Density mixing (Kernel-DIIS) in ONETEP

Álvaro Ruiz Serrano

August 27, 2013

Contents

1 Basic principles

This manual describes how to run density mixing (kernel DIIS) calculations in ONETEP. Currently, kernel DIIS is an alternative to the LNV density kernel optimisation in the ONETEP inner loop (where the NGWFs are fixed). Please note that:

- The current code is experimental and might be unstable.
- Kernel DIIS currently can only perform calculations on insulators, where the constraint of idempotency of the density matrix holds (i.e., the Kohn-Sham occupancies are integers, 1 for the valence states and 0 for the conductions states.)
- Kernel DIIS is cubic-scaling, always, even if the density matrix is truncated and localised.

- Use only in the rare occasions where LNV optimisation might not be optimal.

Density mixing is a type of self-consistent cycle used to minimise the total energy of the system. Currently, there are two types of mixing implemented in ONETEP: density kernel mixing or Hamiltonian mixing. In both cases, Hamiltonian diagonalisation is required, which makes the method cubic-scaling. The current implementation includes the linear mixing, ODA mixing (1; 2), Pulay mixing (3), LiSTi (4) and LiSTb (5) mixing schemes. Pulay, LiSTi and LiSTb offer the best performance of all, but they must be used in conjunction with linear mixing in order to obtain numerical stability.

2 Compilation

By default, ONETEP is linked against the LAPACK library (6) for linear algebra. The LAPACK eigensolver DSYGVX (7), can only be executed in one CPU at a time. Therefore, kernel DIIS calculations with LAPACK are limited to small systems (a few tens of atoms). Calculations on large systems are possible if, instead, ONETEP is linked against SCALAPACK library (8) during compilation time. The SCALAPACK eigensolver, PDSYGVX (9), can be run in parallel using many CPUs simultaneously. Moreover, SCALAPACK can distribute the storage of dense matrices across many CPUs, thus allowing to increase the total memory allocated to a given calculation in a systematic manner, simply by requesting more processors. For the compilation against SCALAPACK to take effect, the flag `-DSCALAPACK` must be specified during the compilation of ONETEP.

3 Commands for the inner loop

3.1 Mixing schemes

The keyword

`kernel_diis_scheme:` *string*
[String, default `kernel_diis_scheme: NONE`].

is used to select the mixing method during the kernel-DIIS loop. By default, this keyword takes the value "`NONE`", which disables kernel DIIS and tells the program to proceed with the LNV optimisation. The following options are available:

- `kernel_diis_scheme: DKN_LINEAR`
  linear mixing of density kernels. The new input density kernel is built from the *in* and *out* density kernels of the current iteration as $K_i n^{n+1} = (1-\lambda)K^n_{in} + \lambda K^n_{out}$.

- `kernel_diis_scheme: HAM_LINEAR`
  linear mixing of Hamiltonians. The new input Hamiltonian is built from the in and *out* Hamiltonians of the current iteration as $H_i n^{n+1} = (1-\lambda)H^n_{in} + \lambda H^n_{out}$.

- `kernel_diis_scheme: DKN_PULAY`
  Pulay mixing of density kernels (see Ref. (3)). The new input density kernel is built as a linear combination of the *output* density kernels of the N_{mix} previous iterations as $K_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m K_{out}^m$. Pulay mixing requires the storage of N_{mix} matrices.

- `kernel_diis_scheme: HAM_PULAY`
  Pulay mixing of Hamiltonians (see Ref. (3)). The new input Hamiltonian is built as a linear combination of the *output* Hamiltonians of the N_{mix} previous iterations as $H_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m H_{out}^m$. Pulay mixing requires the storage of N_{mix} matrices.

- `kernel_diis_scheme: DKN_LISTI`
  LiSTi mixing of density kernels (see Ref. (4)). The new input density kernel is built as a linear combination of the *output* density kernels of the N_{mix} previous iterations as $K_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m K_{out}^m$. LiSTi mixing requires the storage of $4 \times N_{mix}$ matrices.

- `kernel_diis_scheme: HAM_LISTI`
  LiSTi mixing of Hamiltonians (see Ref. (4)). The new input Hamiltonian is built as a linear combination of the *output* Hamiltonians of the N_{mix} previous iterations as $H_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m H_{out}^m$. LiSTi requires the storage of $4 \times N_{mix}$ matrices.

- `kernel_diis_scheme: DKN_LISTB`
  LiSTb mixing of density kernels (see Ref. (5)). The new input density kernel is built as a linear combination of the *output* density kernels of the N_{mix} previous iterations as $K_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m K_{out}^m$. LiSTb mixing requires the storage of $4 \times N_{mix}$ matrices.

- `kernel_diis_scheme: HAM_LISTB`
  LiSTb mixing of Hamiltonians (see Ref. (5)). The new input Hamiltonian is built as a linear combination of the *output* Hamiltonians of the N_{mix} previous iterations as $H_i n^{n+1} = \sum_{m=n-N_mix}^{n} \lambda_m H_{out}^m$. LiSTb mixing requires the storage of $4 \times N_{mix}$ matrices.

- `kernel_diis_scheme: DIAG`
  Hamiltonian diagonalisation only - no mixing takes place. Strongly NOT recommended.

**** NOTE**: linear mixing can be used simultaneously with Pulay, LiSTi or LiSTb mixing to create a history of density kernels/Hamiltonians with optimal numerical properties. See the keywords `kernel_diis_coeff` and `kernel_diis_linear_iter` in Sec. 3.2 for more information.

3.2 Basic setup: controlling the mix

- `kernel_diis_coeff: x`
  [Real, default `kernel_diis_coeff: 0.1`].

Linear-mixing λ coefficient. Must be in the range $[0, 1]$. If set to a negative value, the ODA method (see Refs. (1; 2)) will automatically calculate the optimal value of λ. The value of `kernel_diis_coeff` can be made arbitrarily small in order to attain numerical stability. However, this can make convergence slow. A small value can be used in order to create a history of matrices before Pulay, LiSTi or LiSTb mixing. A value close to 1 will make the calculation potentially unstable.

- `kernel_diis_linear_iter: n`
  [Integer, default `kernel_diis_linear_iter: 5`].
  Number of linear mixing iterations before Pulay, LiSTi or LiSTb mixing. This keyword will create and store a history of `n` previous matrices generated by linear mixing before Pulay, LiSTi or LiSTb mixing begin. Required for large systems in order to achieve stability.

- `kernel_diis_size: n`
  [Integer, default `kernel_diis_size: 10`].
  Number of matrices (N_{mix}) to be mixed with the Pulay, LiSTi or LiSTb schemes.

- `kernel_diis_maxit: n`
  [Integer, default `kernel_diis_maxit: 25`].
  Maximum number of iterations during the density mixing inner loop.

3.3 Tolerance thresholds

- `kernel_diis_threshold: x`.
  [Real, default `kernel_diis_threshold: 1.0e-9`].
  Numerical convergence threshold for the DIIS inner loop. Use in conjunction with `kernel_diis_conv_criteria`.

- `kernel_diis_conv_criteria: string`.
  [String, default `kernel_diis_conv_criteria: 1000`]. Select the convergence criteria for the DIIS inner loop.`kernel_diis_conv_criteria` takes a string value 4 characters long. Each position acts as a logical switch, and can only take the values "`1`" (on) or "`0`" (off). The order is the following:

  - Position 1: residual $|K_{out} - K_{in}|$, if density kernel mixing, or $|H_{out} - H_{in}|$, if Hamiltonian mixing.
  - Position 2: Hamiltonian-density kernel commutator.
  - Position 3: band-gap variation between two consecutive iterations (in Hartree).
  - Position 4: total energy variation between two consecutive iterations (in Hartree).

  For example, `kernel_diis_conv_thres: 1101` will enable criteria 1,2 and 4 and disable criterion 3.

3.4 Advanced setup: level shifter

Extra stability can sometimes be achieved if the conduction energy values are artificially increased. This technique is known as level-shifting. See Ref. (10) for further details.

- `kernel_diis_lshift: x units.`
  [Real physical, default `kernel_diis_lshift: 1.0 Hartree`].
  Energy shift of the conduction bands.

- `kernel_diis_ls_iter: n.`
  [Integer, default `kernel_diis_ls_iter: 0`].
  Total number of DIIS iterations with level-shifting enabled.

4 Commands for the outer loop

The standard ONETEP commands for NGWF optimisation apply.

5 Restarting a kernel DIIS calculation

- `write_denskern: T/F.`
  [Boolean, default `write_denskern: F`].
  Save the last density matrix on a file.

- `read_denskern: T/F.`
  [Boolean, default `read_denskern: F`].
  Read the density kernel matrix from a file, and continue the calculation from this point.

- `write_tightbox_ngwfs: T/F.`
  [Boolean, default `write_tightbox_ngwfs: T`].
  Save the last NGWFs on a file.

- `read_tightbox_ngwfs: T/F.`
  [Boolean, default `read_tightbox_ngwfs: F`].
  Read the NGWFs from a file and continue the calculation from this point.

  NOTE: if a calculation is intended to be restarted at some point in the future, then run the calculation with
  `write_tightbox_ngwfs: T`
  `write_denskern: T`
  to save the density kernel and the NGWFs on disk. Two new files will be created, with extensions `.dkn` and `.tightbox_ngwfs`, respectively. Then, to restart the calculation, set
  `read_tightbox_ngwfs: T`
  `read_denskern: T`

to tell ONETEP to read the files that were previously saved on disk. Remember to keep a backup of the output of the first run before restarting the calculation.

6 Controlling the parallel eigensolver

Currently, only the SCALAPACK PDSYGVX parallel eigensolver is available. A complete manual to this routine can be found by following the link in Ref. (9). If ONETEP is interfaced to SCALAPACK, the following directives can be used:

- `eigensolver_orfac: x.`
  [Real, default `eigensolver_orfac: 1.0e-4`].
  Precision to which the eigensolver will orthogonalise degenerate Hamiltonian eigenvectors. Set to a negative number to avoid reorthogonalisation with the SCALAPACK eigensolver.

- `eigensolver_abstol: x.`
  [Real, default `eigensolver_abstol: 1.0e-9`].
  Precision to which the parallel eigensolver will calculate the eigenvalues. Set to a negative number to use SCALAPACK defaults.

The abovementioned directives are useful in calculations where the SCALAPACK eigensolver fails to orthonormalise the eigenvectors. In such cases, the following error will be printed in the input file:

```
(P)DSYGVX in subroutine dense_eigensolve returned info= 2.
```

Many times (although not always) this error might cause the calculation to fail.

Bibliography

[1] E. Cancès and C. Le Bris. Can we outperform the diis approach for electronic structure calculations? *Int. J. Quantum Chem.*, 79(2):82, 2000.

[2] E. Cances. Self-consistent field algorithms for Kohn–Sham models with fractional occupation numbers. *J. Chem. Phys.*, 114(24):10616, 2001.

[3] P. Pulay. Convergence acceleration of iterative sequences - the case of SCF iteration. *Chem. Phys. Lett.*, 73(2):393, 1980.

[4] Y. A. Wang, C. Y. Yam, Y. K. Chen, and G. Chen. Linear-expansion shooting techniques for accelerating self-consistent field convergence. *J. Chem. Phys.*, 134(24):241103, 2011.

[5] Y. K. Chen and Y. A. Wang. LISTb: a better direct approach to LIST. *J. Chem. Theory Comput.*, 7(10):3045, 2011.

[6] Lapack. http://www.netlib.org/lapack/.

[7] Lapack DSYGVX eigensolver. http://netlib.org/lapack/double/dsygvx.f.

[8] ScaLapack. http://www.netlib.org/scalapack/.

[9] ScaLapack PDSYGVX eigensolver. http://www.netlib.org/scalapack/double/pdsygvx.f.

[10] V. R. Saunders and I. H. Hillier. Level-shifting method for converging closed shell Hartree-Fock wave-functions. *Int. J. Quantum Chem.*, 7(4):699, 1973.