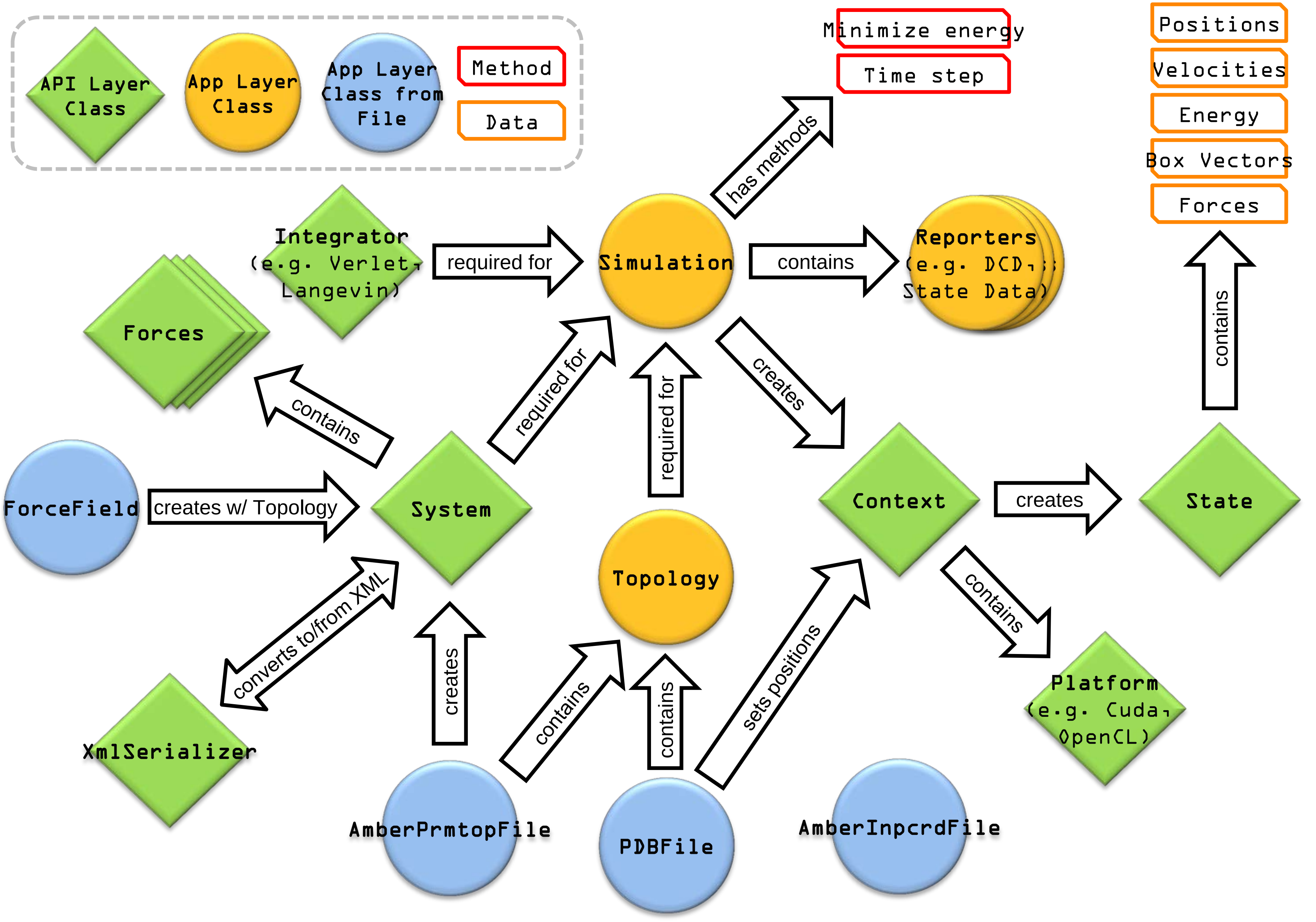


Original publication



Ren, Wu and Ponder, *J. Chem. Theory Comput.* **2011**, *7*, 3143.

Diagram of classes in OpenMM 5.0



The XML force field format

```
<ForceField>
<Residues>
  <Residue name="FML">
   <Atom name="C" type="fml-C"/>
   <Atom name="O" type="fml-O"/>
   <Atom name="H1" type="fml-H"/>
   <Atom name="H2" type="fml-H"/>
   <Bond from="0" to="1"/>
   <Bond from="0" to="2"/>
   <Bond from="0" to="3"/>
  </Residue>
 </Residues>
 <AtomTypes>
  <Type name="fml-C" class="C" element="C" mass="12.0"/>
  <Type name="fml-O" class="O" element="O" mass="16.0"/>
  <Type name="fml-H" class="H" element="H" mass="1.0"/>
 </AtomTypes>
 <NonbondedForce coulomb14scale="0.833333" lj14scale="0.5">
  <Atom type="fml-C" charge="0.450" sigma="0.375" epsilon="0.439"/>
  <Atom type="fml-O" charge="-0.450" sigma="0.296" epsilon="0.878"/>
  <Atom type="fml-H" charge="0.000" sigma="0.242" epsilon="0.063"/>
 </NonbondedForce>
 <HarmonicBondForce>
  <Bond class1="C" class2="O" length="0.12290" k="476976.0"/>
  <Bond class1="C" class2="H" length="0.10900" k="284512.0"/>
 </HarmonicBondForce>
 <HarmonicAngleForce>
  <Angle class1="H" class2="C" class3="O" angle="2.0943985" k="265.73"/>
  <Angle class1="H" class2="C" class3="H" angle="2.0943985" k="265.73"/>
 </HarmonicAngleForce>
</ForceField>
```

Contents of a force field XML file:

A **Residue** section provides a *residue template*, consisting of an ordered list of **atoms** and a list of **bonds**.

Each atom has a **name** and a **type**.

In this file, the **residue name** and **atom names** are unused.

Atom types are grouped into **atom classes** and **elements**.

Elements are used to recognize molecules from the PDB file.

Atom types and **atom classes** specify physical interactions.

See Chapter 7 in User's Guide for more details.

The PDB format

The PDB contains a list of atoms and bonds called the Topology – this is needed to make the System.

```
HETATM     1  C    FML      0        1.057  -0.271   0.000  1.00  0.00           C
HETATM     2  O    FML      0        2.177   0.200   0.000  1.00  0.00           O
HETATM     3  H1   FML      0        0.156   0.355   0.000  1.00  0.00           H
HETATM     4  H2   FML      0        0.893  -1.360   0.000  1.00  0.00           H
HETATM     5  HW1  HOH      1       -2.871   0.841   0.000  1.00  0.00           H
HETATM     6  OW   HOH      1       -2.331   0.044   0.000  1.00  0.00           O
HETATM     7  HW2  HOH      1       -2.972  -0.673   0.000  1.00  0.00           H
CONECT     1    2    3    4
CONECT     2    1
CONECT     3    1
CONECT     4    1
```

How OpenMM reads the PDB file:

Each residue should have a distinct **name** and/or **number**.

The **CONECT** records specify which atoms are bonded.

Residues are matched to *residue templates* in the force field using only the **elements** and the **bonds** between them.

The **atom names** and **residue names** don't need to match the force field, but it's preferable that they do (for clarity.)

However, the **atom names** and **residue names** *are* used to look up standard residues in the OpenMM internal databases.

See Chapter 7 in User's Guide for more details.

Building a topology

The PDB contains a list of atoms and bonds called the Topology – this is needed to make the System.

```
> MyPDB = PDBFile('input.pdb')          # Create a PDB object.
> Topo = MyPDB.topology                 # Assign variable name to topology.
> Atoms = list(Topo.atoms())            # Create a list of atom objects.
> Bonds = list(Topo.bonds())            # Create a list of bonded atom pairs.
> for A in Atoms:                       # Loop through the atoms.
      print A.name,                     # Print the name of the atom.
C O H1 H2 H1 O H2

> for B in Bonds:                       # Loop through the bonded atom pairs.
      print B[0].name, B[1].name        # Print the names of atoms in each bond.
H1 O
H2 O
C O
C H1
C H2

> FF = ForceField('fml.xml', 'tip3p.xml')     # Read the force field XML files.
> System = FF.createSystem(Topo)              # Create the system using ForceField
                                              # and Topology objects.
```

Storing a system object on disk

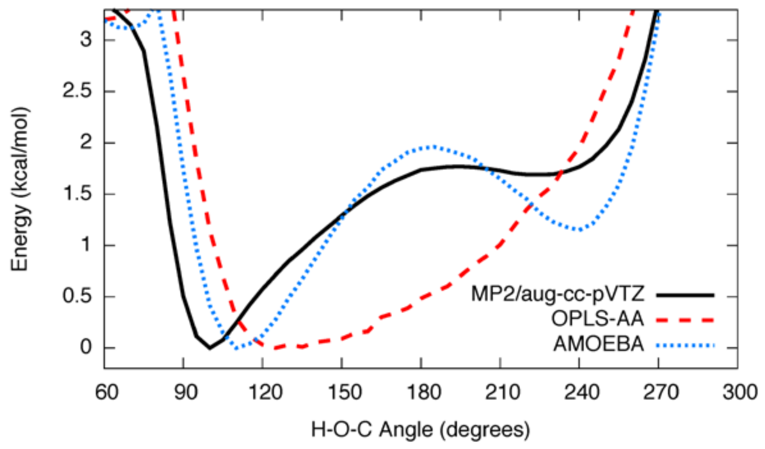
The XmlSerializer saves System objects to disk.

```
> Serial = XmlSerializer.serializeSystem(System)     # Convert System to XML text.
> print Serial                                       # Print the XML text to terminal.
<?xml version="1.0" ?>
<System type="System" version="1">                   # This is a System XML file
        <PeriodicBoxVectors>                         # containing a complete
                <A x="2" y="0" z="0" />              # specification of the System.
                <B x="0" y="2" z="0" />
                <C x="0" y="0" z="2" />              # It is comparable to the GROMACS
        </PeriodicBoxVectors>                        # .tpr or AMBER .prmtop formats.
        <Particles>
                <Particle mass="12" />
                <Particle mass="16" />
...

> XmlOut = open('opls-sys','w')                      # Open file for writing.
> print >> XMLOut, Serial                            # Write XML text to file.
> XMLOut.close()                                     # Close file.

# Once you have written the XML file, it is very easy to load.
> Serial2 = open('opls-sys.xml').read()
# Deserialize the XML text to create a System object.
> System2 = XmlSerializer.deserializeSystem(Serial2)
```

Exercise

Geometries

Energy Profile



• *Scenario*: You are asked to work on a project that was started by a senior grad student – but he or she has graduated and is not responding to email!

1) The Residues section of the force field XML file is incomplete. Repair it such that it correctly contains a residue template for the formaldehyde molecule.

2) Execute the `EnergyScan.py` script and write the results to a file, e.g., `python EnergyScan.py >> results.txt`. Plot the resulting OPLS-AA energy profile (red curve)

3) Modify the force field parameters (charge, sigma, epsilon); can you