Running linear-scaling DFT calculations for metallic systems in ONETEP with the AQUA-FOE method

Jolyon Aarons and Chris-Kriton Skylaris

May 3, 2018

This is a guide to performing FOE calculations on metals in ONETEP using the EDFT functionality. The methodology will be explained briefly, but for detailed information about how these methods work and are implemented, appropriate references are presented. We provide a example calculation, walk through and input files at the end of this document. Finally, to perform these calculations, a recent (as of 2018) version of ONETEP is required, for instance version 4.5.4.4, or the upcoming version 5.0.

1 Metals in ONETEP

Calculations on conductors have been possible in ONETEP with EDFT since 2013. In its default mode, EDFT will perform a diagonalisation of the Hamiltonian matrix and form the density kernel matrix from the eigenvectors and a Fermi-Dirac occupancy function of the eigenvalues. If your calculations have fewer than 1000 atoms, this mode is probably your best option. The method scales cubically with system size, however, and when you have a few thousand atoms calculations are impractically slow on current supercomputers.

On larger systems, it is possible that you will have sufficient sparsity in the Hamiltonian matrix that a Fermi-Operator-Expansion based approach to constructing the density kernel will be faster. The idea behind this approach is that a power series expansion of the Fermi-Dirac distribution function can be applied directly to the Hamiltonian matrix, as matrix multiplications are linear operations. Hence, the density kernel matrix can be constructed from matrix powers of the Hamiltonian matrix, without diagonalisation. The main caveat with this is that sparsity must be exploited, or such a method is likely to be slower than the diagonalisation based alternative, due to the many matrix multiplications that are required. Since sufficient sparsity is only obtained for metals with more than about 1000 atoms, we recommend the use of this method only for larger systems.

The way that we apply the matrix analogue of the Fermi-Dirac distribution function,

$$K = [I + \exp((H - \mu I)\,\beta)\,]^{-1}, \tag{1}$$

without diagonalisation or the poor conditioning of applying the above operations directly is to firstly compute an expansion of a system of electrons at an

integer multiple (2^n) of the electronic temperature, so that:

$$K^{\mathrm{HOT}} = [I + \exp((H - \mu I)\beta/2^n)\,]^{-1}. \tag{2}$$

The advantage of applying an expansion to hotter electrons is that the argument of the exponential function is reduced in magnitude, so fewer terms in expansion of the Fermi-Dirac distribution function are needed for a good approximation. In practice, we use a low order Chebyshev approximation to equation 2. To recover K, the density kernel matrix at the target temperature, we need to anneal K^{HOT} by halving the temperature, $n-1$ times. This can be achieved by applying the matrix analogue of the hyperbolic tangent double angle formula after noting that

$$K^{\mathrm{HOT}} = \frac{1}{2}\left(I + \tanh\left((H - \mu I)\frac{\beta}{2^{n+1}}\right)\right). \tag{3}$$

The annealing formula is given as:

$$K^{\mathrm{HOT}/2} = \frac{1}{2}\left(\frac{4K^{\mathrm{HOT}} - 2I}{I + (2K^{\mathrm{HOT}} - I)^2} + I\right), \tag{4}$$

which we do not apply directly in practice (this would cost 1 matrix product and 1 matrix solve / inversion), but expand equation 4 as another Chebyshev polynomial expansion, which is cheaper due to equation 4 being very well conditioned (we know that the eigenvalues of the denominator are necessarily between 1.0 and 2.0).

The application of equation 2 assumes that we know the chemical potential, μ. In practice, we do not. We instead start with a trial chemical potential and improve K in the sense that we drive it towards the electron conserving chemical potential. We achieve this by knowing how wrong we were at a given μ_i,

$$\Delta N_e = N_e - \mathrm{trace}\left(K(\mu_i)\right), \tag{5}$$

and by using hyperbolic trigonometry to correct by this by a change in chemical potential, $\Delta\mu$ without recalculating the expansion:

$$[I + \exp\{(H - \mu I)\beta \pm \beta\Delta\mu I\}]^{-1} = \frac{1}{2}\left(I + \frac{2K \pm \tanh\left(\frac{\beta\Delta\mu}{2}\right) I - I}{I \pm \tanh\left(\frac{\beta\Delta\mu}{2}\right) \times (2K - I)}\right). \tag{6}$$

Likewise with this equation, we do not apply it directly but expand it as a Chebyshev polynomial to avoid an inversion. Once again, this works well because we know that this is a well conditioned problem. To know how much to adjust the chemical potential we can use the derivative:

$$\frac{\partial N_e}{\partial \mu} = -\frac{\beta}{4}\left(1 - \mathrm{trace}(K^2)\right), \tag{7}$$

and the electron residual ΔN_e.

The final piece in this puzzle is to construct the electronic entropy, which we find as a matrix, taking its trace to find the scalar electronic entropy. We avoid calculating the Entropy as

$$\mathcal{S} = \mathrm{tr}[\mathbf{K}\ \ln(\mathbf{K}) + [\mathbf{I} - \mathbf{K}]\ \ln(\mathbf{I} - \mathbf{K})]\ , \tag{8}$$

because this involves calculating two expensive matrix logarithms. Instead we use a further expansion for this quantity, this time taking that of the Fermi-Dirac entropy expression. When expanding this expression as a Chebyshev expansion, many terms are required to reach a good approximation at zero and one occupancy. To avoid this problem, we expand a form which maintains the property that the single particle entropy is zero at zero and one occupancy with fewer terms. This is:

$$s(x) \simeq ax^2 + \frac{b}{c + dx - dx^2} - e - ax = y(x), \tag{9}$$

where $a = 1.96056$, $b = 0.0286723$, $c = 0.114753$, $d = 1.98880$ and $e = 0.249860$. We then refine this with further expansions.

In practice, because sparsity is imposed on the density kernel and entropy matrices, we do not have perfectly accurate implicit eigenvalues. Some of these may be above one, so when taking large powers of the matrix, as we do in the power expansion of equation 9, the (implicit) eigenvalues of the entropy may explode, causing a problem. To detect and avoid this problem, we use a very simple quadratic approximation to the entropy.

The quadratic

$$f(x) = 3x^2 - 3x, \tag{10}$$

the coefficients of 3 minimise the integral of $x*Ln(x)+(1-x)*Ln(1-x)-cx^2-cx$ over [0:1]. The maximum error of this approximation is at $x = 0.5$, where the error is 0.05685 or 8.2% of the correct value at 0.5. This indicates that in the very worst case where every eigenvalue of **KS** is 0.5, then the error on the entropy from this approximation is 8.2%. The value of this is that **KSKS** is easy to calculate reliably, compared with a high order series expansion... i.e. if there is an eigenvalue of K which is a little above 1 because of noise due to truncation, the corresponding eigenvalues in the entropy matrix can be very large. This can accumulate in the entropy value, if there are many of these greater than 1 eigenvalues, to give a completely wrong value. We can, therefore, check the value of entropy we have calculated by comparing it with this quadratic approximation. If the absolute difference between the two values is less than 8.2the high order value, then we assume that things have gone well and do nothing. If this is not the case, then there was a high amount of error accumulated and so we'll print a warning and a refined quadratic approximation (a quartic).

In practice, when we take all of these power expansions, we need to consider the tensorial transformation properties of the quantities we are considering, for instance: the Hamiltonian matrix, $H_{\alpha\beta}$ is covariant, so taking a power requires that we raise the appropriate indices using the metric : $(H_{\alpha\beta})^n = (H_{\alpha\gamma}S^{\gamma\delta})^n S_{\delta\beta}$. In practice we can do this in one of two ways, either invert the metric / overlap matrix $S^{\alpha\beta} = (S_{\alpha\beta})^{-1}$ and multiply by the Hamiltonian to give a Hamiltonian matrix in the natural representation (contra-covariant) or solve an approximate problem in the style of Haydock *et al*[2]. We then need to store the mixed contra-covariant Hamiltonian matrix, which we choose to represent in the sparsity pattern suggested by Haydock *et al*. This is the covariant Hamiltonian matrix sparsity pattern by default, but the user may specify an optional argument to

effectively increase the interaction regions of this sparsity pattern with respect to the covariant case.

More information on our method may be found in [1].

2 Setting up a Calculation

The most important thing to understand firstly is that this method is probably only sensible for large metallic systems. If your system has a band-gap, it would be wise to explore standard insulating ONETEP (LNV) first. This method should still work, but LNV ought to be much faster.

The calculation should perform EDFT as the electronic minimiser. Please set

```
EDFT : T
```

in your input file. As you will be performing a finite electronic temperature calculation with the Fermi-Dirac occupancy distribution function, it would also be wise to choose an electronic smearing (temperature). The default value is 0.1 eV, which may well be sufficient in your case. Please bear in mind that with FOE enabled, unlike with EDFT with diagonalisation the calculation is likely to be faster with higher electronic temperature. Depending on the property you wish to probe, you may get away with a higher temperature, but it is very unlikely that the smearing should be above 1.0 eV. Set the smearing as

```
EDFT_SMEARING_WIDTH : 0.25 eV .
```

You may also at this point wish to set some EDFT stopping criteria, as the defaults are likely to be too tight for EDFT with FOE. It is possible to tweak the FOE options to get under tight thresholds, but at the expense of reduced performance.

```
EDFT_FREE_ENERGY_THRES : 1e-5 Ha
EDFT_COMMUTATOR_THRES : 1e-4
```

should be quite realistic in most cases and will certainly serve as an adequate starting point.

The number of extra bands parameter in EDFT should always be set to the maximum, which is done by setting:

```
EDFT_EXTRA_BANDS : -1
```

This should be sufficient for the EDFT part of the calculation, next we will need to configure the FOE. Firstly, please enable it in your calculation with

```
FOE : T
```

This alone will perform a FOE calculation as we have described in this document, but unless you have already set a kernel truncation, then your density kernel matrix will be dense and you might as well just run a normal EDFT calculation with diagonalisation. One possible exception to this is if you do not have access to Scalapack or a ONETEP binary compiled against Scalapack, or your Scalapack distribution is poorly optimised or broken. In this case it may still be preferable to avoid the diagonalisation.

To achieve some sparsity in the density kernel matrix, we have various options and strategies.

1. Just to use a geometric kernel truncation:
   Here we enforce a simple sparsity pattern on the density kernel matrix by ensuring that matrix elements resulting from NGWFs on atoms separated by a large enough distance are zero.

   ```
   kernel_cutoff : 20 Ang
   ```

   a larger value will result in a more accurate answer, but also give a less sparse matrix which will have performance implications. This parameter is material-dependent and you are encouraged to play with it to find a happy medium.

2. Use a sparsity pattern inspired by the power series expansions which we use extensively in the FOE methods. The FOE method is based on the idea that we only need a finite number of terms in the power expansion to get sufficient accuracy. In the same spirit, we can truncate the expansion to give a sparsity pattern of the density kernel matrix as a low order power of the Hamiltonian matrix. In ONETEP, this can currently be set to be the squared power by using:

   ```
   H2DENSKERN_SPARSITY : T
   ```

If you opt for the second option, then the quadratic order term in the expansion may not be a sufficiently accurate sparsity pattern to achieve the accuracy you were hoping for, but higher order powers are likely to be too dense to be useful and are not exposed to the user. Instead to reach your desired tolerances, we recommend you consider using the radius multiplier option.

This option extends the effective range of the Hamiltonian matrix (when represented in contra-covarient, mixed form (as described above) and the density kernel matrix is formed from the square of this increased range matrix. The option is given in terms of a multiplier, which multiplies the NGWF radius to produce the resulting sparsity patterns. This option increases accuracy at the expense of performance and it is up to you as a user to find an appropriate value. A starting point may be around 1.2 to 1.5 times:

```
CONTRACOHAM_RADMULT : 1.2
```

It is worth pointing out here that you must also ensure your calculation is converged with respect to plane wave kinetic energy cutoff and NGWF radius, as well as number of NGWFs per atom. If you do opt for Hamiltonian matrix squared sparsity in the density kernel matrix, however, you will find that NGWF radius plays a particularly big role in the accuracy of your calculation. This is because it is effectively the parameter which controls the sparsity of your density kernel in this mode. Increasing the radius multiplier will be cheaper (to an extent) than increasing the NGWF radii, but you should make sure that your NGWF radii are sufficient first!

The final parameter which is important to achieve your tolerances is the μ-tolerance in the chemical potential search. In practice this should be *at least* an order of magnitude smaller than your EDFT energy tolerances. You may find that you can improve the performance of your calculations by adjusting this, but you will find that the gains will be minor, because the number of extra matrix multiplications near to convergence is minimal compared with the number far from the correct chemical potential. This is set as:

```
FOE_MU_TOL : 1e-6 eV
```

Beyond this, there is little else to know, from a user perspective about performing FOE calculations in ONETEP. A general point is that these calculations tend to be expensive. If you are running a metallic system which is big enough to exhibit exploitable sparsity in its density kernel matrix, then you are probably looking at a system with thousands of atoms. In this case you will need to be running on a supercomputer and using a parallel version of ONETEP with as many cores as you can use.

If you are looking for ways to speed up your calculations, look at OpenMP / MPI parallelism rather than just MPI, also try to optimise your convergence criteria to have the lowest values you can get away with while maintaining your desired level of accuracy. You may also want to look at the option of using fixed NGWFs and only optimising the density kernel. If you go down this route, please ensure that you use a multiple ζ set of NGWFs and probably polarisation functions. All of these extra points are really general ONETEP comments and information on them can be found in the other tutorials on the ONETEP website.

3 An Example

As an example we will set up a single-point energy calculation on bulk Magnesium, using PAW. We'll opt for a decent cut-off energy of 700 eV and the PBE exchange-correlation functional.

```
task singlepoint
cutoff_energy 700 eV
charge 0
xc_functional PBE
paw : T
```

We need to enable EDFT and set it up as we have described:

```
!EDFT
edft : T
edft_smearing_width: 0.5 eV
edft_maxit: 10
```

10 EDFT iterations may be too few, and 0.5 eV smearing may be too hot for production calculations, but for the sake of a demonstration, they speed things up a bit.

We also should opt for as much threading as we can get away with because this will be made use of in every matrix product:

```
!THEADING
threadsmax 8
threadsperfftbox 1
threadsnumfftboxes 8
threadspercellfft 8
threadsnummkl 8
```

For instance, if your supercomputer has two 8 core processors per node, you could opt for 8-way OpenMP maximally, although in practice, you might want to reduce this to 4-way, if you need the MPI ranks.

Now lets save some matrices and information:

```
! properties
 do_properties F

! I/O
 output_detail VERBOSE
 timings_level 1
 write_xyz T
 write_denskern T
 read_denskern F
 write_tightbox_ngwfs T
 read_tightbox_ngwfs F
 write_hamiltonian T
 read_hamiltonian F
```

In practice it is a good idea to write everything, in case you need to continue this calculation, but in the extreme case where your NGWFs and matrices are too big to write to disk (not enough space or it's taking too long) then you can disable this.

Now we can set up NGWF optimisation:

```
! NGWF optimization
 k_zero 2.5
 maxit_ngwf_cg 25
```

and FOE:

```
 foe : T
 dense_foe : F
 H2DENSKERN_SPARSITY : T
```

Before setting up the Mg:

```
 %block species
  Mg Mg 12 4 8.0
 %endblock species

 %block species_atomic_set
 Mg "SOLVE"
 %endblock species_atomic_set
```

for instance. Or, if you wanted to set maxit_ngwf_cg to be zero and run a fixed NGWF calculation, you could run a multiple ζ + polarisation calculation with:

```
 %block species
  Mg Mg 12 13 8.0
 %endblock species
```

```
 %block species_atomic_set
Mg "SOLVE conf=2s2 2p6 3s2 3p0|P"
%endblock species_atomic_set
```

For more info on this, please see the pseudoatomic solver tutorial. Add the PAW data:

```
%block species_pot
Mg "mg_pbe_v1_abinit.paw"
%endblock species_pot
```

Finally lets add a simulation cell and some atoms:

```
%BLOCK lattice_cart
ang
 13.2119999999999997   0.0000000000000213   0.0000000000000213
  0.0000000000000000  10.8379999999999992   0.0000000000000175
  0.0000000000000000   0.0000000000000000  10.4299999999999997
%ENDBLOCK lattice_cart

%BLOCK positions_abs
ang
Mg         3.30300        1.99300        0.00000
Mg         1.65100        4.70200        0.00000
Mg         3.30300        0.11500        2.60800
Mg         1.65100        2.82500        2.60800
Mg         0.00000        1.99300        0.00000
Mg         4.95400        4.70200        0.00000
Mg         0.00000        0.11500        2.60800
Mg         4.95400        2.82500        2.60800
Mg         3.30300        1.99300        5.21500
Mg         1.65100        4.70200        5.21500
Mg         3.30300        0.11500        7.82300
Mg         1.65100        2.82500        7.82300
Mg         0.00000        1.99300        5.21500
Mg         4.95400        4.70200        5.21500
Mg         0.00000        0.11500        7.82300
Mg         4.95400        2.82500        7.82300
Mg         3.30300        7.41200        0.00000
Mg         1.65100       10.12100        0.00000
Mg         3.30300        5.53400        2.60800
Mg         1.65100        8.24400        2.60800
Mg         0.00000        7.41200        0.00000
Mg         4.95400       10.12100        0.00000
Mg         0.00000        5.53400        2.60800
Mg         4.95400        8.24400        2.60800
Mg         3.30300        7.41200        5.21500
Mg         1.65100       10.12100        5.21500
Mg         3.30300        5.53400        7.82300
Mg         1.65100        8.24400        7.82300
Mg         0.00000        7.41200        5.21500
```

```
Mg         4.95400       10.12100        5.21500
Mg         0.00000        5.53400        7.82300
Mg         4.95400        8.24400        7.82300
Mg         9.90900        1.99300        0.00000
Mg         8.25700        4.70200        0.00000
Mg         9.90900        0.11500        2.60800
Mg         8.25700        2.82500        2.60800
Mg         6.60600        1.99300        0.00000
Mg        11.56000        4.70200        0.00000
Mg         6.60600        0.11500        2.60800
Mg        11.56000        2.82500        2.60800
Mg         9.90900        1.99300        5.21500
Mg         8.25700        4.70200        5.21500
Mg         9.90900        0.11500        7.82300
Mg         8.25700        2.82500        7.82300
Mg         6.60600        1.99300        5.21500
Mg        11.56000        4.70200        5.21500
Mg         6.60600        0.11500        7.82300
Mg        11.56000        2.82500        7.82300
Mg         9.90900        7.41200        0.00000
Mg         8.25700       10.12100        0.00000
Mg         9.90900        5.53400        2.60800
Mg         8.25700        8.24400        2.60800
Mg         6.60600        7.41200        0.00000
Mg        11.56000       10.12100        0.00000
Mg         6.60600        5.53400        2.60800
Mg        11.56000        8.24400        2.60800
Mg         9.90900        7.41200        5.21500
Mg         8.25700       10.12100        5.21500
Mg         9.90900        5.53400        7.82300
Mg         8.25700        8.24400        7.82300
Mg         6.60600        7.41200        5.21500
Mg        11.56000       10.12100        5.21500
Mg         6.60600        5.53400        7.82300
Mg        11.56000        8.24400        7.82300
%ENDBLOCK positions_abs
```

Bibliography

[1] J. Aarons and C.-K. Skylaris. Electronic annealing Fermi operator expansion for DFT calculations on metallic systems. *The Journal of Chemical Physics*, 148(7):074107, 2018.

[2] A. Gibson, R. Haydock, and J. P. LaFemina. Ab initio electronic-structure computations with the recursion method. *Physical Review B*, 47(15):9229, 1993.