Spectral Quadrature Density Functional Theory (SQDFT)

Developed by
Material Physics & Mechanics Group
(PI: Phanish Suryanarayana)
Georgia Institute of Technology

User guide

August 13, 2017

Contents

Chapter 1

Introduction

Spectral Quadrature Density Functional Theory (SQDFT) code is a C/C++ code for performing high-temperature Born-Oppenheimer Quantum Molecular Dynamics (QMD) in the framework of Kohn-Sham Density Functional Theory (DFT) [1, 2]. SQDFT is being developed by Phanish Suryanarayana's Material Physics & Mechanics Group at Georgia Institute of Technology. The main features of SQDFT can be summarized as follows:

- Electronic and dynamic properties of extended systems at high-temperatures.
- Local Density Approximation (LDA) [2].
- Norm-conserving Troullier-Martins pseudopotentials [3].
- Higher-order finite-difference discretization.
- $\mathcal{O}(N)$ Spectral Quadrature (SQ) method [4, 5].
- Parallelization via domain decomposition.

Please direct any questions and report any bugs to phanish.suryanarayana@ce.gatech.edu.

Acknowledgements This work was supported in part by the National Science Foundation (Grant number 1333500), and, in part, under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Support from the Laboratory Directed Research and Development program at the Lawrence Livermore National Laboratory is gratefully acknowledged. .

Chapter 2

Compilation and execution

SQDFT uses the following external library:

- MVAPICH2 2.1, http://mvapich.cse.ohio-state.edu .

Compilation SQDFT can be compiled from within the SQDFT folder using the commands:

```
make clean
make
```

A successful compilation will lead to the creation of the executable SQDFT/lib/sqdft .

Execution SQDFT can be executed in parallel using the mpirun command. Sample PBS script file is available in the SQDFT/tests folder. Note that all simulations using SQDFT require the .input and .atoms files in the root folder, and the corresponding pseudopotential files in the ./pseudopotentials subfolder. A detailed description of the contents of these files can be found in Chapter 3, with example prototypes available in the SQDFT/tests folders.

Example: To run a simulation on 8 processors with input files filename.input and filename.atoms, use the following command:

```
mpirun -np 8 ./lib/sqdft -name filename
```

Chapter 3

Inputs

Density Functional Theory (DFT) calculations using SQDFT require the following files as input:

- `.input` file: User options and parameters.
- `.atoms` file: Atomic information.

In addition, SQDFT requires Troullier-Martins pseudopotential files as generated by the “atom" code (`http://bohr.inesc-mn.pt/~jlm/pseudo.html`).

3.1 The `.input` file

The `.input` file contains the user options and parameters—identified through specific case sensitive *keywords*—for the QMD DFT simulation. The *keywords* can be listed in any order, unless specified otherwise. Each line of this file must contain only one *keyword*.

3.1.1 System information parameters

- `atoms_file` *(a string)*, default=none

  The name of the .atoms file is specified here.

- `domain` *(set of three real numbers, each separated by a whitespace delimiter)*, default=none, unit=Bohr

  The first, second, and third real numbers represent the cubical unit cell size in the x, y, and z-directions, respectively.

  Example: To specify a domain with x, y, and z-direction lengths of 7.78, 7.78, and 7.78 Bohr, use:

  ```
  domain 7.78 7.78 7.78
  ```

- `n_int` *(set of three integers, each separated by a whitespace delimiter)*, default=none

  The first, second, and third integers represent the number of finite-difference nodes in the x, y, and z-directions, respectively. The number of FD nodes should be identical in all the three directions. Note that the convergence of results with respect to spatial discretization needs to be verified.

  Example: To specify a grid with 10, 10, and 10 finite-difference nodes in the x, y, and z-directions, respectively, use:

  ```
  n_int 10 10 10
  ```

- `ncell` *(a positive integer)*, default = `1`

  The number of times the unit cell is repeated in each direction. It gets multiplied to the `domain` and `n_int` to give the total simulation domain size and total number of intervals in each direction, respectively.

  Example: If `domain`, `n_int`, and `ncell` are specified to be 7.78 7.78 7.78, 10 10 10 and 3 then the simulation domain size and number of intervals in each direction will be 23.34 and 30, respectively.

- `perturb` *(a real number)*, default=0.0, unit=Bohr

  Perturbs all the atoms randomly by a maximum of `perturb` in each direction.

  Example: To perturb the atoms by a maximum of 2.0 Bohr in each direction use:

  ```
  perturb 2.0
  ```

- `rand_seed` *(a natural number)*, default=`2`

  The number used as a random seed for perturbation of the atoms.

3.1.2 SQ parameters

- `T` *(a positive real number)*, default=none, unit=Kelvin

  The electronic and ionic temperature in Kelvin.

  Example: To specify the temperature as 100000 K use:

  ```
  T 100000
  ```

- `npl` *(a natural number)*, default=none

  The quadrature order for spectral quadrature. It should be a multiple of 4. Note that the convergence of results with respect to the quadrature order needs to be verified. It decreases with increasing mesh size and temperature.

  Example: To specify the quadrature order as 40 use:

  ```
  npl 40
  ```

- `Rcut` *(a positive real number)*, default=none

  It gives the truncation or localization radius. Note that the convergence of results with respect to `Rcut` need to be verified. It decreases with increasing temperature.

  Example: To specify the value of `Rcut` as 6.2 Bohr use:

  ```
  Rcut 6.2
  ```

3.1.3 Numerical parameters

- `FD_order` *(a positive integer)*, default=`12`

  The order of the finite-difference approximation. The default value of 12 has been found to be an efficient choice for most systems.

  Example: To specify the finite-difference order as 12, use:

  ```
  FD_order 12
  ```

- `poisson_tol` *(a real number)*, default=`1e-6`

  The convergence tolerance for Poisson solver.

  Example: To specify the Poisson tolerance as `1e-7`, use:

  ```
  poisson_tol 1e-7
  ```

- `poisson_maxiter` *(a positive integer)*, default=`1000`

  Maximum number of iterations for the Poisson solver.

  Example: To specify the `poisson_maxiter` as 500, use:

  ```
  poisson_maxiter 500
  ```

- `lanczos_tol` *(a real number)*, default=`1e-6`

  The convergence tolerance for Lanczos Algorithm.

  Example: To specify the Lanczos tolerance as `1e-7`, use:

  ```
  lanczos_tol 1e-7
  ```

- `fermi_tol` *(a real number)*, default=`1e-6`

  The convergence tolerance Fermi level calculation.

  Example: To specify the Fermi level tolerance as `1e-7`, use:

  ```
  fermi_tol 1e-7
  ```

- `scf_tol` *(a real number)*, default=`1e-4`

  The convergence tolerance for Self-consistent field (SCF) calculations.

  Example: To specify the SCF tolerance as `1e-6`, use:

  ```
  scf_tol 1e-6
  ```

- `scf_miniter` *(a positive integer)*, default=`3`

  Minimum number of SCF iterations in each MD step.

  Example: To specify the minimum iterations as 5, use:

  `scf_miniter 5`

- `scf_maxiter` *(a positive integer)*, default=`40`

  Maximum number of SCF iterations in each MD step.

  Example: To specify the maximum iterations as 100, use:

  `scf_maxiter 100`

3.1.4 Linear Solver parameters

- `beta_aaj` *(a floating point number between 0 and 1)*, default=`0.6`

  Relaxation parameter for Alternating Anderson-Richardson (AAR) linear solver [6, 7].

  Example: To specify the relaxation parameter as 0.5, use:

  `beta_aaj 0.5`

- `m_aaj`: *(a natural number)*, default=`7`

  Number of previous iterates used for extrapolation in the AAR linear solver.

  Example: To specify the extrapolation history parameter as 5, use:

  `m_aaj 5`

- `p_aaj`: *(a natural number)*, default=`6`

  Frequency of Anderson extrapolation in the AAR method.

  Example: To specify the frequency parameter as 8, use:

  `p_aaj 8`

3.1.5 Mixing parameters

- `beta_scf` *(a floating point number between 0 and 1)*, default=`0.1`

  Relaxation parameter for the Periodic Pulay mixing scheme [8].

  Example: To specify the mixing parameter as 0.5, use:

  `beta_scf 0.5`

- `m_scf`: *(a natural number)*, default=`7`

  Number of previous iterates used for extrapolation in the Periodic Pulay mixing scheme.

  Example: To specify the mixing extrapolation history parameter as 5, use:

  `m_scf 5`

- `p_scf`: *(a natural number)*, default=`2`

  Frequency of Anderson extrapolation in the Periodic Pulay mixing scheme.

  Example: To specify the mixing frequency parameter as `8`, use:

  ```
  p_scf 8
  ```

3.1.6 Other Options

- `non_blocking` (`0` or `1`), default=`0`

  Flag for usage of non-blocking MPI collective communication routines.

  Example: To utilize non-blocking MPI, use:

  ```
  non_blocking 1
  ```

- `print_atoms` (`0` or `1`), default=`0`

  Flag for printing atomic positions and forces into a `.aout` file.

  Example: To print the atomic positions and forces to a file, use:

  ```
  print_atoms 1
  ```

- `restart_scf` (`0` or `1`), default=`0`

  Flag for determining whether the electron density guess should be read from the `.restart` file.

  Example: To read the guess electron density from the `.restart` file, use:

  ```
  restart_scf 1
  ```

- `print_scf` (`0` or `1`), default=`0`

  Flag for determining whether the electron density at the end of the last SCF iteration should be written into the .restart file.

  Example: To write the electron density into a .restart file, use:

  ```
  print_scf 1
  ```

- `fermi_alg` (`1` or `2`), default=`1`

  Flag for choosing between Brent's algorithm and Newton-Raphson for calculation of the Fermi level.

  Example: To use Newton-Raphson algorithm , use:

  ```
  fermi_alg 2
  ```

3.1.7 MD Options

- `qmass` *(a positive real number)*, default=`1`, unit=(atomic mass unit) $\times$ Bohr^2

  Thermostat mass in NVT simulations.

  Example: To set the thermostat mass as 0.1, use:

  ```
  qmass 0.1
  ```

- `vel_dstr` (`1` or `2`), default=`1`, unit=Bohr/fs

  Flag for choosing between the initial velocity distribution for the atoms in the QMD simulation. 1 corresponds to Maxwell-Boltzmann distribution and 2 corresponds to uniform.

  Example: To specify uniform initial velocity , use:

  ```
  vel_dstr 2
  ```

- `ensemble` (`1` or `2`), default=none

  Flag for specifying either canonical (NVT) or microcanonical (NVE) ensemble.

  Example: To perform an NVT simulation , use:

  ```
  ensemble 1
  ```

- `time_step` *(a positive real number)*, default=none, unit=fs

  Time step for QMD simulations.

  Example: To specify a time step as 0.5, use:

  ```
  time_step 0.5
  ```

- `MaxMDsteps` *(a positive integer)*, default=none

  Number of time steps in the QMD simulation.

  Example: To specify the number of steps as 1000, use:

  ```
  MaxMDsteps 1000
  ```

- `ChgExtrap` (`0` or `1`), default=`1`

  Flag for choosing electron density extrapolation between MD steps.

  Example: To employ charge extrapolation, use:

  ```
  ChgExtrap 1
  ```

- `restart_md` (`0` or `1`), default=`0`

  Flag for restarting the QMD simulation with the information stored in the `.restartMD` file.

  Example: To restart the QMD simulation, use:

  ```
  restart_md 1
  ```

- `name` (a string), default=`position2.txt`

  Name of file in which trajectories of a QMD simulation are stored.

- `prnt_md` (`0` or `1`), default=`1`

  Flag for choosing whether the current QMD step's information should be stored in .restartMD file.

  Example: To specify the parameter as 0, use:

  ```
  prnt_md 0
  ```

- `RelaxAtoms` (`0` or `1`), default=`0`

  Flag for specifying whether atomic relaxation is performed.

  Example: To perform atomic relaxation, use:

  ```
  RelaxAtoms 1
  ```

3.2 The `.atoms` file

The `.atoms` file is used to provide the required atomic information, i.e, the type, number, and position of atoms. The layout of the file is as follows. The first line specifies the total number of atoms. It is followed by number of types of atoms in the second line. Subsequently, data *blocks* are used to provide information related to each of the atomic species in the system. The first line of each *block* contains—separated by a whitespace delimiter—the symbol for the chemical element, the number of valence electrons, atomic mass, number of such atoms, local component of pseudopotential, cut-off radii for the pseudocharge density, and ends with the path of pseudopotential file on the next line. The remainder of the *block* contains their atomic positions, with the x, y, z-coordinates—separated by a whitespace delimiter—of each atom listed on a separate line.

Example: The contents of the `.atoms` file for an 8-atom unit cell of Lithium Hydride :

```
8
n_typ 2

Atoms
Li 1 6.941 4 0 8
./pseudopotentials/psd_Li.pot
0.570000000000000 0.430000000000000 0.370000000000000
3.685000000000000 0.000000000000000 3.685000000000000
3.685000000000000 3.685000000000000 0.000000000000000
0.000000000000000 3.685000000000000 3.685000000000000

H 1 1.00794 4 0 8
./pseudopotentials/psd_H.pot
```

```
3.685000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.000000000000000 3.685000000000000
0.000000000000000 0.000000000000000 3.685000000000000
3.685000000000000 3.685000000000000 3.685000000000000
```

Chapter 4

Outputs

In addition to the output file options listed in the previous chapter, SQDFT prints the following information into the output (`.out`) file as separate *blocks*:

- `Input parameters`: The options and parameters specified by the user.
- `Initialization`: The initialization details.
- `Pseudocharge formulation`: The total pseudocharge, self energy of the pseudocharge densities, and the repulsive energy correction.
- `Self Consistent Field`: The free energy and the normalized residual of the effective potential in each Self Consistent Field (SCF) iteration.
- `Energy and atomic forces`: The free energy as well as its individual components, the Fermi energy, forces in all directions and mean energies and temperature for MD calculations.
- `Timing summary`: The summary of timings of different parts of the calculation.

Chapter 5

Frequently Asked Questions (FAQs)

- **Question**: Is SQDFT compatible with other versions of MVAPICH?

  Answer: Though SQDFT has been tested with other versions of MVAPICH, there is a possibility of facing compilation issues.

- **Question**: Are any exchange-correlation functionals other than Local Density Approximation (LDA) supported by SQDFT?

  Answer: SQDFT currently only supports LDA. Other exchange-correlation functionals will be incorporated into future releases of SQDFT.

- **Question**: Are any pseudopotentials other than Troullier-Martins supported by SQDFT?

  Answer: SQDFT currently only supports Troullier-Martins pseudopotentials. Ability to utilize other pseudopotentials will be incorporated into future releases of SQDFT.

- **Question**: Are pseudopotentials with non-linear core corrections and relativistic effect supported by SQDFT?

  Answer: No.

Bibliography

[1] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864–B871, 1964.

[2] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133–A1138, 1965.

[3] N. Troullier and J. L. Martins. Efficient pseudopotentials for plane-wave calculations. *Physical Review B*, 43(3):1993–2006, 1991.

[4] Phanish Suryanarayana. On spectral quadrature for linear-scaling density functional theory. *Chemical Physics Letters*, 584:182–187, 2013.

[5] Phanisri P Pratapa, Phanish Suryanarayana, and John E Pask. Spectral quadrature method for accurate o (n) electronic structure calculations of metals and insulators. *Computer Physics Communications*, 2015.

[6] Phanisri P Pratapa, Phanish Suryanarayana, and John E Pask. Anderson acceleration of the jacobi iterative method: An efficient alternative to krylov methods for large, sparse linear systems. *Journal of Computational Physics*, 306:43–54, 2016.

[7] Phanish Suryanarayana, Phanisri P Pratapa, and John E Pask. Alternating anderson-richardson method: An efficient alternative to preconditioned krylov methods for large, sparse linear systems. *arXiv preprint arXiv:1606.08740*, 2016.

[8] Amartya S Banerjee, Phanish Suryanarayana, and John E Pask. Periodic pulay method for robust and efficient convergence acceleration of self-consistent field iterations. *Chemical Physics Letters*, 2016.