MCSMRFF Documentation

Release 0.0.1

Henry Herbol and James Stevenson

Nov 16, 2016

Contents:

CHAPTER

ONE

MORE COMPLICATED SIMPLE REACTIVE FORCE FIELD (MCSMRFF)

A custom force field for the implementation of reactive, directional bonds in Molecular Dynamics. Force field encompasses Tersoff potentials, Coulombic potentials, and Lennard-Jones potentials. This codebase includes three main codes:

1. LAMMPs code adjustment for implementing MCSMRFF
2. Training set generator
3. Automatic parameterization method

CHAPTER

TWO

THE FORCE FIELD

The MCSMRFF force field works as follows:

1. All atoms within some cutoff are described by Lennard-Jones (LJ) and Coulomb potentials
2. Those atoms for which it is defined, will also have the Tersoff potential applied to it.
3. All remaining atomic interactions will be described by the Optimized Potential for Liquid Simulation (OPLS-AA) forcefield.

Using the *pair_style hybrid/overlay* command, we can specify the pair styles of all atoms to be a function of both Tersoff and a custom pair style: *lj/cut/coul/inout*. This is simply:

```
pair_style hybrid/overlay lj/cut/coul/inout alpha R_cut_inner R_cut_outer tersoff
```

Where α is a damping parameter (in inverse distance units), $R_{cut,inner}$ is an inner cutoff, and $R_{cut,outer}$ is an outer cutoff. The distinction between the two cutoffs is that it allows for a smooth transition between the pair style being completely "on" ($r < R_{cut,inner}$) and "off" ($r > R_{cut,outer}$). When specifying pair coefficients, use the following syntax:

```
pair_coeff i j lj/cut/coul/inout epsilon_ij sigma_ij R_cut_inner
```

Where $\epsilon_{ij} = \sqrt{\epsilon_i * \epsilon_j}$ and $\sigma_{ij} = \sqrt{\sigma_i * \sigma_j}$.

The Tersoff Potential (1, 2, 3) is well described on the LAMMPs website, but in short describes 3-body atomic interactions as follows:

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{i}\sum_{j\neq i} V_{ij} \\
V_{ij} &= f_C(r_{ij})\left[a_{ij} f_R(r_{ij}) + b_{ij} f_A(r_{ij})\right] \\
f_C(r) &= \begin{cases} 1 & , \quad r < R - D \\ \frac{1}{2} - \frac{1}{2}sin(\frac{\pi}{2}\frac{r-R}{D}) & , \quad R - D < r < R + D \\ 0 & , \quad r > R + D \end{cases} \\
f_R(r) &= Ae^{-\lambda_1 r} \\
f_A(r) &= -Be^{-\lambda_2 r} \\
a_{ij} &= (1 + \alpha^n \eta_{ij}^n)^{-\frac{1}{2n}} \\
\eta_{ij} &= \sum_{k\neq i,j} f_C(r_{ik}) e^{\lambda_4^m (r_{ij} - r_{ik})^m} \\
b_{ij} &= (1 + \beta^n \xi_{ij}^n)^{-\frac{1}{2n}} \\
\xi_{ij} &= \sum_{k\neq i,j} f_C(r_{ik}) g(\theta_{ijk}) e^{\lambda_3^m (r_{ij} - r_{ik})^m} \\
g(\theta) &= \gamma_{ijk}\left(1 + \frac{c^2}{d^2} - \frac{c^2}{[d^2 + (cos\theta_{ijk} - cos\theta_0)^2]}\right)
\end{aligned}
$$

Note, this is not the "pure" original implementaion by Tersoff, but instead a generalization to accomodate updates to the method. Fundamentally, the Tersoff potential works by describing potential as a function of $f_R(r)$, the repulsive term, and $f_A(r)$, the attractive term. A smoothing function, $f_C(r)$, is applied so as to maintatain a continuous transition for a cutoff distance. b_{ij} is included to account for the bond-order, and is assumed to be a monotonically decreasing function of the coordination of atoms *i* and *j*. That is, $b_{ij} \propto z^{-\frac{1}{2n}}$, where *z* describes the coordination number. b_{ij} is also a function of ξ_{ij}, which counts the number of bonds to atom *i* (excluding *i-j*). a_{ij} is poorly described in the original implementation, but seems to describe a correction term for systems in which η_{ij} is exponentially large. This occurs for atoms outside of the first-neighbor shell. That is to say that the repulsive term $f_R(r)$ decreases for atoms further away. Interestingly enough, this implementation of the Tersoff potential is not used in LAMMPs, and further not used in example implementations by Tersoff. Thus, it is typical for $a_{ij} = 1$. Finally, the function $g(\theta)$ describes the angular contribution to the bonding terms.

To accomodate variations of this potential, *m* and γ_{ijk} are used. γ_{ijk} scales the angular contribution to b_{ij}, and *m* changes from 3 to 1 depending on whether the Nord et al. variation is to be used.

Looking through these equations, the following are all terms that we must parameterize (note, we follow the trend of setting $a_{ij} = 1$):

R - The cutoff distance at which 50% of the potential is in effect.

D - The buffer distance before and after *R* for the smoothing function $f_C(r_{ij})$.

A - The constant scalar for the repulsive function $f_R(r_{ij})$.

λ_1 - The exponential decay of the repulsive function $f_R(r_{ij})$.

B - The constant scalar for the repulsive function $f_A(r_{ij})$.

λ_2 - The exponential decay of the attractive function $f_A(r_{ij})$.

n - An exponential to decay factor of b_{ij} to accomodate for the change in decay rate of low and high coordinations ξ_{ij}.

beta - A scalar for the impact of the coordination contribution ξ_{ij}.

m - An exponential to accomodate the Tersoff potential (3) or the Nord variation (1).

λ_3 - The exponential factor of the bond coordination term ξ_{ij}.

γ_{ijk} - Scaling term for the impact of the angular contribution to the bond coordination.

c - How strong the angular contribution is to the bond coordination.

d - How sharp the angular contribution is to the bond coordination.

$cos\theta_0$ - The minimum angle for which this contribution is minimized.

CHAPTER

THREE

INSTALLATION

1. Download LAMMPS (older versions here). Some versions will not work - the 7 Dec 2015 is the one we've tested the most, so use that one.
2. Open a terminal in the src directory in your lammps folder
3. In your LAMMPS src directory run *make yes-manybody*
4. If you don't have an SSH key, generate one like this (if you do, skip to step 8):

   ```
   ssh-keygen -t rsa -b 4096 -C "your_email@example.com"  #Creates an ssh key, using␣
   ↪your GitHub e-mail as a label
   ```

   When prompted to "Enter a file in which to save the key," press Enter

   If asked to Overwrite, enter 'y'

   At the prompt, type a secure passphrase

   Retype your secure passphrase
5. Add the new SSH key to your GitHub account

   ```
   gedit ~/.ssh/id_rsa.pub
   ```

   In the top right corner of any GitHub page in your browser, click on your profile photo, then click 'Settings'

   In the user settings sidebar, click 'SSH keys'

   Click 'New SSH key'

   In the "Title" field, add a descriptive label for the new key

   Copy and Paste the contents from the 'id_rsa.pub' file into the "Key" field

   Click 'Add SSH key'
6. Load your keys into your SSH agent

   ```
   eval "$(ssh-agent -s)"
   ssh-add
   ```

   Enter passphrase
7. Test your SSH connection

   ```
   ssh -T git@github.com
   ```

   You should see the message "Hi 'username'! You've successfully authenicated, but GitHub does not provide shell access."

8. Clone repository wherever

```
git clone git@github.com:hherbol/Grad-MCSMRFF.git
```

9. Copy over contents of Grad-MCSMRFF/LAMMPS to your lammps src folder. Note, this overwrites the min.h and pair_tersoff.h files.

10. Make using the new Makefile.mcsmrff

```
make mcsmrff -j 4
```

11. Add the path of Grad-MCSMRFF/MCSMRFF/pys to your PYTHONPATH variable:

```
echo 'export PYTHONPATH=/path/to/Grad-MCSMRFF/MCSMRFF/pys:$PYTHONPATH' >> ~/.zshrc
```

12. You're done! Now you can use lammps with the lmp_mcsmrff file in your lammps/src directory.

CHAPTER FOUR

GENERATING A TRAINING SET

To parameterize a Force Field, first the training set must be generated. MCSMRFF comes with a built in Training Set Generator to assist in generating a robust set. To use it, you must first have a set of "seed" molecules. These should be in the Chemical Markup Language (CML) file format, such as the following example of MAPbCl3:

```
<molecule>
  <atomArray>
    <atom id="a1" elementType="C" label="234" x3="2.856348" y3="3.465525" z3="2.882101
↪"/>
    <atom id="a2" elementType="N" label="230" x3="2.065567" y3="2.498566" z3="2.094128
↪"/>
    <atom id="a3" elementType="H" label="85" x3="3.535110" y3="2.922596" z3="3.560587
↪"/>
    <atom id="a4" elementType="H" label="85" x3="2.178424" y3="4.101417" z3="3.475260
↪"/>
    <atom id="a5" elementType="H" label="85" x3="3.449516" y3="4.098951" z3="2.201873
↪"/>
    <atom id="a6" elementType="H" label="233" x3="1.464495" y3="1.844420" z3="2.661628
↪"/>
    <atom id="a7" elementType="H" label="233" x3="2.631700" y3="1.844694" z3="1.490830
↪"/>
    <atom id="a8" elementType="H" label="233" x3="1.384889" y3="2.928154" z3="1.415175
↪"/>
    <atom id="a9" elementType="Pb" label="356" x3="0.134939" y3="0.094135" z3="0.
↪152826"/>
    <atom id="a10" elementType="Cl" label="344" x3="2.767340" y3="0.232853" z3="0.
↪077164"/>
    <atom id="a11" elementType="Cl" label="344" x3="-0.138816" y3="2.703617" z3="-0.
↪100291"/>
    <atom id="a12" elementType="Cl" label="344" x3="0.070357" y3="0.210233" z3="2.
↪786018"/>
  </atomArray>
  <bondArray>
    <bond atomRefs2="a8 a2" order="1"/>
    <bond atomRefs2="a7 a2" order="1"/>
    <bond atomRefs2="a2 a6" order="1"/>
    <bond atomRefs2="a2 a1" order="1"/>
    <bond atomRefs2="a5 a1" order="1"/>
    <bond atomRefs2="a1 a4" order="1"/>
    <bond atomRefs2="a1 a3" order="1"/>
  </bondArray>
</molecule>
```

Note, however, that a more robust way to define this molecule would be to split apart the two constituents (MA and PbCl3) into different molecules. This is taken advantage of in the training set generator to gain some more conformations. An example of how to accomplish this in CML is as follows:

```
<molecule>
  <molecule>
    <atomArray>
      <atom id="a1" elementType="C" label="234" x3="2.856348" y3="3.465525" z3="2.
↪882101"/>
      <atom id="a2" elementType="N" label="230" x3="2.065567" y3="2.498566" z3="2.
↪094128"/>
      <atom id="a3" elementType="H" label="85" x3="3.535110" y3="2.922596" z3="3.
↪560587"/>
      <atom id="a4" elementType="H" label="85" x3="2.178424" y3="4.101417" z3="3.
↪475260"/>
      <atom id="a5" elementType="H" label="85" x3="3.449516" y3="4.098951" z3="2.
↪201873"/>
      <atom id="a6" elementType="H" label="233" x3="1.464495" y3="1.844420" z3="2.
↪661628"/>
      <atom id="a7" elementType="H" label="233" x3="2.631700" y3="1.844694" z3="1.
↪490830"/>
      <atom id="a8" elementType="H" label="233" x3="1.384889" y3="2.928154" z3="1.
↪415175"/>
    </atomArray>
    <bondArray>
      <bond atomRefs2="a8 a2" order="1"/>
      <bond atomRefs2="a7 a2" order="1"/>
      <bond atomRefs2="a2 a6" order="1"/>
      <bond atomRefs2="a2 a1" order="1"/>
      <bond atomRefs2="a5 a1" order="1"/>
      <bond atomRefs2="a1 a4" order="1"/>
      <bond atomRefs2="a1 a3" order="1"/>
    </bondArray>
  </molecule>
  <molecule>
    <atomArray>
      <atom id="a1" elementType="Pb" label="356" x3="0.134939" y3="0.094135" z3="0.
↪152826"/>
      <atom id="a2" elementType="Cl" label="344" x3="2.767340" y3="0.232853" z3="0.
↪077164"/>
      <atom id="a3" elementType="Cl" label="344" x3="-0.138816" y3="2.703617" z3="-0.
↪100291"/>
      <atom id="a4" elementType="Cl" label="344" x3="0.070357" y3="0.210233" z3="2.
↪786018"/>
    </atomArray>
  </molecule>
</molecule>
```

Now, to generate your training set you need only be in an empty folder, with the sub-folder “seed”. Afterwards, the following can be used to begin automatic training set generation:

```
import mcsmrff_train

mcsmrff_train.create_training_set("set1",
                                  N_perterbations_per_seed=5,
                                  perterbation_dx=0.3,
                                  perterbation_dr=10,
                                  expansion_perterbation_dx=0.1,
                                  expansion_perterbation_dr=0.5,
                                  expansion_step=0.5,
                                  N_expansion=5,
                                  N_perterbations_per_expansion=5,
                                  min_seeds=True,
                                  training_sets_folder="training_set",
                                  queue=None,
                                  procs=1,
                                  persist=False)
```

This will then generate a training set in the subfolder "set1". This set will include the following:

1. An optimized seed for each seed in your seed folder
2. N perterbations per seed in your seed folder (includes optimized seeds). These perterbations are restricted to translations of *perterbation_dx* and general rotation angles of *perterbation_dr*.
3. N expansions, for which systems that have more than 1 molecule. In this case, every pair-wise molecule subsystem is "expanded" in steps of *expansion_step*. For each of these expansions, there are M more perterbations.

Thus, for the case of N seed molecules, with *min_seeds* set to true, we have the following number of training set molecules:

$N_{train\,set} = N * 2 * N_{perterbation} * M * N_{expansion} * N_{perterbation,expansion}$

Where *M* describes the total number of pairs within the seed folder.

CHAPTER

FIVE

PARAMETERIZING FOR MCSMRFF

1. Build example structures for your system in Avogadro (including any OPLS bonds)
2. Optimize with orca (e.g. Opt B97-D3 def2-TZVP)
3. Analyze at higher level (e.g. SP RI-B2PLYP D3BJ def2-TZVP)
4. Great a training set directory that houses directories for each orca result. Each directory must include the .out, .engrad and system.cml file for that system
5. Run *python gradient_tersoff.py RUN_NAME*. The output parameters will start to appear in lammps/RUN_NAME_best.tersoff.
6. ...