Spherical wave resolution of identity (SWRI), Hartree-Fock exchange (HFx), hybrid functionals and distributed multipole analysis (DMA) in ONETEP

Jacek Dziedzic and James C. Womack

v2.00 June 2020

This manual pertains to ONETEP versions v5.3.4.0 and later.

1 The basics

ONETEP offers linear-scaling calculation of Hartree-Fock exchange (HFx) and linear-scaling distributed multipole analysis (DMA) through a variant of density fitting. In this approach products of two NGWFs are approximated (fitted) using an auxiliary basis of spherical waves centred on the atoms on which the two NGWFs reside. Before the expansion can be performed, an overlap matrix between spherical waves on all centres whose NGWFs overlap needs to be calculated. The resultant matrix will be termed the *metric matrix*, and the process through which it is obtained will be termed *spherical wave resolution of identity (SWRI)*. Understanding the keywords controlling SWRI is thus essential for any calculation involving HFx or DMA.

2 Limitations

All calculations using SWRI have these limitations:

- Open boundary conditions (OBCs) are assumed in all uses of SWRI. If your calculation uses OBCs (via cut-off Coulomb, Martyna-Tuckerman or multigrid [2]), this is a non-issue. If your calculation technically uses PBCs, but you have sufficient vacuum padding in every direction (effectively making it approximately OBC), this is a non-issue. If your system is truly extended in any direction (at least one NGWF sphere sticks out of the cell and wraps around), this is an issue, as neither HFx nor DMA will work (producing nonsense). This is not checked!
- All species must have identical NGWF radii. Without this simplification the numerical methods used in SWRI would get ugly. Typically this is a non-issue, just increase the NGWF radii to the largest value. This might admittedly be an issue if you have a single species that requires a large NGWF radius. This is checked against and ONETEP will not let you do SWRI unless all NGWF radii are identical.
- Although well-controllable, density fitting is an approximation. Using settings that are too crude can lead to inaccurate results and poor NGWF convergence.
- HFx is incompatible with PAW and ONETEP will refuse to use both simultaneously.
- HFx and DMA are incompatible with complex NGWFs and ONETEP will refuse to use both.
- HFx does not work with atoms that have 0 NGWFs, although this is hardly a limitation.

3 Spherical wave resolution of identity

3.1 Generating the metric matrix

Calculation of the metric matrix [1, Sec. II.D.1] is challenging. This is because products of spherical waves (SWs) and products of a spherical wave with a potential of a spherical wave (SWpot) oscillate rapidly and cannot be reliably integrated on a Cartesian or radial grid. ONETEP calculates the elements of the metric matrix by a piecewise approximation of SWs and SWpots with high-order Chebyshev polynomials. Once this is done, their overlaps can be calculated by overlapping a large number of polynomials, which is easy, but time consuming.

For a fixed set of atomic positions and chosen NGWF radii the metric matrix needs only be calculated once, this is done during initialisation. The metric matrix does not depend on the NGWFs themselves or even the number of NGWFs per atom.

In ONETEP versions prior to 5.1.2.3, metric matrix elements are evaluated using the method described in Ref. 1 (section II.D.1), i.e. by 3-D integration over piecewise expansions of SWs/SWpots in Chebyshev polynomials. This is the "3-D Chebyshev" (3Dc) scheme.

For versions $\geq$ 5.1.2.3, an alternative metric matrix scheme is available in which metric matrix elements are decomposed into products of 2-D numerical and 1-D analytic integrals. This is the "2-D numerical, 1-D analytic" (2Dn-1Da) scheme.

The separation of the 3-D integral in the 2Dn-1Da scheme is made possible by expressing metric matrix elements for each atom pair in a *molecular* coordinate frame with the z-axis pointing along the vector between atomic centres and with a set of SWs aligned in this coordinate system. When expressed in spherical polar coordinates, the 1-D integration (over the azimuthal angle, ϕ) is simple to evaluate analytically. The 2-D integration over (r, θ) can be performed by the same approach used in the 3Dc method, i.e. evaluating integrals over piecewise expansions of the (r, θ)-dependent parts of the SWs and SWpots in Chebyshev polynomials.

To obtain the metric matrix in the original set of SWs, which are aligned parallel to the z-axis of a *global* (simulation cell) coordinate frame, the SWs in the molecular coordinate frame must be represented in terms of the SWs in the global coordinate frame. This is achieved by applying a rotation matrix to the real-spherical harmonic (RSH) components of the SWs/SWpots associated with each atom pair.

For versions $\geq$ 5.1.5.0, the 2Dn-1Da scheme is the default for evaluating the electrostatic metric matrix. The 3Dc scheme remains the default for the overlap metric matrix (`O`), since overlap metric matrix evaluation is not currently supported in the 2Dn-1Da scheme.

Use of the 2Dn-1Da scheme to evaluate the electrostatic metric matrix is strongly recommended, as it reduces the computational cost (in terms of memory and execution time) of evaluating the matrix by orders of magnitude compared to the 3Dc scheme (with no apparent loss of accuracy).

To set up SWRI (for both 2Dn-1Da and 3Dc schemes), define the following block:

```
%block swri
  myname l_max q_max metric N_i N_o flags
%endblock swri
```

Replace *myname* with a user-readable name for the SWRI, you will use it later to tell HFx or DMA which SWRI to use (as you can have more than one SWRI).

$l_{\max}$ is the maximum angular momentum in the SW basis. Supported values are 0 (s-like SWs only), 1 (s- and p-like SWs), 2 (s-, p- and d-like SWs), 3 (s-, p-, d- and f-like SWs) and 4 (you get the idea). For HFx choose 2 for a crude calculation, 3 for good quality, and 4 for extreme quality. 0 and 1 will likely be too crude to fully NGWF-converge. Expect 2 to recover more than 99% of HFx energy, and 3 to recover about 99.7%. See [1, Fig. 8] for an illustrative guide. For DMA choose 0 if you're only interested in atomic charges, 1 if you want charges and dipoles, and 2 if you want charges, dipoles and quadrupoles. There is no point in using 3 or 4 for DMA. Be aware that the computational cost grows as $\mathcal{O}(l_{\max}^4)$ (for the entire calculation), and the memory requirement grows as $\mathcal{O}(l_{\max}^4)$ (for the entire calculation).

$q_{\max}$ is the number of Bessel (radial) functions in the SW basis. Adding more Bessel functions increases the size of the basis (linearly) and improves quality. However, each subsequent Bessel function oscillates more rapidly and beyond a certain point (at about 15 Bessel functions) they become difficult to represent on Cartesian grids of typical fineness. As a guide, use 7 for a crude calculation, 10 for reasonable accuracy and 14 for extreme accuracy. Again, see [1, Fig. 8] to see a graphical representation of the impact of this setting on accuracy. There is no upper limit on the value of this parameter; however, ONETEP will refuse to include Bessel functions that oscillate too rapidly for your KE cutoff – you will get a warning and high-numbered Bessel functions will be removed from the basis. The computational cost grows as $\mathcal{O}(q_{\max}^2)$, and so does the memory requirement.

Replace *metric* with `V` to use the electrostatic metric in the SWRI, or with `O` to use the overlap metric. Specifying `VO` or `OV` will generate both metric matrices, although that is not usually done. In general, prefer the electrostatic metric – the error in the energy due to the density fitting approximation is then second-order in the fitting error, while for the overlap metric it is first-order.

As mentioned above, for ONETEP versions $\geq$ 5.1.5.0 the electrostatic metric (`V`) is evaluated by default using the more efficient 2Dn-1Da scheme. The overlap metric (`O`) cannot (currently) be evaluated using this scheme, so is evaluated using the more costly 3Dc scheme. Since only a single metric matrix scheme may be used at a time, if both metric matrices are requested (`VO` or `OV`) then ONETEP will fall back to the 3Dc scheme. In this situation, it is worth considering whether your calculation can be run with only the electrostatic metric in order to take advantage of the more efficient 2Dn-1Da scheme.

N_{i} is the number of intervals into which the integration domain will be divided along each axis for the purpose of Chebyshev interpolation. In the 3Dc scheme, this is the localisation sphere of an SW (an NGWF sphere, see [1, Sec. II.D.1]), while in the 2Dn-1Da scheme this is a half-disc with the same radius. For 3Dc, 8 is the bare minimum, 10 is crude, 12 is accurate and 14 is extremely accurate. You should avoiding going overboard (recommended value is 12), since the computational cost grows as $\mathcal{O}(N_{\mathrm{i}}^3)$ (only for the SWRI stage, the remainder of the calculation is not sensitive to this value). The memory requirement grows as $\mathcal{O}(N_{\mathrm{i}}^3)$ (only for the SWRI stage). See [1, Fig. 5] to see how the accuracy of the metric matrix depends on this parameter when using the 3Dc scheme. For 2Dn-1Da, the computational cost ($\mathcal{O}(N_{\mathrm{i}}^2)$) and memory requirements ($\mathcal{O}(N_{\mathrm{i}}^2)$) are considerably lower, so it is practical to use $N_{\mathrm{i}} = 14$ or larger for routine calculations. In this case, it is recommended to use 12 or greater. In particular, very crude (less than 10) values should be avoided when using 2Dn-1Da. Testing of DMA with the 2Dn-1Da scheme suggests that the 2Dn-1Da scheme is more sensitive than 3Dc to lower values of N_{i} (i.e. larger errors are produced in multipole moments compared to values converged with respect to N_{i} and N_{o}).

N_{o} is the order of Chebyshev polynomials used in the interpolation. Just like for N_{i}, for the 3Dc scheme 8 is the bare minimum, 10 is crude, 12 is accurate and 14 is extremely accurate. Again, you should avoid going overboard (recommended value[1] is 12), since the computational cost grows as $\mathcal{O}(N_{\mathrm{o}}^4)$ (only for the SWRI stage, the remainder of the calculation is not sensitive to this value). The memory requirement grows as $\mathcal{O}(N_{\mathrm{o}}^3)$ (only for the SWRI stage). See [1, Fig. 5] to see how the accuracy of the metric matrix depends on this parameter when using the 3Dc scheme. For 2Dn-1Da, the computational cost ($\mathcal{O}(N_{\mathrm{o}}^3)$) and memory requirements ($\mathcal{O}(N_{\mathrm{o}}^2)$) are again considerably lower, so it is practical to use $N_{\mathrm{o}} = 14$ or larger for routine calculations. In this case, it is recommended to use 12 or greater. For the reasons outlined above for N_{i}, very crude (less than 10) values of N_{o} should be avoided when using 2Dn-1Da.

For DMA, which is performed during a properties calculation[2], crude settings will simply lead to less accurate multipoles. In HFx, on the other hand, settings that are too crude would prevent convergence because the exchange matrix would not be sufficiently symmetric. ONETEP will abort your calculation if the exchange matrix is later found to not be symmetric to at least 3.4 digits. To avoid frustration, do not go below $N_{\mathrm{i}} = 10$, $N_{\mathrm{o}} = 10$.

For the 2Dn-1Da scheme, the cost of evaluating the metric matrix is typically significantly smaller than the overall cost of the subsequent calculation. In this case, it is practical to routinely use higher N_{i} and N_{o} values

[1] There is a caveat here when using the overlap metric (which you shouldn't be doing anyway). At the boundary of the NGWF localisation region a SW has a derivative discontinuity, while a SWpot is smooth together with its derivative. This discontinuity is poorly approximated with high-order polynomials (Runge effect), and the problem becomes worse when the number of intervals is small. The setting N_{i}=12, N_{o}=12 is a poor choice in this case. When using the overlap metric, always use the lowest possible Chebyshev order (2, a parabola) and a large number of intervals to compensate. A decent setting, and with a comparable cost, would be N_{i}=72, N_{o}=2. Of course you should not be using the overlap metric in the first place!

[2] Unless you're doing QM/MM calculations with polarisable embedding.

(e.g. $N_\mathrm{i} = N_\mathrm{o} = 14$ or 16).

Generating the metric matrix can be costly (particularly when using the 3Dc scheme). When restarting calculations that crashed, ran out of walltime, for restarts to do properties, or re-runs with different settings it makes sense to save the metric matrix to a file and re-use it during restarts. A metric matrix can be reused as long as the positions of the atoms and the NGWF radii did not change. *flags* is a combination of one or more letters or numbers: `W`, `R`, `P`, `Q`, `X`, `E`, `D`, `2`, `3`, controlling the behaviour of ONETEP during SWRI. The following flags instruct ONETEP to perform particular actions:

- `W` – writes the metric matrix to a file, once it has been calculated in its entirety. The file will have the extension `.vmatrix` for the electrostatic metric matrix, and `.omatrix` for the overlap metric matrix. This is highly recommended.
- `R` – reads the metric matrix from a file, instead of calculating it. The file will have the extension `.vmatrix` for the electrostatic metric matrix, and `.omatrix` for the overlap metric matrix. This is highly recommended for restart calculations. ONETEP will not allow you to use this flag when it knows the ions will move (TASK GEOMETRYOPTIMIZATION, TRANSITIONSTATESEARCH, MOLECULARDYNAMICS, PHONON, FORCETEST), as the metric matrix gets invalidated once an ion moves.
- `P` – will instruct ONETEP to print the metric matrix in text form straight to the output. This can be useful for visual inspection and debugging, although be aware that for larger systems the output can be bulky.
- `Q` – will instruct ONETEP to quit immediately after the metric matrix is calculated (and potentially written and/or printed). This can be useful if the SWRI stage is run separately from the main calculation, e.g. on a large number of CPU cores that would be excessive for the main calculation.
- `X` – means "none of the above" and should be used if you don't intend to write, read, print the metric matrix and you don't want ONETEP to quit at this stage.

The remaining flags change how the SWRI is performed:

- `2` – forces use of the 2Dn-1Da metric matrix evaluation scheme, overriding the default selection. Note that the 2Dn-1Da scheme is currently only available for evaluation of the electrostatic metric matrix (`V`) and ONETEP will abort with an error if the `2` flag is used in combination with the overlap metric matrix (`O`).
- `3` – forces use of the 3Dc metric matrix evaluation scheme, overriding the default selection.

The `2` and `3` flags only have effect when computing the metric matrix. When reading the matrix from disk in full (`R` flag), the flags have no effect, as the matrix has already been precomputed. When reading the matrix in part from atomblocks (see below), the flags will only affect atomblocks that are not read from disk (i.e. need to be computed).

By using `W` and `R` you can re-use a fully calculated metric matrix. For large jobs which take many CPU-core-hours you may want to re-use *partial* results simply because you may not have enough walltime to run the SWRI calculation to completion. By default ONETEP writes partial results (metric matrix atomblocks) to files (`*.[vo]matrixblock`) as it churns through the calculation. These matrixblocks will be automatically read from files if they can be found – i.e. before starting to calculate a block, ONETEP will always first look for a corresponding file to try and avoid the calculation, regardless of your *flags*. Thus, if your SWRI calculation is interrupted, retain the matrixblock files to make the next run complete faster. If you wrote the completed matrix to a file, there is no point in keeping the matrixblock files and you should delete them to save disk space. If you would rather not have to delete them manually, specify `E` (for "erase") in *flags* and they will not be kept (or indeed written to). Each matrixblock file encodes the position of the two atoms between which it is calculated in the filename. This proves useful in TASK PHONON calculations and TASK GEOMETRYOPTIMIZATION calculations with some atoms fixed – atomblocks between pairs of atoms that did not move will not be recalculated needlessly, but rather reloaded from files, unless you specify `E`. Finally, the expert option `D` instructs ONETEP to disassemble the fully calculated metric matrix into atomblocks (best used in combination with `R` and `Q`). This can be useful if you saved the metric matrix to a file, deleted the matrixblock files, and later change your mind.

3.2 Examples

This creates an SWRI called `for_hfx`, with an expansion up to l=3, 10 Bessel functions, using the electrostatic metric. Chebyshev interpolation will use 12 intervals and 12-order polynomials. The metric matrix will be written to a file. That would be standard for a HFx calculation.

```
%block swri
  for_hfx 3 10 V 12 12 W
%endblock swri
```

Like above, but will read the metric matrix from a file instead of calculating it.

```
%block swri
  for_hfx 3 10 V 12 12 R
%endblock swri
```

This creates an SWRI called `for_dma`, with an expansion up to l=2, 12 Bessel functions, using the electrostatic metric. Chebyshev interpolation will use 10 intervals and 12-order polynomials. The metric matrix will not be written to a file. That would be standard for a DMA calculation.

```
%block swri
  for_dma 2 12 V 10 12 X
%endblock swri
```

This creates an SWRI called `hiqh_qual`, with an expansion up to l=4, 16 Bessel functions (extremely large and accurate SW basis set), using the overlap metric (not the best choice). Chebyshev interpolation will use 60 intervals and 2-order polynomials (parabolas). The metric matrix will be written to a file, printed out in text form, the matrixblock files will be erased, and ONETEP will quit.

```
%block swri
  high_qual 4 16 O 60 2 WPEQ
%endblock swri
```

This creates an SWRI called `for_hfx_and_dma`, with an expansion up to l=2, 9 Bessel functions, using the electrostatic metric. Chebyshev interpolation will use 14 intervals and 14-order polynomials. The 2Dn-1Da metric matrix evaluation scheme has been explicitly selected (for versions $\geq$ 5.1.5.0, this would not be necessary, as 2Dn-1Da is the default for the electrostatic metric). The resulting metric matrix will be written to a file and the matrixblock files will be erased.

```
%block swri
  for_hfx_and_dma 2 9 V 14 14 WE2
%endblock swri
```

As above, but with the 3Dc metric matrix scheme explicitly selected. This will likely be very costly compared to using the 2Dn-1Da scheme.

```
%block swri
  for_hfx_and_dma 2 9 V 14 14 WE3
%endblock swri
```

3.3 Choosing which species participate in a SWRI

For every SWRI defined like above you need to specify which atomic species participate in it. This allows performing an SWRI for a subsystem, e.g. doing DMA only for atoms of a solute in the presence of a solvent, but not for atoms of the solvent itself. In such a scenario atomblocks only need to be calculated between atoms such that at least one atom belongs to the SWRI. For HFx this is less meaningful, and you will want to list all your species in the block. Note how the block name includes the name of the SWRI defined above and may look like this:

```
%block species_swri-for_hfx
H
O
C
%endblock species_swri-for_hfx
```

if your SWRI was called `for_hfx` and your system is composed of species H, O and C.

3.4 Advanced SWRI options

`swri_verbose (logical)` – set this to `T` to get detailed information on matrixblock I/O. Useful when you want to know where ONETEP is looking for matrixblock files and whether each file was succesfully loaded or not. This is output from all MPI ranks, so can make the output cluttered. Default: `F`.

`swri_cheb_batchsize (integer)` – sets the size of the batches in which SWs are processed in the calculation of the metric matrix. For the 3Dc scheme, the default is 12. Increasing this value can improve efficiency (by better balancing threads), but will increase memory load. Keep this divisible by the number of OMP threads for best performance. For the 2Dn-1Da scheme, batching has little benefit and can lead to significant load imbalance across MPI processes for larger systems. Thus, the default for 2Dn-1Da is the number of SWs in the auxiliary basis set. For both schemes, if this value is set larger than the number of SWs in the auxiliary basis set, it will be capped accordingly.

`swri_assembly_prefix (string)` – sets the prefix for the matrixblock files that are assembled into the metric matrix. The default is the rootname of your ONETEP input. Adjusting this can be useful if you keep a large number of matrixblock files in one directory and have multiple calculations, in different directories, using these matrixblock files.

`swri_proximity_sort_point (string of three values in bohr)` – metric matrix blocks are evaluated in order, with blocks between atoms closest to a predefined point done first. This is useful if you have a giant SWRI calculation (say for a solute and a few solvation shells) and would like other calculations to start using first matrix blocks as soon as possible (e.g. for calculations on just the solute). Using this keyword you can choose the point for sorting the atomblocks. The default is `0.0 0.0 0.0`. A unit of bohr is implicitly added (do not specify it).

`swri_swop_smoothing, swri_overlap_indirect, swri_improve_inverse` – these are experimental features, do not use these.

4 Hartree-Fock exchange

Now that you have SWRI set up, a basic HFx (or hybrid functional) calculation should be simple to set up. The following three keywords are mandatory and do not provide defaults:

`hfx_use_ri (string)` – tells HFx which SWRI to use. Specify the name used in the SWRI block, e.g. `hfx_use_ri for_hfx`.

`hfx_max_l (integer)` – specifies the maximum angular momentum in the SW basis. In most scenarios this will be equal to $l_{\max}$ that you specified in the SWRI block. Read the description of $l_{\max}$ (Sec. 3.1) to understand the meaning of this parameter. You can use a *lower* value than the one specified in the SWRI block if you want to use only a subset of the SW basis set (e.g. for benchmarking, or doing DMA with a lower $l_{\max}$ than you use for HFx), but not for HFx (where you must use the same value that you used in the SWRI block).

`hfx_max_q (integer)` – specifies the number of Bessel functions in the SW basis for each angular momentum channel. In most scenarios this will be equal to $q_{\max}$ that you specified in the SWRI block. Read the description of $q_{\max}$ (Sec. 3.1) to understand the meaning of this parameter. You can use a *lower* value than the one specified in the SWRI block if you want to use only a subset of the SW basis set (e.g. for benchmarking, or doing DMA with a lower $q_{\max}$ than you use for HFx), but not for HFx (where you must use the same value that you used in the SWRI block).

With the above set up, the last step is to choose a suitable functional through `xc_functional`. The following hybrid functionals use HFx: `B1LYP`, `B1PW91`, `B3LYP`, `B3PW91`, `PBE0`, `X3LYP`. For a pure Hartree-Fock calculation use `HF`.

The following two keywords might be handy:
`hfx_cutoff (physical)` – specifies the distance-based cutoff for all HFx interactions. The default is 1000 bohr, which effectively corresponds to no truncation. In the absence of truncation ONETEP's HFx implementation scales as $\mathcal{O}(N^2)$, so you are advised to use HFx truncation even if you do not use density kernel truncation. Exchange interactions are rather short-ranged, and for systems with a band-gap it should be safe to truncate them at $20\,a_0$. See [1, Figs. 19, 20] for more details. Do not use a value smaller than twice the NGWF radius.

`hfx_metric (string)` – selects the metric actually used for HFx calculations. The default is `electrostatic`. The other option is `overlap`. The appropriate metric matrix must have been included at the SWRI stage (`V` or `O`, respectively).

Other HFx-related keywords (`hfx_nlpp_for_exchange`, `hfx_read_xmatrix` and `hfx_write_xmatrix`) correspond to experimental features and should not be used.

4.1 Example

The following is a bare-bones example for a reasonably good-quality HFx calculation on a slightly distorted water molecule. That should converge in 11 NGWF iterations within 1.5 minute on a desktop machine (2 MPI ranks, 4 OMP threads each), requiring about 4 GiB of RAM.

```
xc_functional B3LYP
cutoff_energy 800 eV

%block swri
  for_hfx 3 10 V 10 10 WE
%endblock swri

%block species_swri-for_hfx
O
H
%endblock species_swri-for_hfx

hfx_use_ri for_hfx
hfx_max_l 3
hfx_max_q 10

%block lattice_cart
  25.00     0.00     0.00
   0.00    25.00     0.00
   0.00     0.00    25.00
%endblock lattice_cart

%block positions_abs
ang
O 5.79564200 7.40742600 6.63194300
H 5.19938100 8.05407400 6.24141400
H 5.16429100 6.74016800 6.88482600
%endblock positions_abs

%block species
O O 8 4 8.0
H H 1 1 8.0
%endblock species

%block species_pot
O "oxygen.recpot"
H "hydrogen.recpot"
%endblock species_pot
```

5 DMA

DMA (Distributed Multipole Analysis) is a technique for partitioning charge density into single-atom contributions and finding a set of point multipoles that most accurately represent this charge density. The point multipoles are usually, although not universally, atom-centered – this is the case in ONETEP. DMA was proposed by Rein [3] and has been pioneered and popularised by Stone [4] and Alderton [5]. It is typically performed in a Gaussian basis set [6,7], but ONETEP uses a version adapted to the NGWF basis. More details on our approach can be found in Refs. [8,9].

DMA in ONETEP first uses SWRI (cf. earlier Sections) to expand NGWF-NGWF overlaps (not exactly atom-pair densities, because there is no density kernel there) in an auxiliary SW basis set. Depending on the metric, this density fitting will strive to either minimise the difference in electronic density between the original density and the fit (for the overlap metric), or the electrostatic energy of the difference in densities interacting with itself (for the electrostatic metric). The use of electrostatic metric is preferred. Once the NGWF-NGWF overlaps are expressed in an SW basis, owing to certain properties of SWs and to the fact that in ONETEP they are chosen to be atom-centered, it becomes easy to find atom-centered point multipoles that yield the (exactly) equivalent potential. This stage is termed spherical wave expansion (SWX) and its result are atom-centered point multipoles that are the best fit to the original electronic density (under the assumed metric). Apart from calculating electronic multipoles, ONETEP's DMA also calculates total (electronic + ionic core) atom-centered multipoles. Also calculated are the total multipoles of the system (e.g.. the molecular dipole or quadrupole), suitably averaged over all the atoms that were part of the SWRI. For non-neutral molecules the value of the dipole depends on the point where it is calculated (similarly for higher multipoles), and so the total multipoles are calculated *at a reference point* of your choosing.

DMA in ONETEP is performed in two contexts. The most common is during a `task properties` calculation ("properties-DMA"). The other use of DMA is in QM/MM calculations using ONETEP and TINKER (TINKTEP approach, cf. [8], "polemb-DMA"). Some DMA keywords pertain to both contexts, and some pertain only to one of them – this will be carefully highlighted. It is possible to mix the two to a reasonable degree (i.e. to perform QM/MM with one set of DMA parameters, and properties-DMA at the end of the run, with another set of parameters). By "reasonable degree" I mean that some of the parameters are shared.

5.1 DMA: minimal set-up

To use DMA, first set up SWRI (cf. earlier Sections). Now that you have SWRI set up, a basic calculation using DMA should be simple to set up. First, specify `dma_calculate T` to enable DMA, as it is off by default. Once you've done that, the following keywords **become mandatory**:

`dma_use_ri (string)` – tells DMA which SWRI to use. Specify the name used in the corresponding SWRI block, e.g. `dma_use_ri for_dma`.

`dma_max_l (integer)` – specifies the maximum angular momentum in the SW basis used in DMA. In most scenarios this will be equal to $l_{\max}$ that you specified in the SWRI block. Read the description of $l_{\max}$ (Sec. 3.1) to understand the meaning of this parameter. You can use a lower value than the one specified in the SWRI block if you want to use only a subset of the SW basis set (e.g. for benchmarking). This keyword only affects properties-DMA, the equivalent for polemb-DMA is `pol_emb_dma_max_l`. This keyword needs to be specified even if you do not plan to use properties-DMA (in that case, specify 0). If you only care about atom-centered charges, specify 0. If you care about atom-centered charges and dipoles, specify 1. If you care about atom-centered charges, dipoles and quadrupoles, specify 2.

`dma_max_q (integer)` – specifies the number of Bessel functions in the SW basis for each angular momentum channel to be used in DMA. In most scenarios this will be equal to $q_{\max}$ that you specified in the SWRI block. Read the description of $q_{\max}$ (Sec. 3.1) to understand the meaning of this parameter. You can use a lower value than the one specified in the SWRI block if you want to use only a subset of the SW basis set (e.g. for benchmarking). This keyword only affects properties-DMA, the equivalent for polemb-DMA is `pol_emb_dma_max_q`. This keyword needs to be specified even if you do not plan to use properties-DMA (in that case, specify 0).

5.2 Non-mandatory keywords affecting both properties-DMA and polemb-DMA

`dma_metric (string)` – selects the metric used for DMA calculations. The current default is `electrostatic`. The other option is `overlap`. The appropriate metric matrix must have been included at the SWRI stage (`V` or `O`, respectively). This keyword affects both properties-DMA and polemb-DMA.

`dma_bessel_averaging (boolean)` – specifies whether all DMA-based multipoles are to be averaged over an even-odd pair of $q_{\max}$. Multipoles obtained with DMA display an oscillatory behaviour when plotted as a function of $q_{\max}$. This has to do with how the Bessel functions sample the radial profile of NGWFs. In essence, for all even $q_{\max}$ a particular multipole will be overestimated, while for all odd $q_{\max}$ the same multipole will be underestimated (or the other way round). Other multipoles will be affected similarly (except in reverse) to compensate. This oscillatory behaviour decays as the quality of the SW basis is increased, but the decay is slow. Much more stable multipoles are obtained by averaging the results of two SWX runs – one with the $q_{\max}$ the user specified in `dma_max_q`, and one with $q_{\max}$ that is less by one. This even-odd averaging can be performed automatically by specifying `dma_bessel_averaging T`, and this is done by default. When this option is enabled, output files include multipoles obtained with both SWX qualities, followed by the average, except for the `.dma_multipoles_gdma_like.txt` file, which will contain only the final, averaged multipoles. There is no extra effort associated with this option at the SWRI stage, and the effort of the SWX stage (which is usually much, much lower) is practically doubled (two separate SWXs have to be performed). This keyword affects both properties-DMA and polemb-DMA.

`dma_scale_charge (boolean)` – specifies DMA charge-scaling is to be performed (default) or not. This an important option. The multipoles obtained with DMA are always approximate. The total DMA monopole (charge) will be close to, but not exactly equal to, the total charge of the system (or its subset, if DMA's SWRI did not encompass all atoms). This means that the total DMA monopole of a nominally neutral system will not be exactly zero, but typically a very small fraction of an electron. This is inconvenient, because it formally breaks the translational invariance of the total dipole, which begins to depend, very slightly, on the reference point where it is calculated. The easiest workaround is to scale, *a posteriori*, the DMA monopole by the ratio of expected charge to the obtained DMA charge. This scaling is factor will be very close to zero (e.g. 0.9998), unless your SW basis set is very crude (single-digit $q_{\max}$, etc.). The "expected" (electronic) charge either obtained automatically (by default), or can be specified manually using `dma_target_num_val_elec`. When not specified manually, the expected electronic charge is determined as follows. If DMA's SWRI encompasses all atoms ("full-system DMA"), it is equal to the total number of valence electrons in the system (obtained from $\mathrm{Tr}\,[\mathbb{KS}]$). If DMA's SWRI does not encompass all atoms ("subsystem DMA"), Mulliken analysis is performed every time charge-scaling needs to be done (essentially at every LNV step, or twice per LNV step when Bessel averaging is used), and Mulliken charges of all atoms within DMA's SWRI are summed to obtain the expected electronic charge. Using DMA charge-scaling is recommended (hence it's on by default), but care must be taken when using it with polemb-DMA (there are no issues with properties-DMA). The following issues and limitations arise. (1) In polemb-DMA the DMA multipoles enter LNV and NGWF gradient expressions. The quantity $\mathrm{Tr}\,[\mathbb{KS}]$ is not strictly a constant, and has non-zero DKN and NGWF derivatives, leading to additional terms in LNV and NGWF gradients when charge-scaling is used. These extra terms have been implemented for LNV gradients, but *not* for NGWF gradients, where they become really hairy (these are under development). Hence the threefold combination of polemb-DMA, charge-scaling and NGWF optimisation is not permitted (will refuse to run). (2) The threefold combination of polemb-DMA, charge-scaling and `dma_target_num_val_elec` is not permitted (will refuse to run), regardless of whether NGWF optimisation is used or not. This is because the expected number of electrons becomes constant (user-specified value) in this scenario, which is incompatible with the charge-scaling corrections accounting for $\mathrm{Tr}\,[\mathbb{KS}]$. Long story short: use DMA charge-scaling for properties-DMA, but not for polemb-DMA. This keyword affects both properties-DMA and polemb-DMA.

`dma_target_num_val_elec (integer)` – specifies the expected number of valence electrons for DMA. This keyword should only be used when DMA charge-scaling is in effect (see above), and only if the automatic determination of the expected number of electrons (see above) in the part of your system seen by DMA is not satisfactory. The default is for this keyword to be omitted. This keyword affects both properties-DMA and polemb-DMA.

`polarisation_simcell_refpt (real real real)`. The default is `0.0 0.0 0.0`. Specifies the reference point in the simulation cell (in bohr) at which total DMA multipoles are calculated. This is mostly useful if your system (strictly speaking: your DMA subsystem) is not charge-neutral and you are interested in

the value of the total dipole. When the system is non-neutral, the total dipole is not translation invariant, and a reference point for calculating it needs to be specified. This keyword specifies this reference. Also note that when a simcell full-density polarisation calculation is performed (via `task properties` and `polarisation_simcell_calculate`), this keyword also adjusts this calculation's reference point. This keyword affects both properties-DMA and polemb-DMA.

`dma_precise_gdma_output (boolean)`. The default is on (`T`). One of the files output by DMA is the `.dma_multipoles_gdma_like.txt` file, which is formatted as to be compatible with the output generated by Stone's GDMA program. This output can be directly used by other programs expecting input in this format. The original GDMA format is fixed-form, meaning the precision of the output multipoles is rather restricted by the number of digits that can be output. In polemb-DMA mode this precision is insufficient to accurately drive the MM calculation performed by an external program (TINKER). Specifying `dma_precise_gdma_output T` instructs ONETEP to output multipoles with the additional necessary precision, but breaks strict compatibility with the GDMA format. If the external program you use to parse the GDMA file is aware of that (e.g. TINKER can be suitably patched), this is fine. If you have no control over the external program and need ONETEP to adhere strictly to the GDMA format, use `dma_precise_gdma_output F`.

5.2.1 Expert, non-mandatory keywords affecting both properties-DMA and polemb-DMA

Just don't. These are used for experimental purposes, particularly in QM/MM.

`dma_multipole_scaling (real)` – causes all DMA multipoles to be scaled by a constant. This affects both the output multipoles and the multipoles used internally in polemb expressions. Whenever necessary (during charge-scaling, in gradients) this scaling is temporarily undone internally for consistency. This keyword affects both properties-DMA and polemb-DMA.

`dma_dipole_scaling (real)` – causes all DMA dipoles to be scaled by a constant. This affects both the output dipoles and the dipoles used internally in polemb expressions. Whenever necessary (essentially in gradients) this scaling is temporarily undone internally for consistency. This keyword affects both properties-DMA and polemb-DMA.

`dma_quadrupole_scaling (real)` – causes all DMA quadrupoles to be scaled by a constant. This affects both the output quadrupoles and the quadrupoles used internally in polemb expressions. Whenever necessary (essentially in gradients) this scaling is temporarily undone internally for consistency. This keyword affects both properties-DMA and polemb-DMA.

5.3 Non-mandatory keywords affecting only properties-DMA

`dma_output_potential (boolean)`. The default is off (`F`). When turned on (`T`), during properties-DMA the electrostatic potential due to all DMA *electronic* multipoles is calculated on the $z = 0$ and $z = z_{\max}$ faces of the simulation cell (on all fine-grid points lying on those faces). This potential is output to text files. This is useful for assessing the quality of the DMA approximation to the full, distributed charge density. If DMA's SWRI does not span the entire system, the output potential is only due to those atoms included in DMA's SWRI. This keyword affects only properties-DMA.

`dma_output_potential_reference (boolean)`. The default is off (`F`). When turned on (`T`) *and* `dma_output_potential` *is also on*, during properties-DMA the reference electrostatic potential due to the full, distributed *electronic* density is calculated on the $z = 0$ and $z = z_{\max}$ faces of the simulation cell (on all fine-grid points lying on those faces). This potential is output to text files. This is useful to obtain a reference for assessing the quality of the DMA approximation to the full, distributed charge density (see above). Regardless of whether DMA's SWRI spans the entire system or not, the output potential is due to *all* electrons in the system. Thus, the two sets of potentials are only comparable in full-system DMA. The reference potential is calculated by a pointwise integration over the entire volume (fine-grid) *for every point on the two faces*, which is a time-consuming process, so use sparingly. This keyword affects only properties-DMA.

5.4 Non-mandatory keywords affecting only polemb-DMA

Refer to the separate documentation for polarisable embedding in ONETEP.

5.5 Example

The following is a bare-bones example for a reasonably good-quality properties-DMA calculation on a slightly distorted water molecule. That should converge in 11 NGWF iterations within 1 minute on a desktop machine (2 MPI ranks, 4 OMP threads each), requiring about 3 GB of peak RAM.

```
xc_functional PBE
cutoff_energy 800 eV

do_properties T

%block swri
  for_dma 2 14 V 12 12 WE
%endblock swri

%block species_swri-for_dma
O
H
%endblock species_swri-for_dma

dma_calculate T
dma_use_ri for_dma
dma_metric electrostatic
dma_max_l 2
dma_max_q 14

dma_scale_charge T
dma_bessel_averaging T

%block lattice_cart
  25.00     0.00     0.00
   0.00    25.00     0.00
   0.00     0.00    25.00
%endblock lattice_cart

%block positions_abs
ang
O 5.79564200 7.40742600 6.63194300
H 5.19938100 8.05407400 6.24141400
H 5.16429100 6.74016800 6.88482600
%endblock positions_abs

%block species
O O 8 4 8.0
H H 1 1 8.0
%endblock species

%block species_pot
O "oxygen.recpot"
H "hydrogen.recpot"
%endblock species_pot
```

Expected results:

Table 1: Dipole moment of distorted water molecule. Comparison of accuracy: full density vs. DMA point multipoles.

Description	Dipole (au)	Dipole (debye)
Full density (cores + NGWFs)	0.7564	**1.9226**
DMA (point multipoles, $q_{\max}$=14)	0.7477	1.9005
DMA (point multipoles, $q_{\max}$=13)	0.7680	1.9519
DMA (point multipoles, Bessel-averaged)	0.7578	**1.9261**

6 Advanced options

6.1 Making Hartree-Fock exchange faster or less memory-hungry

Hartree-Fock exchange is not fast, although we've made great improvements in v5.3.4.0. For small systems (< 200 atoms), with a bit of luck, it will be an order of magnitude slower than GGA calculations. For large systems (≈ 1000 atoms) expect it to be two orders of magnitude slower.

The main way to improve performance is by using more RAM – this is because there are plenty of opportunities for caching some results that would otherwise have to be recomputed. If HFx was to cache everything, it would quickly exhaust all available RAM, even on well-equipped machines. Therefore, there are limits in place for each of the caches. These limits are expressed in MiB (1048576 bytes) and are per MPI rank.

Remember that OMP threads can share memory, while MPI ranks cannot. This means that the key to obtaining high performance with HFx is to use as many OMP threads as possible. In most HPC settings this will mean using only 2 MPI ranks per node (one MPI rank per NUMA region, most HPC nodes have two NUMA regions). For example on Iridis5, with 40 CPU cores on each node, best performance is obtained by using 2 MPI ranks, with 20 OMP threads each, on every node. This is in contrast to non-HFx calculations, which typically achieve peak performance for 4-5 OMP threads. HFx is well-optimised for high thread counts. By reducing the number of MPI ranks, you allow each rank to use more RAM. This is the key to success. Don't worry about the drop in performance of the non-HFx part, it will be dwarfed by the gain in HFx efficiency.

The easiest way to control how much RAM HFx can use is via the parameter `hfx_memory_limit`. The default value is 4096, meaning HFx will not ask for more than 4 GiB of RAM per MPI rank. This is *in addition* to any memory use from the rest of ONETEP. If you can spare more RAM, definitely tell this to the HFx engine by saying e.g.

```
hfx_memory_limit 16384   !  I have 16 GiB per MPI rank to spare.
```

The HFx engine will automatically distribute this RAM across the three main caches. Or, more specifically, it will first consume the amount of RAM absolutely needed for core HFx functionality, and *then* distribute the rest to the three caches. You will get a banner informing you about how much memory went into satisfying the minimum requirements:

```
+-----------------------------------------------------+
|  HFx TEFCI engine: minimum requirements             |
|  Estimated memory requirement per MPI rank          |
+-----------------------------------------------------+
|  Radial Bessel lookup                :   30.52 MB   |
|  Dd NGWFs hash table                 :    1.66 MB   |
|  All remote NGWFs hash table         :   37.38 MB   |
|  dlists hash table                   :    6.53 MB   |
|  coeffs hash table (estimate)        :   26.93 MB   |
|  V metric matrix hash table          :  377.59 MB   |
|  f auxiliary term                    :  131.78 MB   |
|  P term in NGWF gradient             :  131.78 MB   |
|  Q term in NGWF gradient             :  238.88 MB   |
|  My kets in NGWF grad. (estimate)    :   78.09 MB   |
|  Local kets in NGWF grad. (estim.)   :   43.20 MB   |
|  K^{CD} hash table                   :    3.88 MB   |
|  K^{AB} hash table                   :    3.60 MB   |
|  tcK^A_B hash table                  :    3.60 MB   |
|  tcK^B_A hash table                  :    3.60 MB   |
+-----------------------------------------------------+
|  Estimated peak total per MPI rank   :    1.09 GB   |
+-----------------------------------------------------+
```

In the event that the memory limit specified with `hfx_memory_limit` is below even the minimum requirement (1.09 GB in the example above), you will get an error message explaining how much more RAM you would need to continue. Be aware of two things: (1) calculations with not much (or no) memory above the minimum requirement will be very slow, (2) the above is only an estimate. Under some circumstances HFx may consume slightly more memory, but not much. If you run out of memory, it is usually the NGWF gradient calculation (its non-HFx part) that breaks the camel's back.

Another banner informs you about how the remaining RAM is divided across the three caches ("hash tables"). It may look like this:

```
HFx: - Adjusting cache sizes according to weights: 0.6250, 0.3125, 0.0625.
+-----------------------------------------------------+
|  HFx TEFCI engine: user-adjustable requirements     |
|  Estimated memory requirement per MPI rank          |
+-----------------------------------------------------+
|  SWOP hash table                      :    3.61 GB  |
|  Expansions hash table                :    1.81 GB  |
|  AD NGWF products hash table          :  369.00 MB  |
+-----------------------------------------------------+
|  Estimated peak total per MPI rank    :    5.77 GB  |
+-----------------------------------------------------+
HFx: - Peak memory use capped at 6998.2 MB per MPI rank.
```

Here the user specified `hfx_memory_limit 7000` and HFx distributed the remaining 5.77 GB across the three caches with a default set of weights that is 10:5:1 ($= \frac{10}{16} : \frac{5}{16} : \frac{1}{16} = 0.6250 : 0.3125 : 0.0625$). This is an empirically determined near-optimal default for valence calculations. For conduction calculations the default is to give all remaining RAM to the SWOP hash table, because in conduction calculations expansions are never re-used and the number of NGWF products is so large, that it's faster to give up on storing them entirely.

The three caches store, respectively:

- Spherical waves or potentials thereof ("SWOPs"). Generating SWOPs is typically the main bottleneck of any HFx calculation, and increasing the size of this cache will lead to significant improvements, with returns diminishing after hit ratios exceed 90-95%.
- Spherical wave expansions (potentials of linear combinations of spherical waves on a centre). These can help performance too, but their size quickly becomes unwieldy.
- NGWF products. Less useful than the above, but cheap to store, except in conduction calculations.

In general, it is best to rely on the default division and to specify only `hfx_memory_limit`. However, if you feel you can do better, you can manually set the maximum for any number of caches, using the directives `cache_limit_for_swops`, `cache_limit_for_expansions`, `cache_limit_for_prods`. This might be useful if you specifically want to disable one or more of the caches (by specifying 0). Remember that `hfx_memory_limit` is still in effect by default, even if you do not specify it, and it will interact with the above. If you want to turn off the automatic balancing of memory limits, specify `hfx_memory_limit -1`. Once you do this, the specified cache limits will be used (with a default of 1024). Finally, if you keep the automated balancing, `hfx_memory_weights`, which accepts three real numbers, can be used to set the desired weights, if you are not satisfied with the default. The weights do not need to add to 1, they will be automatically rescaled. They cannot all be zero, but some of them can be zero (which then turns off the associated cache).

The utilisation of each cache is reported at some stage of the calculation (assuming `hfx_output_detail` is at `NORMAL` or higher). You will see banners like this:

```
+-------------------------------------------------------------------------------+
|    MPI |                           |              |           |              |
|   rank |       SWOP cache capacity | SWOPs needed | Cacheable |    Hit ratio |
+-------------------------------------------------------------------------------+
|      0 |       3691 MiB (24189 el) |    109140 el |    22.16% |       58.70% |
|      1 |       3691 MiB (24189 el) |    109140 el |    22.16% |       58.85% |
+-------------------------------------------------------------------------------+
```

This is a breakdown over all MPI ranks (only two in this case), informing you that you devoted about 3.7 GB per MPI rank to the SWOP cache, which enables caching 24189 elements, whereas 109140 elements could be stored, if you had more RAM. You were thus able to cache about 22% of all elements, but because HFx stores the most reusable ones first, the cache hit ratio will be about 59% – that is, in 59% of cases when HFx will be looking for a SWOP, it will find it in the cache. Different SWOPs will be needed on different nodes, hence the hit ratios are not exactly equal. Looking up SWOPs in the cache is at least an order of magnitude faster than recaculating them, so you should aim for a hit ratio of at least 90%.

Banners for the expansion and NGWF product caches will be printed after the first LNV (or EDFT) iteration (for `hfx_output_detail VERBOSE`) or after every energy evaluation (for `hfx_output_detail` at `PROLIX` or higher). They look like this:

```
HFx: +-------------------------------------------------------------------------+
HFx: |    MPI |                            Expansion cache                      |
HFx: |   rank |           hits |         misses |          total | hit ratio |
HFx: +-------------------------------------------------------------------------+
HFx: |      0 |        1982298 |       13369145 |       15351443 |   12.91 % |
HFx: |      1 |        1947916 |       13512601 |       15460517 |   12.60 % |
HFx: +-------------------------------------------------------------------------+
HFx: +-------------------------------------------------------------------------+
HFx: |    MPI |                      AD NGWF product cache                      |
HFx: |   rank |           hits |         misses |          total | hit ratio |
HFx: +-------------------------------------------------------------------------+
HFx: |      0 |        1947809 |        7499676 |        9447485 |   20.62 % |
HFx: |      1 |        1992291 |        7737078 |        9729369 |   20.48 % |
HFx: +-------------------------------------------------------------------------+
```

and show you a per-MPI-rank breakdown of how many hits and misses were recorded in accessing the cache, and what the hit ratio was. You will be able to achieve 100% only for the smallest of systems,

Changing cache limits only affects the tradeoff between RAM and CPU time, it has absolutely no effect on results, only on the time and memory it will take to arrive at them. If you are very pressed for RAM, you can set all the above cache sizes to 0. This will stop caching altogether, conserving memory, but will vastly increase run time, by a factor of several.

A simple, perhaps surprising, way of increasing performance (slightly) is by using PPDs that are not flat. By default ONETEP initialises the third dimension of a PPD to 1, making them flat. This makes sense in the absence of HFx. With HFx the book-keeping of the caching machinery will be faster when the PPDs are slightly larger. Preferably use `ppd_npoints` to make the PPDs $5 \times 5 \times 5$ or $7 \times 7 \times 7$. It might be necessary to explicitly set `psinc_spacing` and carefully choose the box dimensions. An easy solution is to choose `psinc_spacing 0.5 0.5 0.5` which corresponds to a kinetic energy cutoff of 827 eV, and to make your box dimensions divisible by $2.5\,a_0$ (for $5 \times 5 \times 5$ PPDs).

Finally, you can try omitting some terms in the expansion, if the expansion coefficients are below a certain threshold. This will affect the accuracy of your results, and so by default nothing is thrown away. This can be done via the keyword `swx_c_threshold` which takes a real number as an argument. Whenever an NGWF-pair expansion coefficient is below this value, potentials from this particular SW for this pair of NGWFs will not even be generated. This can be used in conjunction with a distance-based truncation. A value like 1E-5 will throw away maybe 2-3% of the terms. 1E-3 will throw away about 10-15% (so, little gain), and this will be enough to impair your NGWF convergence. I do not recommend changing this setting.

6.2 Other advanced options

`swx_output_detail (string)` – controls the verbosity of the spherical wave expansion. Allowed options: `BRIEF`, `NORMAL`, `VERBOSE`. Defaults to `NORMAL`. At `VERBOSE` you might feel overwhelmed by the output from all MPI ranks letting you know which atom they are working on (lines looking like "`+ A: 1`" or "`- B: 1`"). This is useful for big systems, where it takes a while to get from one LNV iteration to the next one, with no output otherwise.

hfx_output_detail (string) – controls the verbosity of Hartree Fock exchange. Allowed options: BRIEF, NORMAL, VERBOSE, PROLIX, MAXIMUM. Defaults to the value of output_detail. At VERBOSE you might feel overwhelmed by the output from all MPI ranks letting you know which atom they are working on (lines looking like "- B: 1 [0] (1)"). This is useful for big systems, where it takes a while to get from one LNV iteration to the next one, with no output otherwise. At PROLIX there is even more feedback. At MAXIMUM even the X matrix is printed, which will make your output file extremely big. This is recommended only when debugging. The recommended setting is VERBOSE.

hfx_bessel_rad_nptsx (integer) – specifies how many points are used in the radial interpolation of Bessel functions. The default is 100000 and should be sufficient. Increasing this value (perhaps to 250000 or so) improves accuracy, particularly if your simulation cell is large, but there is an associated linear memory cost (typically in tens of MB per MPI rank).

use_sph_harm_rot (logical) – Manually activate the sph_harm_rotation (spherical harmonic rotation) module (used to evaluate the metric matrix in the 2Dn-1Da scheme). In normal operation this is not necessary, since the module will be activated if it is detected that spherical harmonic rotation is required. Setting this is to false has no effect, since the option will be overridden if ONETEP detects that the module is needed, anyway.

swx_dbl_grid – experimental functionality, please do not use.

6.2.1 devel_code values

SHROT:DEBUG=[T/F]:SHROT – Activate debug mode for the sph_harm_rotation module

SHROT:UNIT_TEST=[T/F]:SHROT – Activate unit testing for the sph_harm_rotation module

7 Frequently asked questions

7.1 What hybrid functionals are available in ONETEP?

B1LYP, B1PW91, B3LYP, B3PW91, PBE0, X3LYP. For a pure Hartree-Fock calculation use HF.

7.2 How big can my system be when using HFx?

Up to 200 atoms should be a breeze on a desktop machine (64 GB RAM, 16 cores). About 500-600 atoms will be a limit for a desktop machine, but it might take a week or two. Larger systems will be off-limits because you will either run out of memory (if using >1 MPI rank), or hit sparse_mod integer overflows (if using 1 MPI rank).

On a HPC cluster (say, 640 cores) up to 1000 atoms should not be too difficult (3-4 days). Current record is 4048 atoms (640 cores, 20 days, June 2020). With significant resources (5000+ cores) you should be able to do 10000 atoms, but this has not been tried.

7.3 How do I make HFx faster?

Use as many OMP threads as possible, without crossing NUMA regions. On a typical HPC system this will mean using only 2 MPI ranks per node, and on a desktop machine – only 1 MPI rank. This will minimise the number of MPI ranks, allowing you to give much more memory to each of them. Increase hfx_memory_limit from the default value of 4096 to however much you can spare. This is the maximum RAM consumption of HFx (on top of the rest of ONETEP) per MPI rank. Here it is always the higher the better, except you don't want to run out of memory. Use ppd_npoints 5 5 5 or ppd_npoints 7 7 7 for a slight efficiency gain.

7.4 I'm running out of memory

Symptoms: you get error messages like "Killed", or "Out of memory: Kill process *nnn* (onetep.exe)" in `dmesg` output, or your system starts *thrashing* (swapping memory to HDD). What you can do:

- Reduce the number of MPI ranks per node. In the worst case scenario, undersubscribe (leave some cores idle), even to the point of having only 1 MPI rank per node.
- Reduce `hfx_memory_limit` from the default value of 4096 to something smaller.
- Reduce the quality of the SW expansion ($l_{\max}$, $q_{\max}$) (this will affect the quality of results).
- If you're running out of memory at the NGWF gradient stage, reduce `threads_num_fftboxes`. The default is equal to the number of OMP threads. Reduce it to 1. This is a component of ONETEP that consumes quite a bit of RAM irrespective of whether HFx is used or not.
- Use high-memory nodes. Many HPC facilities provide these.
- Increase the number of compute nodes used in the calculation. Some of the necessary data will be distributed across nodes, reducing per-node load. With a large number of OMP threads, you can easily use much more CPUs than you have atoms in your system.

7.5 The HFx engine does not use all the memory I asked it it to use. Why?

If you devote too much memory to one of the caches, only as much will be used as is needed to store all that is needed. Perhaps adjust `hfx_memory_weights` to give this memory to where it is needed more.

7.6 My calculation aborts with `Error in sparse_count_ss: Integer overflow in sparse matrix index detected.` What now?

Your sparse matrices are too large, overflowing integer indices in ONETEP's sparse matrix machinery. Essentially you are trying to run a calculations with too many atoms on too few MPI ranks. Increase the number of MPI ranks or decrease `hfx_cutoff`. The latter will impact results.

7.7 My calculation crashes with a `SIGSEGV` and the last line of output is "`KSKS matrix filling:`". What now?

Same as above, except the integer overflow has not been detected.

7.8 My calculation aborts with `Exchange matrix not deemed accurate enough for a stable calculation`. What now?

One of the approximations broke down. You turned the "speed vs. accuracy" knob too far towards "speed".

- Is your SW expansion quality too low? The minimum reasonable quality is about $l_{\max} = 2$, $q_{\max} = 8$, see [1, Fig. 8]. For some systems this might not be enough, particularly in pure Hartree-Fock calculations (`xc_functional HF`). Try $l_{\max} = 3$, $q_{\max} = 10$.
- Is your KE cutoff too low? In general, don't go below 800 eV.
- Is your Chebyshev interpolation quality too low? N_{i} should be at least 10, preferably 12. N_{o} should also be at least 10, preferably 12.
- Is the number of points in the radial Bessel interpolation (`hfx_bessel_rad_nptsx`) too low? Don't go below 100000. For larger simulation cells you might want to use a higher value (say, 250000).

7.9 Can I do conduction calculations with HFx?

Yes!

Just make sure all NGWF radii are equal for all species and across `species` and `species_cond` blocks. You may reuse the metric matrices between the valence and conduction calculation. Have a lot of CPU power available. It shouldn't be too difficult up to 500 atoms, then difficulty ramps up. 1000+ atoms will require considerable resources ($\approx$ 1000+ cores) and patience (weeks). Current record is 1108 atoms (June 2020).

7.10 Can I do LR-TDDFT calculations with HFx?

Yes, but this is at an experimental stage at the moment.

8 Further questions?

General questions should be directed to Jacek Dziedzic, `J.Dziedzic[-at-]soton.ac.uk.`

Questions relating to the 2Dn-1Da metric matrix evaluation scheme or to hybrid LR-TDDFT should be directed at James C. Womack, `J.C.Womack[-at-]bristol.ac.uk`.

References

[1] J. Dziedzic, Q. Hill and C.-K. Skylaris, J. Chem. Phys. **139** 214103 (2013).

[2] N.D.M. Hine, J. Dziedzic, P.D. Haynes and C.-K. Skylaris, J. Chem. Phys. **135** 204103 (2011).

[3] R. Rein, On Physical Properties and Interactions of Polyatomic Molecules: With Application to Molecular Recognition in Biology, in Advances in Quantum Chemistry,ed. Per-Olov Löwdin, Academic Press, 1973.

[4] A.J. Stone, Chemical Physics Letters **2** 83 233-239 (1981).

[5] A.J. Stone and M. Alderton, Molecular Physics **5** 56 (1985).

[6] A.J. Stone, GDMA: distributed multipoles from Gaussian98 wavefunctions (technical report), University of Cambridge (1998).

[7] A.J. Stone, Journal of Chemical Theory and Computation **6** 1128-1132 (2005).

[8] J. Dziedzic, Y. Mao, Y. Shao, J. Ponder, T. Head-Gordon, M. Head-Gordon and C.-K. Skylaris, J. Chem. Phys. **145** 12 124106 (2016).

[9] V. Vitale, J. Dziedzic, S.M.-M. Dubois, H. Fangohr and C.-K. Skylaris, Journal of Chemical Theory and Computation **11** 7 3321-3332 (2015).