

List of updates in download file

The source code, force field file, examples, etc., available for download will be updated from time to time as we implement new changes in the code or force field parameters that we feel may be useful for others. Because of the relatively small size of the code, we are not indexing a version number, rather we just replace the old download with the newer one. While all new versions of the code are tested and we believe are free of any major bugs, we do not issue any guarantees. We would appreciate any comments or suggestions regarding the above considerations.

2/22/2013

We have implemented the ability to use a Feynman-Hibbs effective potential (see source code for citations) to incorporate nuclear quantum effects (we have used this for simulating hydrogen adsorption in MOFs at low temperature). The Feynman-Hibbs contribution to the energy will either be calculated with a cutoff, or with a PME implementation, and this choice will be made based on the setting of “pme_disp”, so that it is consistent with the treatment of the dispersion energy.

9/12/2013

Quite a few updates here.

MC code: changed the indexing of Drude oscillators. Before, we assumed a Drude oscillator on every atom. This was inefficient though, for large molecules with a lot of Hydrogen atoms, where there is not a Drude oscillator on every atom. Therefore, now we introduce an array with maps Drude oscillators to their atom, so we don't have to make such an assumption. Also we have added the ability to calculate 3-body dispersion interactions

Lattice code: Added this source to the download. Made the same change with Drude oscillator indexing. Also scripted the execution of the three steps to a lattice model isotherm, namely the construction on the lattice free energy grid, discretizing the coarse grained potential, and running the lattice simulation.

1/13/2015

Quite a few updates here.

MC code:

1) changed code so that a hybrid lj/bkghm force field could be run, this was done so that we could compute h2o adsorption in MOFs using the SWM4-NDP water model for solute-solute interactions, and our force fields for solute-framework interactions

2) fixed small bug in pair_int.f90 evaluating lj energies. In lj_bkghm .eq. 2, if statement

```
! don't shift potential to values at cutoff (for molecule close to cutoff)

if( r_ij > lj_cutoff_use) then

    sum_long = sum_long + min (E_short + E_long -
lj_shift_store(atom_id1,atom_id2) , 0d0)

else

    sum_short=sum_short + E_short

    sum_long=sum_long + E_long - lj_shift_store(atom_id1,atom_id2)

endif
```

seems to be using uninitialized variable r_ij, which is only defined for the lj_bkghm .eq. 1 case. Pretty sure that this is the source of the bug in the previous version compile info.

3) implemented a Verlet neighbor list for lj interactions. This was done for lj energy and force routines. In order to do this, the subroutines were reorganized a bit, which is better for code readability anyway

4) changes made were creating look up tables for the Tang-Toennies functions for C6,C8,C10,C12. Also, the subroutines were clogged up with Case and if statements. We removed the scale_damp_exponents parameter since this probably never will be used again, and it added an if statement, and created an array for atom type pairs that flags whether the pair gets damped. All of this saves if statements, also reorganized into new subroutines to save case statements

5) found a very inefficient portion of code in the pme energy and force routines

in the reciprocal space energy and force pme routine, we need to form the product $F(Q) * C * B$, where Q, C, B are the standard matrices of the pme algorithm, and F(Q) is the fourier transform of Q

since F(Q) is complex, this product matrix is complex. But B, C are real. We previously used the code

$FQ = FQ * \text{cmplx}(CB, 0., 16)$

where we are overwriting FQ with the new product and CB is already C*B

change this to

$FQ = FQ * CB$

gives the same result but is much faster, and this makes a significant difference in the net time of the reciprocal space pme subroutines

6) added the "md" option for variable select_ensemble. This will perform an nve molecular dynamics simulation. This was trivial to implement, as we already have the hybrid_md_mc move subroutine. We use this subroutine, without resampling velocities, and always accepting moves. The time step must be small, and not updated.

To use the md option, one should be careful to use a small time step, and use appropriate energy options so that the forces are accurate. For instance, certain energy evaluations don't have forces coded in (which wasn't necessary for monte carlo); Three-body dispersion, pme dispersion, etc.

We tested the energy conservation on OPLS methane, and SAPT methane, and found nearly perfect energy conservation (<< 1kJ/mol drift) over hundreds of pico seconds for both systems using a long (20 Å) cutoff.

We tested neat methanol SAPT force field, and found significant energy drift. We think this is due to the Drude oscillators. This is being investigated.

7) Because the MD simulations take a long time, we tried to improve the speed of the pme-force calculation which is a bottleneck. A new subroutine "pairwise_ewald_real_space_force" was created, allowing for the option of gridding the pairwise real-space interactions for each atom-type pair. We tested the speed-up of using such table look-up functions, but found that large table arrays (1000000) were required for good accuracy, and the memory allocation introduced by these large arrays slowed the program down such that there was no speedup in using these lookups. So we probably won't use these lookups, but left them coded in as an option,

Also, in this same subroutine, we introduce a cutoff distance "screen_distance_max_sq" so that Tang-Toennies screening functions are not evaluated in this pme force calculation after this distance, which leads to negligible error, but saves some time.

8) added the ability to read in charges from the .fconf file

Lattice code:

1) The source code has been modified so that the lattice free energy grid can be scripted for large-scale parallelization, by using input parameters `start_grid_a`, `finish_grid_a`, etc, that tell the code which grid points to construct free energies for.

2) module has been added to calculate Q_{eq} charges for the crystal framework

3) Fixed an important bug. In the `pme.f90` subroutine, `drude_force_usegrid`, there was a bug in converting from scaled to absolute forces. In particular, the line

`force(i_mole,i_atom,:) = force(i_mole,i_atom,:) + matmul(ftmp(:),inv_box(:,:))*dble(pme_grid)`
was changed to

`force(i_mole,i_atom,:) = force(i_mole,i_atom,:) + matmul(inv_box(:,:),ftmp(:))*dble(pme_grid)`

because, the `inv_box` matrix is diagonal for orthorhombic cells, this bug was only expressed for non-orthorhombic unit cells, and because it's a force subroutine, it only matters if there are drude oscillators. The origin of the bug comes from the fact that the `inv_box` matrices in the monte carlo and lattice model codes are transposes of each other. This is unfortunate, and probably should be fixed. So be careful!

4) added the ability to read in charges in the `read_fconf` subroutine in `initialization_routines.f90`. This subroutine automatically detects the presence of charges by the number of arguments on each line, and overwrites the previous `chg` array if charges exist here. This modification is important to do high-throughput screening with ab-initio charges