JDFTx Effective Atom Theory

Brandon Li

January 22, 2024

Effective atom theory is a potential way of finding new materials with interesting properties, attempting to do so by parameterizing atom species with continuous variables. In algebraic DFT, the ion dependent part of the total energy, E_{ion}, is equal to

$$(Jn)^\dagger V_{loc} + \sum_{k,n,s} w_k \mathrm{tr}(C_k^\dagger V_{snk} M_s V_{snk}^\dagger C_k F_k) \tag{1}$$

where k runs over k-points and V_{loc}, the local part of the pseudopotential, is equal to $V_{loc} = \sum_{n,s} (V_{loc})_{sn}$. Here s, n run over the species and atom numbers, respectively. A mixed atom has a potential that is a linear combination of other potentials.

$$((V_{loc})_{mixed})_n = \sum_s \theta_{sn} (V_{loc})_{sn} \tag{2}$$

$$((V_{nl})_{mixed})_n = \sum_s \theta_{sn} V_{snk} M_s V_{snk}^\dagger \tag{3}$$

Optimizing energy over the weights θ_{sn} requires finding the gradient of E_{ion} with respect to the weights. The chain rule tells us

$$\frac{dE_{ion}}{d\theta_{sn}} = \frac{\partial E_{ion}}{\partial C} \frac{dC}{d\theta_{sn}} + \frac{\partial E_{ion}}{\partial \theta_{sn}} \tag{4}$$

If the wavefunction minimizes energy, then the term $\frac{\partial E_{ion}}{\partial C}$ is equal to zero. Thus,

$$\frac{dE_{ion}}{d\theta_{sn}} = \frac{\partial E_{ion}}{\partial \theta_{sn}} = (Jn)^\dagger (V_{loc})_{sn} + \sum_{k,n,s} w_k \mathrm{tr}(C_k^\dagger V_{snk} M_s V_{snk}^\dagger C_k F_k) \tag{5}$$

1 Using mixed atoms in calculations

An example JDFTx input file that employs mixed atoms is given below.

```
lattice face-centered Cubic 10.2612
latt-scale 1 1 1
kpoint-folding 2 2 2
elec-cutoff 40 160
elec-ex-corr gga-pbe
elec-smearing Fermi 0.005
symmetries none
```

```
ion-species SG15/$ID_ONCV_PBE.upf
add-mix mixA Si C
add-mix mixB Si P Al

coords-type lattice
ion mixA 0.3 0.7      0.00 0.00 0.00  0
ion mixB 0.3 0.4 0.3  0.25 0.25 0.25  0

dump-name mix.$VAR
dump End Ecomponents MixGrad
electronic-minimize energyDiffThreshold 1e-6 nIterations 100
```

The first seven lines set up the lattice and electronic parameters. Smearing must be enabled via the `elec-smearing` command, as this is necessary for determining the energy gradient. As of now, symmetries must turned off through the command "`symmetries none`".

Next, "`ion-species SG15/$ID_ONCV_PBE.upf`" tells the program to use those pseudopotentials which includes all the atomic elements that are referenced by mixed species. In the following lines, we create two mixed atom types which we call `mixA` and `mixB` comprised of elements Si+C and Si+P+Al respectively:

```
add-mix mixA Si C
add-mix mixB Si P Al
```

Finally, we add a `mixA` atom comprised of 30% Silicon and 70% Carbon at lattice coordinates `0.00 0.00 0.00` and prevent it from moving with the final parameter `0`. This is complimented by a `mixB` atom with 30% Silicon, 40% Phosphorus, and 30% Aluminum at coordinates `0.25 0.25 0.25`. As we see here, the mixing parameters always come after the name of the mixed ion followed by a coordinate triplet, and finally an ionic movement parameter.

After the physical system is specified, we instruct the program to dump the various DFT energies that comprise `Ecomponents` and the gradient of free energy with respect to mixing parameters, `MixGrad`. The "mix gradient" will be stored in a file ending with the extension `.mixgrad`. In our case, the file will look like

```
mix name   | atom | parameters
mixA         0     -8.398854665927    -10.361851493251
mixB         0    -14.715620907526    -19.274660265031    -58.706919943436
```

Each line contains the species name, atom number (starting from 0 in the same order they were specified), and the gradient of free energy with respect to the components of each species.

2 Downloading EAT

JDFTx with effective atom theory may be obtained by cloning my repository with the command

```
git clone https://github.com/StunningLlama/jdftx.git
```

and afterwards switching to the `jdftxmix` branch via

```
cd jdftx
git checkout jdftxmix
```

Updates and new improvements can be downloaded with the command `git pull`. JDFTx can then be compiled by following the instructions at `https://jdftx.org/CompilingBasic.html`.