ForceBalance Tutorial: Creating Accurate Force Fields based on Experimental Measurements and *ab initio* Calculations

Author: Lee-Ping Wang

Original version written March 24, 2013

Table of Contents:

I. How to use this document

Welcome! This document will guide you through how to run a ForceBalance calculation. ForceBalance is free software that allows you to create accurate force fields for molecular mechanics simulation.

```
Instructions for you to type into the command line are colored blue.
```

Occasionally I will show you pieces of code, data, or input / output files. You will only need to modify something if it's shown in blue.

If you are attending the March 2013 OpenMM Workshop, hopefully ForceBalance and OpenMM will have been installed for you on either (1) the Amazon Machine Image, (2) the VirtualBox Virtual Machine, or (3) locally on your computer. Otherwise, if you need to install ForceBalance, OpenMM and related dependencies for running this tutorial, please see the Appendix for brief installation instructions.

At the time of writing this document, the newest version of ForceBalance is version 1.1, corresponding to SVN revision 278. I may ask you to update ForceBalance to take advantage of features and bug fixes. You may do so with the following commands:

```
cd ~/ForceBalance/Source_Code/forcebalance/ (path may not be exact)
svn update
sudo python setup.py install
```

Go into the exercise folder and update it as well:

```
cd ~/Workshop-Exercises/ForceBalance/
svn update
```

Let's go make some force fields!

II. Components of a ForceBalance calculation

This chapter walks through the components of a ForceBalance calculation, but the actual calculations won't start until the next chapter. Skip to chapter 3 if time is short; read this chapter at your leisure. In this exercise, we will be optimizing the AMBER force field for the dimethyl sulfide (DMS) molecule using the experimentally measured density and heat of vaporization, as well as high-level *ab initio* interaction energy calculations.

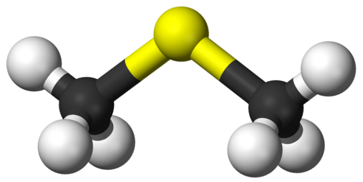


Figure 1. 3-D rendering of dimethyl sulfide, $(CH_3)_2S$ or DMS.

Dimethyl sulfide is a colorless liquid at standard conditions. It is commonly emitted by marine algae into the atmosphere and may have an important role in planetary homeostasis. Also, DMS is a good example for a force field optimization workshop exercise! This is because:

1) It is relatively simple (easy to understand for workshop purposes).
2) It is a liquid at standard conditions with experimentally measured density and heat of vaporization.
3) There are existing force fields that we can compare against.

Some results from existing force fields are provided at http://virtualchemistry.org/molecules/75-18-3/index.php and recapitulated here:

Property (298.15 K, 1 atm)	Experiment	GAFF Force Field	OPLS Force Field
Density (kg m^{-3})	841.9	801.9(3)	837.1(2)
Heat of vaporization (kJ mol^{-1})	27.65	23.88(5)	29.65(3)

Table 1. Some experimental properties of DMS calculated using two different force fields.

II.A. Force Field file

Our optimization will start from the GAFF (Generalized Amber Force Field) parameters. I have provided an OpenMM XML-format force field file containing the initial parameters. Let's view it:

```
cd ~/Workshop-Exercises/ForceBalance/targets
less forcefield/dms.xml
```

This file contains extra info that tells ForceBalance which parameters need to be optimized. You will see lines like this:

```
<Atom type="dms-ss-1" charge="-0.2812000000" sigma="0.3563594873"
epsilon="1.0460000000" parameterize="charge,sigma,epsilon"/>
```

The *parameterize* attribute gives ForceBalance the ability to create new XML files with modified values for *charge*, *sigma*, and *epsilon*. There is also an important attribute called *parameter_eval*:

```
<Atom type="dms-h1-7"
charge="0.0617000000" sigma="0.2471353044" epsilon="0.0656888000"
parameter_eval="sigma=PARM['Atom/sigma/dms-h1-3'],
epsilon=PARM['Atom/epsilon/dms-h1-3']
charge=-1.0*(1*PARM['Atom/charge/dms-ss-1']+2*PARM['Atom/charge/dms-c3-2'])/6,"/>
```

The *parameter_eval* attribute contains three formulas in Python code. The first formula tells ForceBalance to look up a parameter named *Atom/sigma/dms-h1-3* from a dictionary called *PARM*. This is because DMS has six hydrogen atoms that all share the same *sigma* parameter; ForceBalance only optimizes the first hydrogen *sigma* and the other five fields derive their values from the first.

The final formula looks more complicated; it sets the hydrogen charges such that the molecule maintains overall charge neutrality.

Finally, you will note that the attributes `parameterize` and `parameter_eval` only appear in elements under `NonbondedForce`. Even though the `parameterize` and `parameter_eval` attributes are easily added to any XML element with parameters, our example calculation only contains data that concerns the intermolecular (nonbonded) interactions. Now let's look at the data.

II.B. Reference data and the Targets directory

ForceBalance gives you the unique ability to optimize a force field using flexible combinations of experimental measurements and/or theoretical calculations as reference data.

Each separate data set is called a **target**. A target can be the binding energy of a cluster calculated using *ab initio* quantum chemistry, the heat capacity of a liquid, or any of a wide variety of molecular and condensed phase properties. ForceBalance is designed in such a way that new types of targets should be easy to implement. Let's look at the targets that I have set up:

```
cd ~/Workshop-Exercises/ForceBalance/targets; ls
```

You will see four directories: `dms-liquid, S2BPose, S2CPose, S2EPose`.

II.B.1 Condensed phase properties

`dms-liquid` contains condensed phase data for dimethyl sulfide and files needed for running these simulations. Let's go inside:

```
cd ~/Workshop-Exercises/ForceBalance/targets/dms-liquid
less data.csv (or download and view the file in Excel)
```

Most of the lines are comments, but the data is specified in the column headings and values on the final two lines. This file is highly customizable (read the documentation), but for this example it is very simple:

T	P	Rho	Hvap
298.15	1.0 atm	841.9	27.65

Table 2. Reference data for dimethyl sulfide in `targets/dms-liquid/data.csv`.

This means we will be fitting the force field to reproduce the density and heat of vaporization at 298.15 K, 1.0 atm.

This directory also contains `conf.pdb` and `mono.pdb`, which are PDB files for the condensed phase and the gas phase monomer. You may download and view them in VMD. Finally, `runcuda.sh` and `npt.py` are external scripts that are launched by ForceBalance during the optimization; you don't need to worry about them.

II.B.2 *Ab initio* interaction energies

```
cd ~/Workshop-Exercises/ForceBalance/targets
```

The three directories `S2BPose, S2CPose, S2EPose` each contain an interaction energy curve of dimethyl sulfide calculated using density functional theory (BSSE-corrected ωB97X-D/6-311+G*). The relative poses (S2B, S2C, S2E) are taken from Ref. 1; their names reflect the original authors' naming convention.

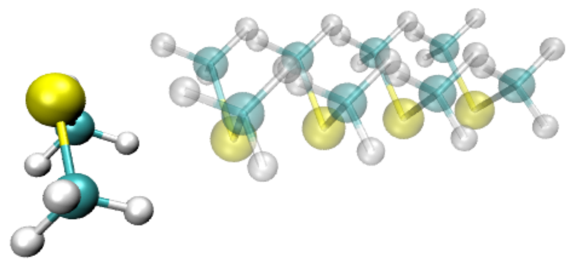
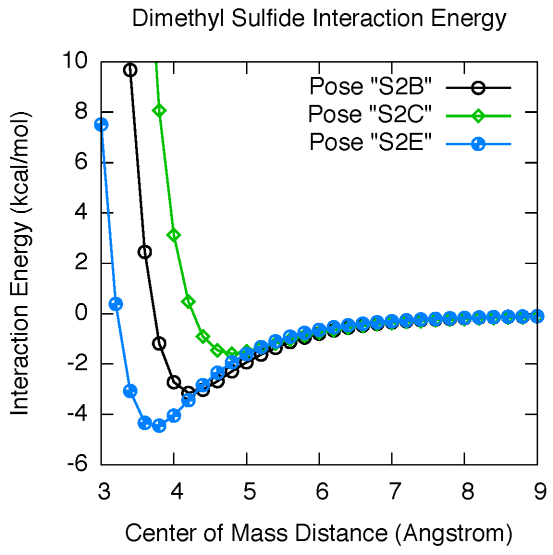


Figure 2. (Top) Several geometries used in the "S2E" interaction energy curve with different center-of-mass distances.

(Right) Interaction energy curves for three dimer poses calculated using DFT.



If you download and view the *S2EPose/all.pdb* file, you will see a number of dimer geometries with the second molecule moved progressively away.

The *S2EPose/qdata.txt* file contains the QM interaction energies in Hartree and a "label" which is convenient for plotting (here it's equal to the center-of-mass distance). To get a list of the energies which you can plot, run *./get-energies.sh*; this will allow you to generate the above interaction energy curves, using Excel or similar software.

II.C ForceBalance input file

The ForceBalance input file contains all of the settings for running a force field optimization.

```
cd ~/Workshop-Exercises/ForceBalance/
less optimize-simple.in
```

The file is organized using a "general options" section denoted using *$options*, and each target has a "target options" section denoted using *$target*. Sections are terminated using *$end*. Note that each target gets its own independent set of options. Options are generally case sensitive. If you ever need to know all of the available options in ForceBalance, run `MakeInputFile.py`. It will automatically generate an input file from the source code, putting the most important options at the top.

```
$options                                    # General options begin here
forcefield      dms.xml                     # Name of the force field file
jobtype         newton                      # Job type – Newton-Raphson optimization
$end                                        # End general options section

$target                                     # Target options begin here
name            dms-liquid                  # Name of this target
type            Liquid_OpenMM               # Target type; OpenMM liquid properties
$end                                        # End target section

$target                                     # Another target section
name            S2EPose                     # Name of this target
type            Interaction_OpenMM          # Target type; OpenMM interaction energies
fragment1       1-9                         # Specification of fragments for interaction
fragment2       10-18
$end
```

Table 3. Example of a minimal ForceBalance input file with the most important options shown – not part of exercise. Your actual input file – provided in the exercise directory – may have more options.

III. Running a ForceBalance calculation

It is easy to run a ForceBalance calculation once you have set things up. The main command `ForceBalance` takes the input file as its only argument.

III.A. Optimizing parameters to fit QM interaction energies

The following input file tells ForceBalance to only use the QM interaction energy target called *`S2EPose`*; it will be a fast calculation, taking only a minute or two.

```
cd ~/Workshop-Exercises/ForceBalance/
ForceBalance optimize-qm.in
```

You will notice that ForceBalance generates color output, which you can suppress by supplying an optional “-b” argument. The output will also be written to the terminal, and also to *`optimize-qm.out`*. If the output file already exists (e.g. if you run ForceBalance a second time), ForceBalance will quit (so that it doesn’t overwrite the output file), unless you supply an optional “-f” argument.

ForceBalance will now print lots of numbers. It will print out the calculated interaction energies along with the QM reference data, the difference, and the contribution to the objective function. You can plot the second vs. the third column in Excel to see the quality of fit.

```
#########################################################################
#|                          Target: S2EPose                             |#
#|     Interaction Energies (kcal/mol), Objective =  1.14598e+00        |#
#|   Label            Calc.       Ref.      Delta   Divisor        Term  |#
#########################################################################
3.0                11.428      7.508      3.920      5.594      0.49098
3.2                 2.029      0.381      1.648      5.000      0.10859
3.4                -1.530     -3.069      1.539      5.000      0.09475
3.6                -2.668     -4.340      1.672      5.000      0.11185
3.8                -2.813     -4.448      1.635      5.000      0.10695
4.0                -2.588     -4.050      1.463      5.000      0.08556
4.2                -2.254     -3.441      1.187      5.000      0.05634
4.4                -1.916     -2.843      0.926      5.000      0.03431
4.6                -1.614     -2.346      0.732      5.000      0.02143
4.8                -1.356     -1.935      0.579      5.000      0.01341
5.0                -1.141     -1.609      0.469      5.000      0.00878
5.2                -0.963     -1.332      0.368      5.000      0.00542
5.4                -0.818     -1.100      0.282      5.000      0.00318
5.6                -0.698     -0.914      0.216      5.000      0.00186
5.8                -0.600     -0.767      0.168      5.000      0.00113
6.0                -0.518     -0.646      0.128      5.000      0.00065
6.2                -0.450     -0.543      0.092      5.000      0.00034
6.4                -0.393     -0.460      0.066      5.000      0.00018
6.6                -0.345     -0.397      0.052      5.000      0.00011
6.8                -0.305     -0.345      0.040      5.000      0.00006
7.0                -0.271     -0.302      0.031      5.000      0.00004
7.2                -0.241     -0.264      0.023      5.000      0.00002
7.4                -0.216     -0.234      0.018      5.000      0.00001
7.6                -0.194     -0.208      0.013      5.000      0.00001
7.8                -0.175     -0.186      0.011      5.000      0.00000
8.0                -0.159     -0.167      0.008      5.000      0.00000
8.2                -0.144     -0.150      0.006      5.000      0.00000
8.4                -0.132     -0.135      0.004      5.000      0.00000
8.6                -0.120     -0.123      0.002      5.000      0.00000
8.8                -0.111     -0.112      0.001      5.000      0.00000
9.0                -0.102     -0.102      0.000      5.000      0.00000
-------------------------------------------------------------------------
```

Table 4. ForceBalance output for interaction energies. The second column contains the energies calculated using the initial force field parameters, and the third column is the QM data. The difference is in the third column, and a divisor is given in the fourth column. Note that the closest geometry is highly repulsive, and it is automatically given a larger divisor.

Next, it will print out the objective function and take a step in the parameter space (Table 4). If *`trust0`* is set to negative, it will now perform a line search, which is recommended when we're only fitting *ab initio* data.

From here on out, ForceBalance will optimize the force field parameters. If the objective function (or one of its components) decreases from the previous step, it will be printed in **green** whereas if it increases, it will be printed in **red**. If the total objective function goes up, the step will be rejected and retried with a smaller trust radius.

After several iterations, the convergence criteria will be met and ForceBalance will exit. It will print the final parameter values, and the final force field into the *`result`* directory.

```
######################################################################
#|                  Objective Function Breakdown                    |#
#|   Target Name              Residual  x  Weight  =  Contribution  |#
######################################################################
S2EPose                        1.14598      1.000      1.14598e+00
Regularization                 0.00000      1.000      0.00000e+00
Total                                                  1.14598e+00
----------------------------------------------------------------------
  Step       |k|        |dk|       |grad|       -=X2=-     Delta(X2)    StepQual
     0   0.000e+00   0.000e+00   5.267e+01   1.14598e+00   0.000e+00      0.000


##########################################################
#|                   Total Gradient                     |#
##########################################################
   0 [  2.03568812e+00 ] : Atom/charge/dms-ss-1
   1 [  8.90709493e-01 ] : Atom/epsilon/dms-ss-1
   2 [  3.07414212e+01 ] : Atom/sigma/dms-ss-1
   3 [  8.88115071e-01 ] : Atom/charge/dms-c3-2
   4 [  7.72859383e-01 ] : Atom/epsilon/dms-c3-2
   5 [  2.32955887e+01 ] : Atom/sigma/dms-c3-2
   6 [  1.80907412e+01 ] : Atom/epsilon/dms-h1-3
   7 [  3.08699483e+01 ] : Atom/sigma/dms-h1-3
----------------------------------------------------------

Searching! Hessian scaling = 1.0000e+00, L =  1.0000e+00, length 1.9825e-01, result -9.4480e-01
Searching! Hessian scaling = 1.0000e+01, L =  4.0000e+00, length 5.5225e-02, result -3.7797e-01
Searching! Hessian scaling = 2.4562e+01, L = -3.8541e+00, length 2.5114e-02, result -1.9919e-01
Searching! Hessian scaling = 1.0000e+00, L =  1.0000e+00, length 1.9825e-01, result -9.4480e-01
Searching! Hessian scaling = 4.4377e+00, L = -8.5410e-01, length 1.0641e-01, result -5.8340e-01
Searching! Hessian scaling = 2.3131e+00, L =  2.1459e+00, length 1.7413e-01, result -7.5498e-01
Searching! Hessian scaling = 1.0135e+00, L =  8.8384e-01, length 2.1458e-01, result -8.9202e-01
Searching! Hessian scaling = 1.1635e+00, L =  1.4044e+00, length 2.6987e-01, result -9.1604e-01
```

Table 5. ForceBalance output after the zeroth iteration. The objective function is shown, followed by the gradient (note that each index in the gradient vector contains a parameter, which has a name. This is followed by a line search, which will lead us to the next step.

```
  Step       |k|        |dk|       |grad|       -=X2=-     Delta(X2)    StepQual
     3   2.251e-01   2.294e-02   1.604e+00   1.48027e-01   5.319e-03      1.000

Convergence criterion reached for objective function (1.00e-02)

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@|            Final objective function value            |@
@|   Full:  1.480274e-01  Un-penalized:  9.734944e-02   |@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

##########################################################
#|            Final optimization parameters:            |#
#|            Paste to input file to restart            |#
##########################################################
read_mvals
   0 [ -1.1671e-01 ] : Atom/charge/dms-ss-1
   1 [  3.0678e-02 ] : Atom/epsilon/dms-ss-1
   2 [ -4.5314e-02 ] : Atom/sigma/dms-ss-1
   3 [ -4.9358e-02 ] : Atom/charge/dms-c3-2
   4 [  8.0598e-02 ] : Atom/epsilon/dms-c3-2
   5 [  2.4296e-02 ] : Atom/sigma/dms-c3-2
   6 [  1.3905e-01 ] : Atom/epsilon/dms-h1-3
   7 [ -7.2136e-02 ] : Atom/sigma/dms-h1-3
/read_mvals
```

Table 6. Final output from ForceBalance. The final optimization parameters may be pasted into a new input file to restart from where we left off.

Now that the optimization is done, it's time to look at our results. The results are the *final parameters* and the *quality of fit*. We will also perform some simple force field validation in the next section.

```
From forcefield/dms.xml
    <Atom type="dms-ss-1" charge="-0.2812000000" sigma="0.3563594873" epsilon="1.0460000000"
    <Atom type="dms-c3-2" charge="-0.0450000000" sigma="0.3399669508" epsilon="0.4577296000"
    <Atom type="dms-h1-3" charge="0.0617000000" sigma="0.2471353044" epsilon="0.0656888000"
From result/dms.xml
    <Atom type="dms-ss-1" charge="-3.140198779e-01" sigma="3.402114380e-01" epsilon="1.078089083e+00"
    <Atom type="dms-c3-2" charge="-5.887949762e-02" sigma="3.486249321e-01" epsilon="5.420348448e-01"
    <Atom type="dms-h1-3" charge="7.196314553e-02" sigma="2.214288865e-01" epsilon="2.111363601e-01"
```

Table 7. Snippets of before / after force field XML files, showing how the parameters have changed during the optimization.

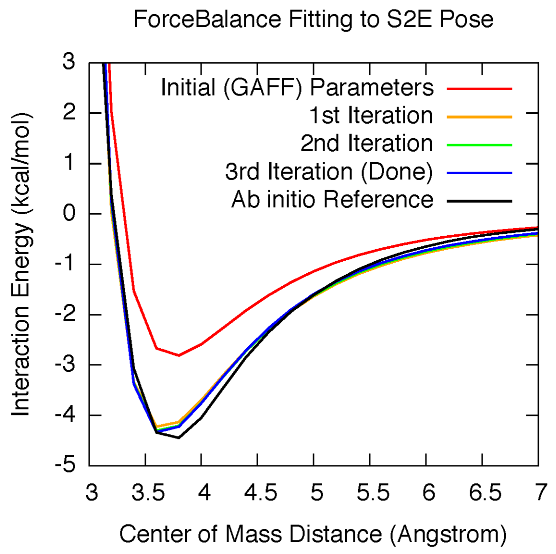


Figure 3. Quality of fit for interaction energy. Note how the shape of the interaction energy curve doesn't change very much after the first iteration.

What do you notice about the parameters? Which parameters changed the most, and are the final values physically reasonable? What impact might this have on the condensed phase properties?

You will see that the `epsilon` parameter for hydrogen more than doubled. More likely than not, this was due to over-fitting the well depth in the interaction energy. In the next section, we will evaluate the quality of this force field by looking at some condensed phase properties.

III.B. Single point calculation of condensed phase properties

In this section we will use the force field parameters from the previous section and calculate some condensed phase properties. First, let's create a new input file:

```
   cp optimize-qm.in single-expt_optimize-qm.in
   nano single-expt_optimize-qm.in
```

(change the following line)

```
   jobtype ~~newton~~single
```

(add the following lines to the end of the file)

```
   $target
   name dms-liquid
   type Liquid_OpenMM
   liquid_equ_steps 2000
   liquid_prod_steps 20000
   $end
```

We changed to a single-point evaluation of the objective function and added the *dms-liquid* target. Now we need to copy over the parameters from the last step.

Option 1: Copy the force field file from the *result* directory back over to the *forcefield* directory – but make sure you create a backup!

```
   cp forcefield/dms.xml forcefield/dms.xml.bak
   cp result/dms.xml forcefield/dms.xml
```

Option 2 (preferred): Paste the optimization variables from the end of the previous output file into the new input file. This saves us from juggling force field files around.

(paste from optimize-qm.out into single-expt_optimize-qm.in inside the $options section, make sure it's before $end):

```
read_mvals
   0 [ -1.1671e-01 ] : Atom/charge/dms-ss-1
   1 [  3.0678e-02 ] : Atom/epsilon/dms-ss-1
   2 [ -4.5314e-02 ] : Atom/sigma/dms-ss-1
   3 [ -4.9358e-02 ] : Atom/charge/dms-c3-2
   4 [  8.0598e-02 ] : Atom/epsilon/dms-c3-2
   5 [  2.4296e-02 ] : Atom/sigma/dms-c3-2
   6 [  1.3905e-01 ] : Atom/epsilon/dms-h1-3
   7 [ -7.2136e-02 ] : Atom/sigma/dms-h1-3
/read_mvals
```

Now let's run ForceBalance again:

```
ForceBalance single-expt_optimize-qm.in
```

Since this is a single-point calculation, it's not going to take any optimization steps. However, it will take a few minutes because ForceBalance needs to launch a liquid calculation in order to calculate the condensed phase properties.

```
#############################################################################################
#|                                    Density (kg m^-3)                                    |#
#|  Temperature  Pressure  Reference  Calculated +- Stdev     Delta    Weight    Term      |#
#############################################################################################
      298.15      1.0 atm    841.900      953.877 +- 35.303   111.977   1.00000  125.38916
---------------------------------------------------------------------------------------------

##########################################################
#|         Density objective function:  125.389         |#
##########################################################

#############################################################################################
#|                          Enthalpy of Vaporization (kJ mol^-1)                           |#
#|  Temperature  Pressure  Reference  Calculated +- Stdev     Delta    Weight    Term      |#
#############################################################################################
      298.15      1.0 atm     27.650       39.573 +- 0.898    11.923   1.00000   5.68618
---------------------------------------------------------------------------------------------

##########################################################
#|           H_vap objective function:  5.686           |#
##########################################################

######################################################################
#|                   Condensed Phase Properties:                    |#
#|   Property Name                Residual x Weight = Contribution  |#
######################################################################
Density                          125.38916    0.500     6.26946e+01
Enthalpy of Vaporization           5.68618    0.500     2.84309e+00
Total                                                   6.55377e+01
----------------------------------------------------------------------

######################################################################
#|                   Objective Function Breakdown                   |#
#|   Target Name              Residual  x  Weight  =  Contribution  |#
######################################################################
S2EPose                       0.09735      0.500      4.86756e-02
dms-liquid                   65.53767      0.500      3.27688e+01
Regularization                0.05068      1.000      5.06768e-02
Total                                                 3.28682e+01
----------------------------------------------------------------------
The objective function is: 32.8681876826
```

Table 8. Condensed phase properties evaluated using the QM-optimized force field.

The values highlighted in **red** are the condensed phase properties calculated using our QM-optimized force field. This should be alarming – the density and enthalpy of vaporization are far too high, giving worse results than GAFF (check Table 1)! This may be a surprising result, but there are many potential reasons for why force fields that are fitted to only QM calculations may perform badly. To recover from this, let's perform a *combined* optimization, which is the main distinguishing feature of ForceBalance.

III.C. Combined optimization of experimental data and *ab initio* calculations

Congratulations on making it this far! It's time to perform a combined optimization with multiple targets, which should give us a quality force field for dimethyl sulfide. We'll start from the original input file and disregard the QM-only parameters from Section III.A.

```
cp optimize-qm.in optimize-combine.in
```

```
nano optimize-combine.in
```

(With condensed phase properties, make trust0 a positive number!)

```
convergence_objective ~~0.01~~0.5
trust0 ~~-0.1~~0.1
```

(add the following lines to the end of the file)

```
$target
name dms-liquid
type Liquid_OpenMM
liquid_equ_steps 5000
liquid_prod_steps 50000
$end
```

(If you modified forcefield.xml, make sure to restore your backup.)

```
cp forcefield/dms.xml.bak forcefield/dms.xml
```

```
ForceBalance optimize-combine.in
```

This job will take longer than the previous job, because ForceBalance will now differentiate the simulated condensed phase property with respect to the force field parameter. The behavior will also be more erratic than the QM-only optimization, because the simulated properties contain some statistical noise. However, after two or three optimization steps, you should arrive at a result that performs better than GAFF (Table 9, see values highlighted in green).

Congratulations, you now know how to run ForceBalance! If you would like to explore more, read on to the next section.

```
#######################################################################
#|                            Target: S2EPose                         |#
#|     Interaction Energies (kcal/mol), Objective =  2.56323e-01     |#
#|   Label          Calc.      Ref.     Delta   Divisor        Term   |#
#######################################################################
3.0                 8.023      7.508      0.515      5.594     0.00846
3.2                 0.132      0.381     -0.249      5.000     0.00248
3.4                -2.645     -3.069      0.424      5.000     0.00719
3.6                -3.361     -4.340      0.979      5.000     0.03835
3.8                -3.270     -4.448      1.178      5.000     0.05548
4.0                -2.907     -4.050      1.143      5.000     0.05226
4.2                -2.490     -3.441      0.951      5.000     0.03618
4.4                -2.099     -2.843      0.744      5.000     0.02214
4.6                -1.760     -2.346      0.585      5.000     0.01370
4.8                -1.477     -1.935      0.457      5.000     0.00836
5.0                -1.244     -1.609      0.365      5.000     0.00532
5.2                -1.054     -1.332      0.278      5.000     0.00309
5.4                -0.898     -1.100      0.202      5.000     0.00164
5.6                -0.769     -0.914      0.145      5.000     0.00084
5.8                -0.664     -0.767      0.104      5.000     0.00043
6.0                -0.576     -0.646      0.070      5.000     0.00019
6.2                -0.503     -0.543      0.039      5.000     0.00006
6.4                -0.442     -0.460      0.018      5.000     0.00001
6.6                -0.390     -0.397      0.007      5.000     0.00000
6.8                -0.346     -0.345     -0.001      5.000     0.00000
7.0                -0.309     -0.302     -0.007      5.000     0.00000
7.2                -0.277     -0.264     -0.013      5.000     0.00001
7.4                -0.249     -0.234     -0.015      5.000     0.00001
7.6                -0.225     -0.208     -0.017      5.000     0.00001
7.8                -0.204     -0.186     -0.018      5.000     0.00001
8.0                -0.185     -0.167     -0.019      5.000     0.00001
8.2                -0.169     -0.150     -0.019      5.000     0.00001
8.4                -0.155     -0.135     -0.019      5.000     0.00001
8.6                -0.142     -0.123     -0.019      5.000     0.00001
8.8                -0.131     -0.112     -0.019      5.000     0.00001
9.0                -0.121     -0.102     -0.019      5.000     0.00001
-----------------------------------------------------------------------

############################################################################################
#|                                   Density (kg m^-3)                                    |#
#|  Temperature  Pressure  Reference  Calculated +- Stdev     Delta    Weight    Term     |#
############################################################################################
      298.15      1.0 atm   841.900      840.458 +- 5.051    -1.442   1.00000   0.02080
--------------------------------------------------------------------------------------------

############################################################################################
#|                         Enthalpy of Vaporization (kJ mol^-1)                           |#
#|  Temperature  Pressure  Reference  Calculated +- Stdev     Delta    Weight    Term     |#
############################################################################################
      298.15      1.0 atm    27.650       27.072 +- 0.742    -0.578   1.00000   0.01335
--------------------------------------------------------------------------------------------

######################################################################
#|                   Condensed Phase Properties:                    |#
#|   Property Name                Residual x Weight = Contribution  |#
######################################################################
Density                            0.02080    0.500    1.04008e-02
Enthalpy of Vaporization           0.01335    0.500    6.67449e-03
Total                                                  1.70753e-02
----------------------------------------------------------------------

#####################################################################################
#|                            Objective Function Breakdown                        |#
#|   Target Name              Residual  x  Weight  =  Contribution (Current-Prev)  |#
#####################################################################################
S2EPose                        0.25632      0.500      1.28162e-01 ( -7.662e-02 )
dms-liquid                     0.01708      0.500      8.53766e-03 ( +4.072e-03 )
Regularization                 0.02543      1.000      2.54335e-02 ( +1.604e-02 )
Total                                                  1.62133e-01 ( -5.651e-02 )
-------------------------------------------------------------------------------------
  Step       |k|        |dk|       |grad|       -=X2=-     Delta(X2)    StepQual
     2   1.595e-01   1.032e-01   1.653e+01   1.62133e-01   5.651e-02      0.866
```

Table 9. Simultaneous optimization of QM interaction energy and condensed phase properties.

III.D. Beyond this exercise

There are many other useful features of ForceBalance that you can explore. For example:

1) Try to fit all three interaction energy curves at the same time. Copy *`optimize-qm.in`* to *`optimize-qm3.in`*. Duplicate the `$target` section for `S2EPose` and change the target name to `S2BPose` and `S2CPose`. Run `ForceBalance optimize-qm3.in`. Does the force field reproduce some interaction energies better than others?
2) Try an L1-regularized optimization. Copy *`optimize-qm.in`* to *`optimize-qm-L1.in`*. Change `penalty_type L2` to `penalty_type L1`. Run `ForceBalance optimize-qm-L1.in`. This optimization will only modify the parameters that strongly affect the objective function; some of the parameters will stay exactly where they are. Compare the initial and final parameters; what do you see? Do the L1-regularized parameters perform any better for the condensed phase?
3) Increase the length of the condensed phase simulations. Copy *`optimize-combine.in`* to *`optimize-combine-100k.in`*. In the `$target` section for `dms-liquid`, change the value for `liquid_prod_steps` to `100000`. This job will take longer, because it's running a longer condensed phase simulation. Can you now converge the density to a more accurate value?

IV. Appendices: Installation and Setup

This is a chapter that explains how to set up ForceBalance and the exercise. OpenMM Workshop participants: you won't need to do any of this, since everything has already been installed for you.

IV.A. Installing ForceBalance software

To install ForceBalance for this tutorial, a few dependencies are required. First you should be running Linux, since ForceBalance hasn't been tested on other operating systems. Python 2.7 is required; Python 2.7.3 or newer is recommended.

Install OpenMM 5.0.1 (see https://simtk.org/home/openmm for instructions). Next install the Python packages `numpy`, `scipy`, `networkx` and `lxml`. The latter requires the `libxml2` and `libxslt` libraries. The `libxml2` library comes with its own Python bindings, which we *don't* want – install this library with the Python option disabled, then install `lxml` (which is an improved Python interface to `libxml2`). If you have root permission on your computer, it should be easy to install packages using the `rpm`, `yum` and `apt-get` utilities. Otherwise, you will need to build the packages from source, which is a bit more challenging, but still very doable.

Now install ForceBalance. Check out the source code from the SVN repository using `svn checkout https://simtk.org/svn/forcebalance`. Go into the forcebalance directory and run `python setup.py install`. That's it!

IV.B. Setting up the dimethyl sulfide calculation

Setting up a calculation is nontrivial and can take a few hours. This is because you need to gather and organize the reference data such that ForceBalance can recognize it, plus you need to create the force field file that identifies the parameters that need to be optimized. My workflow for setting up the calculation involves using different programs for various components (some of which are not free), but you can always swap in your favorite program for doing something. The Python scripts that I used depend on functions in the ForceBalance libraries; they are provided in the *`setup`* directory.

The full list of programs I used to set up the ForceBalance calculation is: AMBER 12, GROMACS 4.5.5, OpenMM 5.0.1, GaussView 4.1, Q-Chem 4.0, and ForceBalance 1.1.

Creating the force field file

Amber's modeling tools are very good, so I used it to create the GAFF force field for dimethyl sulfide. I started by building the dimethyl sulfide molecule in GaussView with Cartesian coordinates, and converted it by hand to the `.xyz` file format. OpenBabel was used to convert this to a `.mol2` file. The `antechamber` program (part of AMBER) was used to generate a new `.mol2` file with GAFF atom types and AM1-BCC charges. The `parmchk` program was used to make an AMBER `.frcmod` file containing all of the GAFF force field parameters for dimethyl sulfide.

Next I converted the `.mol2` and `.frcmod` files to OpenMM format (ForceBalance also has direct interfaces to AMBER, but after all we are in the OpenMM workshop.) I wrote a Python script `convert.py` to do this; this also automatically adds the correct parameterization attributes. Now the force field file has been made, along with a PDB file for the monomer.

Creating the PDB files for the condensed phase simulation

I created a periodic box of dimethyl sulfide using GROMACS. First I created GROMACS `.gro` and `.top` files from the AMBER `.mol2` and `.frcmod` files using the `acpype` program (this is a standalone program, not part of GROMACS or AMBER). I used the `genbox` program to create a periodic box with 216 molecules. `genbox` randomly inserts molecules into a periodic box, and you need to choose a box that's big enough (a fair bit larger than the experimentally correct box size). Then I used GROMACS to minimize and run NPT dynamics on the box until the density approached the GAFF equilibrium value. The final configuration was saved as `conf.pdb` for use in the exercise; then I added CRYST and CONECT records so that OpenMM would set up the simulation properly. The script `conect.py` was used to create the CONECT records (representing chemical bonds). The monomer PDB file was easy to create from the periodic box PDB file; I kept only the first molecule and its CONECT records, and I deleted the CRYST record.

Creating the reference data file for condensed phase simulation

The data for dimethyl sulfide was taken from http://virtualchemistry.org/molecules/75-18-3/index.php. The `data.csv` file was taken from a template located in the ForceBalance directory: *`forcebalance/studies/Settings/data.csv.template`*

Creating the interaction energy profiles

I took the Gaussian input file created by GaussView and converted it by hand to a Q-Chem input file. I optimized the geometry of the monomer using the ωB97X-D exchange-correlation function and the 6-311+G* basis set. I then loaded the optimized monomer back into GaussView, duplicated it and tried to arrange the two molecules in the poses outlined by Reference 1. I arranged the molecules into the "S2B", "S2C", and "S2E" poses (I chose S2E because it was lowest in energy and the other two arbitrarily), then minimized the dimer geometries in Q-Chem at the same level of theory.

I loaded the minimized dimer geometries and generated a series of geometries by varying the dimer center-of-mass distances from 3.0 to 9.0 Å in 0.2 Å steps. These were saved to all.pdb. I calculated the interaction energy curve by running a BSSE-corrected Q-Chem calculation for each geometry (i.e. dimer – monomer_A – monomer_B, all in the dimer basis.) This was a total of 93 calculations per interaction energy curve; the Work Queue software was used to distribute the calculations across multiple nodes to make things faster. Another Python script was used to parse the Q-Chem output files and generate qdata.txt for each interaction energy curve.

These files are all provided (in a somewhat disorganized fashion) in the `Setup.tar.bz2` archive file.

References

(1) Cabaleiro-Lago, E. M.; Hermida-Ramon, J. M.; Rodriguez-Otero, J. *J. Phys. Chem. A* **2004**, *108*, 4923.