CONICET
INQUIMAE

LIO User Guide

V 1.11

January, 2020

Version Developers

Contents

Contents **iii**

1 Introduction **1**

1.1 What is LIO? 1

1.2 Instalation 1

1.3 Running LIO 4

1.4 Tips and tricks - Optimizing your runs 5

2 General Settings **6**

2.1 General System Description 6

2.2 Common Inputs and Outputs 7

2.3 Properties Calculations 8

2.4 GPU Options 9

3 Model Hamiltonian **11**

3.1 Density Functional Theory 11

3.2 Convergence Options 12

3.3 External Electric Fields 14

3.4 Effective Core Potentials 14

4 Ground State Calculations **19**

4.1 Single-point and Born-Oppenheimer Molecular Dynamics 19

4.2 Geometry optimizations 20

4.3 Restraints 23

4.4 Changing Lennard-Jones parameters 25

5 Electron Dynamics **27**

5.1 Real Time TD-DFT 27

5.2 Electronic transport 27

5.3 Ehrenfest Dynamics 27

6 Post-Processing Tools **28**

6.1 TD-Analize: Electronic Spectra 28

6.2 CubeGen: Orbital and Density Visualization 28

7 Reference Section **29**

7.1 Command line options 29

7.2 Keywords - General Setup 30

7.3 Keywords - GPU Options 32

7.4 Keywords - DFT Hamiltonian 33

7.5 Keywords - Effective Core Potentials 35

7.6 Keywords - DFTB Embedding 37

7.7 Keywords - Fields and Biases 38

7.8 Keywords - Self Consistent Field 40

7.9 Keywords - Geometry Optimization 42

7.10 Keywords - Real Time TD-DFT 43

7.11 Keywords - Transport 44

7.12 Keywords - Ehrenfest 45

7.13 Keywords - CubeGen 47

References 49

Chapter 1

Introduction

1.1 What is LIO?

Welcome to the LIO project! LIO is a library that can perform electronic structure calculations using density functional theory.

1.2 Instalation

If you are reading this manual, you probably already have a version of LIO ready to compile. If you don't, or if you want to make sure you have the most most up-to-date version of the code, all you need to do is either download it the git repository online or use git to clone a copy.

For the first option, go to ***https://github.com/MALBECC/lio*** and click on the green button that says *clone or download* and click on *Download ZIP*.

For the second one, you can directly run the following command:

```
git clone https://github.com/MALBECC/lio.git .
```

Pre-requisites

In addition to an UNIX-like OS, fortran and c++ compilers, LIO depends on LAPACK *LAPACK* and BLAS libraries for linear algebra calculations. In addition, *CUDA 6.5*

or higher is required for GPU calculations,which unleash LIO's true potential. As of the writing of this manual, LIO has not yet been tested with CUDA 9.0.
In addition, *libxc* is required for its usage, although said library is CPU-only. This is entirely optional as LIO can run without libxc, using only the PBE functional.

Compilation

By default, LIO compiles with GPU options enabled. It is highly recommended to specify the GPU architecture as a compilation option, since the compiler performs additional enhancements. After compilation, LIOHOME environment variable should be set to the current LIO installation directory:

```
export LIOHOME=/dir/to/lio/
```

For a CPU-only compilation, use:

```
make cuda=0
```

If INTEL compilers are present, they can be used by setting the *intel* option to 1, or to 2 if INTEL MKL usage is also desired.

```
make intel=2
```

The following is a list of available compilation and their meanings. They can be used in any combination possible (including several GPU architectures for greater compatibility). For example, the default LIO compilation could be written as:

```
make cuda=2 sm30=1 sm52=1 sm61=1
```

Variable	Description
cuda *0, 1, 2*	*default cuda=2* Sets the level of GPU dependencies. cuda=0 means a CPU-only compilation, =1 enables GPU, and =2 includes CUBLAS usage for linear algebra operations.
intel *0, 1, 2*	*default intel=0* Sets the usage of INTEL compilers. intel=0 means only GNU compilers, =1 means INTEL, and =2 uses INTEL MKL instead of BLAS/LAPACK routines.

Table 1.1: Compilation options

Variable	Description
precision *0, 1*	*default precision=0* Sets level of precision in XC calculations. By default LIO uses mixed precision; precision=1 sets everything in double precision. This is specially useful when attempting high-precision geometry optimizations.
analytics *0, 1, 2, 3, 4*	*default analytics=0* Setting analytics = 1 specifies a profiling compilation, while analytics = 2 and 3 set higher levels of debugging information (for usage with gdb, for example) and 4 includes additional checks for some integrals.
smXX *0, 1*	*default sm30=1 sm52=1 sm61=1* Sets a specific GPU architecture for compilation. Available options are sm30, sm35 (Kepler, CUDA $\geq$ 5.0), sm50, sm52 (Maxwell and GeForce 980, CUDA $\geq$ 6.5), sm60 and sm61 (Pascal and GeForce 1080, CUDA $\geq$ 8.0).

Table 1.1: Compilation options

MD-Engine Interfacing

LIO can be linked with *AMBER*, our own *GROMACS* fork, our own *HYBRID* code for QM/MM calculations.

In all three cases, the LIOHOME environment variable should be set to the current LIO installation directory. In addition to this section, please refer to each of the software packages' installation manual for further clarifications.

In order to compile AMBER with LIO, AMBER should be compiled after LIO with the following options set:

```
export AMBERHOME=/dir/to/amber/
./configure -lio -noX11 -netcdfstatic gnu
make clean
make install
```

For a GROMACS-LIO compilation, GROMACS should be compiled after LIO with the following options:

```
cd gromacs_compilation_directory/
cmake gromacs_src_dir/ -DGMX_QMMM=1
    -DGMX_QMMM_PROGRAM="lio" -DLIO_LINK_FLAGS="-L/usr/lib
    -L/usr/lib64 -L$LIOHOME/g2g -L$LIOHOME/lioamber -lg2g
    -llio -g2g" -DGMX_GPU=0 -DGMX_THREAD_MPI=0
make
make install
```

1.3 Running LIO

In order to run LIO as a QM stand-alone program, you must first set the LIOHOME environment variable to that of the current LIO installation path (if it was not set before), and add /liosolo folder to the current excecutable paths.

```
export LIOHOME=/dir/to/lio/
export PATH=$PATH:$LIOHOME/liosolo
```

Then, LIO can be excecuted as follows:

```
liosolo -i input_file -c coords.xyz
```

Where input_file contains the input options and coords.xyz contains the coordinates in XYZ format. For more options, please refer to Chapter 2.
When running LIO with AMBER, you only need to specify '***qm_theory=extern***' in the &qmmm section of the code (please refer to the AmberTools manual for further &qmmm options), in addition to specifying both the QM system and its charge. When running with GROMACS, you only need to specify the QM system and its charge. In both cases, additional LIO options will be read from a 'lio.in' file in the working directory.

1.4 Tips and tricks - Optimizing your runs

There are several ways to optimize your production runs, and it is highly recommended to fine tune these settings if the same system or very similar systems are going to run for extended periods of time. Please see the corresponding sections in this manual for a more detailed explanation of these variables.

The first set to tune is ***rmax*** and ***rmaxs***, which are related to the integral cut-offs used in both Coulomb and fitting set integrals. The *higher* the value of ***rmaxs*** and the *lower* the value of ***rmax***, the calculations will run faster. Keep in mind, however, that ***rmax*** should *never* be lower that ***rmaxs***, and that tuning those values for a faster calculation will certainly result in a loss of precision. Therefore, the idea is to decrease ***rmax*** and increase ***rmaxs*** until the difference in energies and forces stops being negligible.

Next is the set of options available for the GPU library which performs the exchange-correlation calculations. The option ***max_function_exponent*** indicates the maximum exponent considered for a fuction in a point of the grid; it should be tweaked taking into account the aforementioned criterium: the faster the calculation is performed, the lesser precision is achieved. There are also three other options whose optimal values depend on the GPU architecture available: ***little_cube_size***, ***min_points_per_cube***, and ***sphere_radius***. These can be tweaked for maximum speed without worrying about the resulting precision.

Chapter 2

General Settings

In this chapter we will describe the basic settings for running the code. These will be necessary or useful for any of the features of the code that you might be interested in using.

2.1 General System Description

Variable	Description
natom	$integer = 0$ Sets the number of atoms only in the QM region. This is ignored when calling LIO from another MM engine.
nsol	$integer = 0$ Sets the number of "solvent" atoms, meaning atoms in the MM region. This is also ignored when calling LIO externally.
charge	$integer = 0$ Sets the total charge for the QM region.
OPEN	$logical = .false.$ Indicates whether an open-shell calculation should be performed.
nUnp	$integer = 0$

Table 2.1: General setup

Variable	Description
	Indicates the number of unpaired electrons for an open-shell calculation, in a closed-shell calculation this variable is ignored. Keep in mind that the number of unpaired electrons is NOT the multiplicity, rather nUnp = Mult+1.

Table 2.1: General setup

2.2 Common Inputs and Outputs

Variable	Description
style	$logical = .false.$ Activates a more stylish version of the output.
verbose	$integer = 0$ Sets verbose level for output. 0 = Nothing is printed save for a small LIO start message. 1 = only the input namelist and the final convergence results are printed. 2 = information for each iteration is added. 3 = GPU module information is added. 4 = additional miscellaneous information is printed.
writeXYZ	$logical = .false.$ Writes and xyz file containing the QM region, using the filename specified in fcoord. This file is written in each of the MD steps when LIO is used with AMBER/GROMACS/HYBRID.
fCoord	$character * 20 =' qm.xyz'$ Output file containing the QM region.
timers	$integer = 0$ Sets the timer level for benchmarking and profiling. 0 = no timers. 1 = only timers in certain subroutines. 2 = full timer summary.
dBug	$logical = .false.$ Checks for NaNs in Fock and Rho matrices.

Table 2.2: Common inputs and outputs

Variable	Description
gaussian_convert	$logical = .false.$ Reads an electron density from Gaussian09 as starting guess. This feature is still experimental.
print_coeffs	$logical = .false.$ Prints coefficients and energies for each molecular orbital.
VCInp	$logical = .false.$ Reads a MO coefficient restart from frestartin.
restart_freq	$integer = 1$ Writes a MO coefficient restart every restart_freq iterations into frestart.
frestart	$char * 20 =' restart.out'$ Name of the output density or coefficient restart file.
frestartin	$char * 20 =' restart.in'$ Name of the input density or coefficient restart file.
rst_dens	$integer = 0$ When rst_dens >= 1, the restart read by VCInp must be a density matrix (not coefficient matrix) restart. Likewise, when rst_dens = 2, the restart written with restart_freq is a density matrix restart (instead of a coefficient matrix restart).

Table 2.2: Common inputs and outputs

2.3 Properties Calculations

Variable	Description
writeForces	$logical = .false.$ Performs forces calculation and prints the result to a file named "forces".
dipole	$logical = .false.$ Performs dipole calculation and prints the result to a file named "dipole".

Table 2.3: Common inputs and outputs

Variable	Description
mulliken	$logical = .false.$ Performs a Mulliken population analysis and outputs the result to a file called "mulliken".
lowdin	$logical = .false.$ Performs a Lowdin population analysis and outputs the result to a file called "mulliken".
fukui	$logical = .false.$ Calculates the condensed-to-atoms Fukui function for the system, calculating the spin-polarized version in open-shell systems.

Table 2.3: Common inputs and outputs

2.4 GPU Options

These options affect the calculations in the GPU module. For the exchange correlation integrals, the integration grid is separated into cubes and spheres, with the smaller cubes (less points per cube) being calculated in CPU while the rest is calculated in GPU. Most of these options should be tweaked for optimal performance in a given system.

Variable	Description
gpu_level *integer*	*default = 4* Determines which calculations are performed by the GPU, only available when compiled with cuda > 0. 0/1 = only exchange-correlation integrals. 2 = adds QM/MM interaction energies and gradients. 3 = adds Coulomb energies and gradients. 4 = adds nuclear attraction energies and gradients. 5 = adds Coulomb basis fitting energies and gradients.
max_function_exponent *integer*	*default = 10* Ignores functions with $\|exponent\| > max\_function\_exponent$. This is only for the exchange-correlation calculations.

Table 2.4: GPU Module Options

Variable	Description
little_cube_size *double precision*	*default = 8.0d0* Small cube-type point group size (in Angstrom).
min_points_per_cube *integer*	*default = 1* Minimum number of grid points in a cube.
assign_all_functions *logical*	*default = .false.* Calculate **all** functions (ignores $max\_function\_exponent$). This is intended only as a debug option and its usage is not recommended.
sphere_radius *double precision*	*default = 0.6d0* Radius of the sphere-type point groups. 0 means there are no sphere-type groups, 1 means all points are contained in sphere-type groups.
remove_zero_weights *logical*	*default = .true.* Discard functions for those whose weight is zero (.false. option only remains as a debug option).
energy_all_iterations *logical*	*default = .false.* Calculate Exc energy in all SCF iterations. Usually, XC energy is only calculated in the final step in order to accelerate calculations.
free_global_memory *double precision*	*default = 0.0d0* Fraction of global GPU memory available for the calculation (1 means 100%).

Table 2.4: GPU Module Options

Chapter 3

Model Hamiltonian

3.1 Density Functional Theory

Variable	Description
iexch	$integer = 9$ Identifies the exchange-correlation potential to use with the calculation when not using libxc. Iexch=9 is the only option currently available.
use_libxc	$logical = .false.$ Activates the use of libxc version of the XC potential.
ex_functional_id	$integer = none$ Exchange functional to use with libxc. Please refer to libxc for the desired functional.
ec_functional_id	$integer = none$ Correlation functional to use with libxc. Please refer to libxc for the desired functional.
int_basis	$logical = .false.$ If true, looks for the internal basis indicated in variables $basis\_set$ and $fitting\_set$ (defaults are $'DZVP'$ and$'DZVPCoulombFitting'$. If false, an external basis file must be provided in the keyword $basis\_set$.
basis_set	$char * 20 =' DZVP'$

Table 3.1: DFT Hamiltonian

Variable	Description
	Name of the basis set used in the calculation when int_basis is set to true. If not, it is the name of the file containing the custom basis data.
fitting_set	$char * 100 =' DZVPCoulombFitting'$ Name of the fitting set used in the calculation when int_basis is set to true.
n_ghosts	$integer = 0$ Number of ghost atoms. Ghost atoms are considered for the basis functions of the system, but they are considered as having zero electrons and zero nuclear charge.
ghost_atoms	$integer = 0$ A list containing the atom indeces for those considered ghost_atoms. For example, $ghost\_atoms = 1, 2$ considers the first and second atoms as ghosts.
rmax	$double = 16.0d0$ Maximum exponent in 3-center integrals. If exponent is greater than $rmax$, the current term is ignored.
rmaxs	$double = 5.0d0$ If the exponent in 3-center integral is within $rmax$ and $rmaxs$, calculation is performed using double precision. If it is between 0 and $rmaxs$, the calculation is performed in single precision. This is ignored when using the AINT module.
iGrid	$integer = 2$ Grid type when iterating through SCF. Available values are 1 (less dense) and 2 (more dense).
iGrid2	$integer = 2$ Grid type for the final density calculation in SCF. Available values are 1 (less dense) and 2 (more dense).

Table 3.1: DFT Hamiltonian

3.2 Convergence Options

Variable	Description
initial_guess	$integer = 0$ Selects the method for calculating a starting guess. When set to 0, the initial guess comes from the 1-electron integrals, while setting it to 1 performs an Aufbau-like initial guess.
nMax	$integer = 100$ Maximum number of SCF iteration steps.
told	$double\ precision = 1.0d-6$ Criterium for the maximal square deviation of the density matrix to consider that the convergence has been achieved.
etold	$double\ precision = 1.0d0$ Criterium for the maximal energy difference to consider that the convergence has been achieved.
DIIS	$logical = .true.$ Uses DIIS algorithm for convergence. If disabled, it tries to converge using a damping factor between iterations.
nDIIS	$integer = 30$ Number of matrices considered for DIIS. Only change this if you know what you are doing.
gold	$double\ precision = 1.0d+1$ Determines the weight of the previous density matrix in the linear combination with the new one when using the damping convergence method (DIIS = .false. / hybrid_converg).
hybrid_converg	$logical = .false.$ Uses the hybrid convergence algorithm: it starts using the damping factor, and after a threshold is met, changes to DIIS. This method usually gives the best results in cases which are difficult to converge.
good_cut	$double\ precision = 1.0d-3$ Sets the threshold to start DIIS when activating hybrid convergence (hybrid_converg = 1). When the mean square deviation for the density matrix between two iteration steps reaches this threshold, DIIS is activated.

Table 3.2: BO-MD useful settings.

3.3 External Electric Fields

Variable	Description
field	$logical = .false.$ Use an external field as a simple uniform field in SCF or as a perturbation in TD-DFT calculations.
a0	$double = 1.0d3$ A dividing factor in electric field calculations. Do not touch under any circumstance unless you know what you are doing.
epsilon	$double = 1.0d0$ Relative electric permitivity of the medium.
Fx, Fy, Fz	$double = 0.05d0$ The value of the external electric field in the x, y and z directions. In the case of time-dependent fields, it sets the maximum value.
nfields_iso	$integer = 0$ Number of shape-isotropic fields. If $nfields\_iso = 0$, the inputs in $field\_iso\_file$ are ignored.
field_iso_file	$char * 20 =' field.in'$ Isotropic fields input file.
nfields_aniso	$integer = 0$ Number of shape-anisotropic fields. If $nfields\_aniso = 0$, the inputs in $field\_aniso\_file$ are ignored.
field_aniso_file	$char * 20 =' field.in'$ Anisotropic fields input file.

Table 3.3: Fields and Biases

3.4 Effective Core Potentials

Lio can replace the inner electron representation using effective core potentials (ECP), which can make more efficient the calculations especially on molecules containing heavy elements with irrelevant core electrons for the electronic problem of analysis.

The contribution of an atomic ECP to the Fock Hamiltonian it is given by [1]

$$\hat{H}^{ECP} = \frac{e^2}{4\pi\epsilon_0} \sum_{I=1}^{M} \sum_{i=1}^{n} \frac{N_I^C}{|\vec{R}_I - \vec{r}_i|} + \hat{V}^{ECP} \tag{3.1}$$

where the first term of 3.1 corrects the nuclear charge to compensate the removed internal electrons (N_I^C) in the electrostatic interaction and the second one reproduces the repulsion of the electronic density in the vicinity of the nucleus which should be produced by the electrons removed in the calculation.

This second term is given by:

$$\hat{V}^{ECP} = \sum_{A \in \substack{\text{átomos} \\ \text{con ECP}}} \left[\hat{V}_L(A) + \sum_{l=0}^{L-1} \sum_{m=-l}^{l} |lm^A\rangle \hat{V}_l(A) \langle lm^A| \right] \tag{3.2}$$

where $|lm^A\rangle$ and $\langle lm^A|$ correspond to real orthonormal spherical harmonics centered on atom A and $\hat{V}_l(A)$ are functions that depends only on the distance to the core A, generally parameterized as a product of Gaussian functions by polynomials.

The implementation of 3.1 matrix elements was made following the works of Khan [1] and Bode [2], full documentation may be viewed in [3].

The derivatives of matrix elements $V_{\mu\nu}^{ECP}$ ($\langle \phi_\mu^B | V^{E\hat{C}P,A} | \phi_\nu^C \rangle$ are obtained by simple derivation of Gausian basis sets and translational invariance.[4]

$$\frac{\partial V_{\mu\nu}^{ECP}}{\partial x_B} = \frac{\partial \langle \phi_\mu^B|}{\partial x_B} V^{E\hat{C}P,A} |\phi_\nu^C\rangle = \langle x_B \phi_\mu^B | V^{E\hat{C}P,A} | \phi_\nu^C \rangle a_B - \langle x_B^{-1} \phi_\mu^B | V^{E\hat{C}P,A} | \phi_\nu^C \rangle l_{x_B} \tag{3.3}$$

$$\frac{\partial V_{\mu\nu}^{ECP}}{\partial x_A} = \langle \phi_\mu^B | \frac{\partial V^{E\hat{C}P,A}}{\partial x_A} | \phi_\nu^C \rangle = -\frac{\partial \langle \phi_\mu^B|}{\partial x_B} V^{E\hat{C}P,A} |\phi_\nu^C\rangle - \langle \phi_\mu^B | V^{E\hat{C}P,A} \frac{\partial |\phi_\nu^C\rangle}{\partial x_C} \tag{3.4}$$

where a_B and l_{x_B} are the gaussian exponent and the power of (x/r) of the function $\langle \phi_\mu^B || r \rangle$ respectively.

Diferent parametrizations of ECP are contained in Lio data among them the effective compact potentials (CEP) [5–7] (also called SBKJC), CRENBL [8–11], Los Alamos (LAN) [12–14], and the pseudo-potentials of Stuttgart [15].

The essential variables to be defined in input for use ECP in Lio are ECPMode, ECP-Types, tipeECP, and ZListECP. ECPMode turns on ECP calculations, ECPTypes

defines the number of types of atoms that will have ECP, tipeECP define the ECP to be used and ZListECP contains de atomic number of atoms with ECP. For example in a calculation in wich Fe, C, N and O will have a SBKJC ECP the input have to have:

```
&lio
...
...
ECPMode = t
tipeecp = "SBKJC"
ecptypes = 4
ZlistECP = 26,6,7,8
...
...
&end
```

The description of these and other variables is presented in tables 3.4 and 7.5.

tips & triks:

Restart:

For fixed nuclei calculations (as RT-TDDFT) all Fock terms of ECP are constant and can be writed/readed form a restart file. This is done by turning True fock_ECP_read & fock_ECP_write in Lio input.

cutoff ECP interaction:

Lio use 2 cutoffs to skip the calculation of those ECP integrals that will be null, cut2_0 and cut3_0. cut2_0 neglects all 2 center integrals ($\langle\phi_\mu^A|V^{\hat{ECP},A}|\phi_\nu^B\rangle$) wich $distance^2_{A-B} * a_B > cut2\_0$, beeing a_B the exponent of $\langle r||\phi_\nu^B\rangle$. In same way cut3_0 neglects all 3 center integrals $\langle\phi_\mu^B|\hat{V}^{ECP,A}|\phi_\nu^C\rangle$) wich $(distance^2_{A-B} * a_B + distance^2_{A-C} * a_C) > cut3\_0$. Both values are predefined, but can be optimized for each particular calculation.

Variable	Description
ECPMode	$logical = .false.$ Activate effective core potentials.
ECPTypes	$integer = 0$ Number of atoms with ECP.
tipeECP	$char * 30 =' NOT_DEFINED'$ Type of ECP used.
ZListECP	$integer = 0$ Array with Z of atoms with ECP enabled.
cutECP	$logical = .true.$ Enables cuts for ECP integrals. Don't turn off unless you know what you are doing.
cut2_0	$double = 15.0d0$ Cut value for 2-center ECP integrals.
cut3_0	$double = 12.0d0$ Cut value for 3-center ECP integrals.
ECP_debug	$logical = .false.$ Enables ECP debug mode.
local_nonlocal	$integer = 0$ Calculates only local terms (when = 1) or only non-local terms (when = 2).
ECP_full_range_int	$logical = .false.$ Enables full-range integral calculations.
verbose_ECP	$integer = 0$ Controls ECP verbose levels.
fock_ECP_read	$logical = .false.$ Enables restart read in ECP.
fock_ECP_write	$logical = .false.$ Enables restart write in ECP.
fullTimer_ECP	$logical = .false.$ Enables full timers in ECP.

Table 3.4: Effective Core Potentials

Variable	Description

Table 3.4: Effective Core Potentials

Chapter 4

Ground State Calculations

4.1 Single-point and Born-Oppenheimer Molecular Dynamics

Single-point calculations consist in finding the ground state density that minimizes the energy for a given nuclei distribution. Once found, the program can also calculate the force field of that density for the MD-engine to move the nuclei.

Variable	Description
initial_guess	$integer = 0$ Selects the method for calculating a starting guess. This is only useful for the first MD step, since after that the starting guess is the electron density from the previous step.
nMax	$integer = 100$ Maximum number of SCF iteration steps.
told	$double\ precision = 1.0d - 6$ Criterium for density matrix convergence.
etold	$double\ precision = 1.0d0$ Criterium for energy convergence.
DIIS	$logical = .true.$ Uses DIIS algorithm for convergence.
hybrid_converg	$logical = .false.$

Table 4.1: BO-MD useful settings.

Variable	Description
	Uses the hybrid convergence algorithm.
good_cut	$double\ precision = 1.0d-3$ Sets the threshold to start DIIS when activating hybrid convergence.
VCInp	$logical = .false.$ Reads a MO coefficient restart.
restart_freq	$integer = 1$ Writes a MO coefficient restart every restart_freq steps.
frestart	$char*20 =' restart.out'$ Name of the output restart file.
frestartin	$char*20 =' restart.in'$ Name of the input restart file.
rst_dens	$integer = 0$ rst_dens = 1 reads a density matrix restart, while rst_dens = 2 both reads and writes a density matrix restart.
basis_set	$char*20 =' DZVP'$ Name of the basis set used in the calculation.

Table 4.1: BO-MD useful settings.

4.2 Geometry optimizations

Geometry optimizations or energy minimization is the process of finding an atomic arrangement in space where the force on each atom is acceptably close to zero.

Implemenation

LIO has a simple steepest-descent algorithm. The idea is to move the system in the force direction, at a λ step value.

$$\vec{r}^{i}_{new} = \vec{r}^{i} + \lambda \vec{F}^{i} \tag{4.1}$$

Without a linear search algorithm λ is obtained as $\frac{steep\_size}{|\vec{F}_{max}|}$. If the energy decreases with the movement, the step is accepted; but if the energy increases with the step, the steep is rejected and λ is reduced. Each accepted move increases step size a 20% and each rejected move decreases step size a 50%.

In a linear search algorithm the system scans the energy as function of λ and predicts the best value of λ to move the system in the gradient direction.

Best λ in lineal search algorithm is obtained by a quadratic function ajusted using minimum energy of the scan and previous and next points.

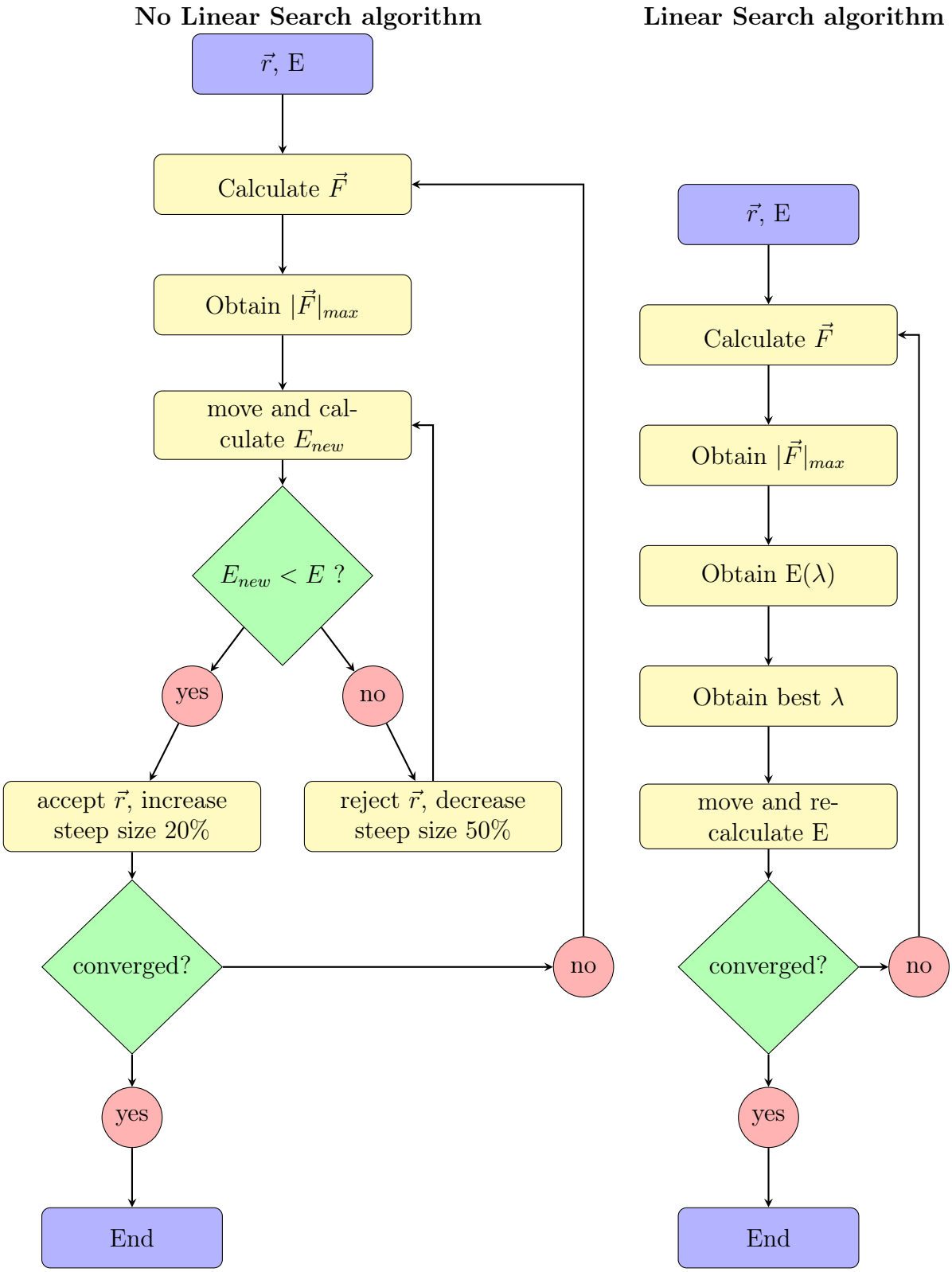
Using geometry optimizations

Adding steep=t in LIO input enables geometry optimization (steepest descent, lineal search by default). Convergence criteria are set by Force_cut and Energy_cut (5E-4 Hartree/bohr and 1E-4 Hartree by Default). The number of minimization steeps is set by n_min_steps (500 by default) and initial distance steep is set by minimzation_steep (by default 0.05 bohr)

It is highly advisable to compile LIO in double precision in order to minimise the error in exchange-correlation forces (precision=1). Outputs of geometry optimizations are traj.xyz (atoms coordinates in each steepes descent movement) and optimization.out (steep, energy and others). If verbose=true optimization.out includes the energy of each linear search point.

Examples

Examples of geometry optimization are made in lio/test/13_geom_optim.


No Linear Search algorithm
$\vec{r}$, E
Calculate $\vec{F}$
Obtain $|\vec{F}|_{max}$
move and calculate E_{new}
$E_{new} < E$?
yes
no
accept $\vec{r}$, increase steep size 20%
reject $\vec{r}$, decrease steep size 50%
converged?
no
yes
End
Linear Search algorithm
$\vec{r}$, E
Calculate $\vec{F}$
Obtain $|\vec{F}|_{max}$
Obtain E(λ)
Obtain best λ
move and recalculate E
converged?
no
yes
End

4.3 Restraints

LIO may add an extra potential term to the Hamiltonian in order to restrain the distance between specified pairs of atoms.

Implemenation

The implementation is a simple harmonic potential over a generalized coordinate r.

$$U = \frac{1}{2}k[r - l_0]^2 \tag{4.2}$$

r may be defined as a weighted combination of distances between pairs of atoms.

$$r = \sum_i \sum_{j>i} w_{ij} |\vec{r_i} - \vec{r_j}| \tag{4.3}$$

In this formulation the force over an atom l is:

$$\vec{F_l} = -k[r - l_0] \sum_i \sum_{j>i} w_{ij} \frac{\vec{r_{ij}}}{r_{ij}} \eta_{ijl} \tag{4.4}$$

Where η_{ijl} is defined as:

$$\eta_{ijl} = \begin{cases} 1 & \text{if } l = i \\ -1 & \text{if } l = j \\ 0 & \text{in other case} \end{cases}$$

Using Restraints

The number of pairs of atoms to be added in the restraint potential(s) is defined by setting the variable number_restr, and a list of distance restrains have to be added to in an additional lio.restrain file. For example:

a_i	a_j	index	k	w_{ij}	l_0
1	2	0	0.1	1.0	7.86
3	4	0	0.1	-1.0	7.86
7	9	1	0.4	2.0	-2.3
13	1	1	0.4	1.0	-2.3
14	3	1	0.4	-3.0	-2.3
14	2	2	0.2	1.0	0.5
8	5	3	0.3	1.0	3.2

Columns a_i and a_j contain the atom numbers in the QM system to be restrained, while the index number determines which distances contribute to a same generalized reaction coordinate. The remaining columns are the force constants (k), weights of that distance in the generalized coordinate (w_{ij}) and equilibrium positions in atomic units (l_0).

Examples

1)In lio.in:

number_restr = 1

in lio.restrain:

a_i	a_j	index	k	w_{ij}	l_0
1	2	0	0.1	1.0	7.86

Potential added to system:

$$U = \frac{1}{2}0.1\Big[1.0|\vec{r_1} - \vec{r_2}| - 7.86\Big]^2 \tag{4.5}$$

2)In lio.in:

number_restr = 2

in lio.restrain:

a_i	a_j	index	k	w_{ij}	l_0
1	2	0	0.1	1.0	7.86
3	4	0	0.1	-1.0	7.86

Potential added to system:

$$U = \frac{1}{2}0.1\Big[1.0|\vec{r_1} - \vec{r_2}| - 1.0|\vec{r_3} - \vec{r_4}| - 7.86\Big]^2 \tag{4.6}$$

3)In lio.in:

number_restr = 4

in lio.restrain:

a_i	a_j	index	k	w_{ij}	l_0
1	2	0	0.1	1.0	7.86
3	4	0	0.1	-1.0	7.86
1	3	1	0.3	3.5	-2.31
7	8	1	0.3	-2.2	-2.31

Potential added to system:

$$U = \frac{1}{2}0.1\Big[1.0|\vec{r_1}-\vec{r_2}|-1.0|\vec{r_3}-\vec{r_4}|-7.86\Big]^2+\frac{1}{2}0.3\Big[3.5|\vec{r_1}-\vec{r_3}|-2.2|\vec{r_7}-\vec{r_8}|+2.31\Big]^2 \tag{4.7}$$

4.4 Changing Lennard-Jones parameters

In QM/MM calculations involving chemical reactions, it my be desirable to have reaction-coordinate dependent Lennard-Jones parameters. The LJ Switch module contained within LIO implements a charge-dependent Lennard-Jones parameters scheme, interpolating said parameters between two reference states (for example, reactants and products). A special input section must be added in the LIO input file, using the follwing structure:

{LJSWITCH}
index Q_1 Q_2 σ_1 σ_2 ϵ_1 ϵ_2
... ...
index Q_1 Q_2 σ_1 σ_2 ϵ_1 ϵ_2
{END}

Each line belongs to a different QM atom, up to the number of desired LJ-switching atoms. No empty lines should be found between the LJSWITCH and END statements. *index* indicates the atom's index in the input files, while Q_i, σ_i, and ϵ_i indicate the

atomic charge and LJ parameters for the reference states. σ_i should be given in A, and ϵ_i in $kJ.mol^{-1}$. The LJ Switch implementation is self-consistent, adding terms to Fock matrix elements, and therefore adding the energy difference to the SCF energy (and not to the classical MM vdW energy).

Special care must be taken when defining the reference states. ϵ and σ might be taken from classical forcefields, but the reference charge should be taken from the mulliken charge in a QM/MM scheme (and not from the forcefields).

Refer to the tests in the /test/AMBER_test/lennard_jones_switch directory for some examples.

Chapter 5

Electron Dynamics

5.1 Real Time TD-DFT

5.2 Electronic transport

5.3 Ehrenfest Dynamics

Chapter 6

Post-Processing Tools

6.1 TD-Analize: Electronic Spectra

6.2 CubeGen: Orbital and Density Visualization

Chapter 7

Reference Section

This section contains a quick reference for all of LIO's input variables and commandline options. For more detailed descriptions, please refer to the previous chapters.

7.1 Command line options

Variable	Description
-i file_name *character*20*	*default = 'lio.in'* Name of the input file containing LIO options.
-c crd_file.xyz *character*20*	*default = 'qm.xyz'* Name of thte XYZ file containing coordinates.
-b basis_file *character*20*	*default = 'basis'* A file containing the basis set and fitting set data, only used when int_basis=f.
-v *logical*	*default = .false.* Sets verbose level to 4.

Table 7.1: Command Line

7.2 Keywords - General Setup

Variable	Description
natom *integer*	*default = 0* Number of QM atoms in the system.
nsol *integer*	*default = 0* Number of classical atoms in the system.
charge *integer*	*default = 0* Total charge of the QM system.
open *logical*	*default = .false.* Perform an open-shell calculation.
nunp *integer*	*default = 0* Number of unpaired electrons for open-shell calculations.
style *logical*	*default = .false.* Activates a formatted version of the output.
fcoord *character*20*	*default = 'qm.xyz'* Name of the output file for the coordinates of the QM system.
writexyz *logical*	*default = .false.* Writes an xyz file containing the QM system.
verbose *integer*	*default = 1* Determines the amount of information printed, from 0 (nothing) to 5 (everything).
timers *integer*	*default = 0* Activates timers (=1 or =2).
dbug *logical*	*default = .false.* Checks for NaNs.
writeForces *logical*	*default = .false.* Writes final forces to an output file.

Table 7.2: General Setup

Variable	Description
dipole *logical*	*default = .false.* Calculates and prints dipole moment.
mulliken *logical*	*default = .false.* Performs a Mulliken Population Analysis.
lowdin *logical*	*default = .false.* Performs a Lowdin Population Analysis.
fukui *logical*	*default = .false.* Calculates atomic Fukui function.
gaussian_convert *logical*	*default = .false.* Reads a Gaussian09 density matrix.
print_coeffs *logical*	*default = .false.* Prints MO coefficients in AO basis.

Table 7.2: General Setup

7.3 Keywords - GPU Options

Variable	Description
gpu_level *integer*	*default = 4* Determines which calculations are performed by the GPU. (0 = only XC, 5 = everything).
max_function_exponent *integer*	*default = 10* Ignore functions with $\|exponent\| > max\_function\_exponent$.
little_cube_size *double precision*	*default = 8.0d0* Small cube-type point group size.
min_points_per_cube *integer*	*default = 1* Minimum number of grid points in a cube.
assign_all_functions *logical*	*default = .false.* Calculate all functions (ignores $max\_function\_exponent$).
sphere_radius *double precision*	*default = 0.6d0* Proportion of points contained in sphere-type groups (from 0 to 1).
remove_zero_weights *logical*	*default = .true.* Discard functions for those whose weight is zero.
energy_all_iterations *logical*	*default = .false.* Calculate Exc energy in all SCF iterations.
free_global_memory *double precision*	*default = 0.0d0* Fraction of global GPU memory available for the calculation (1 means 100%).

Table 7.3: GPU Module Options

7.4 Keywords - DFT Hamiltonian

Variable	Description
iexch *integer*	*default = 9* Identifies the exchange-correlation potential to use with the calculation when not using libxc. Iexch=9 is the only option currently available.
use_libxc *logical*	*default = .false.* Activates the use of libxc version of the XC potential.
ex_functional_id *integer*	*default = ?* Exchange functional to use with libxc.
ec_functional_id *integer*	*default = ?* Correlation functional to use with libxc.
int_basis *logical*	*default = .true.* If true, looks for the internal basis indicated in variables *basis_set* and *fitting_set*; if false, an external basis file must be provided.
basis_set *character*20*	*default = 'DZVP'* Name of the basis set used, or the name of the custom basis set file.
fitting_set *character*100*	*default = 'DZVP Coulomb Fitting'* Name of the fitting set used in the calculation.
n_ghosts *integer*	*default = 0* Number of ghost atoms.
ghost_atoms *integer*	*default = 0* A list containing the ghost atom indeces.
rmax *double precision*	*default = 16.0d0* Maximum exponent in 3-center integrals.
rmaxs *double precision*	*default = 5.0d0* Maximum exponent for 3-center integrals in single precision.

Table 7.4: DFT Hamiltonian

Variable	Description
iGrid *integer*	*default = 2* Grid type when iterating through SCF.
iGrid2 *integer*	*default = 2* Grid type for final energy calculation in SCF.

Table 7.4: DFT Hamiltonian

7.5 Keywords - Effective Core Potentials

Variable	Description
ECPMode *logical*	*default = .false.* Activate effective core potentials.
ECPTypes *integer*	*default = 0* Number of atoms with ECP.
tipeECP *character*30*	*default = 'NOT-DEFINED'* Type of ECP used.
ZListECP *integer*	*default = 0* Array with Z of atoms with ECP enabled.
cutECP *logical*	*default = .true.* Enables cuts for ECP integrals.
cut2_0 *double precision*	*default = 15.d0* Cut value for 2-center ECP integrals.
cut3_0 *double precision*	*default = 12.d0* Cut value for 3-center ECP integrals.
ECP_debug *logical*	*default = .false.* Enables ECP debug mode.
local_nonlocal *integer*	*default = 0* Calculates only local terms (when = 1) or only non-local terms (when = 2).
ECP_full_range_int *logical*	*default = .false.* Enables full-range integral calculations.
verbose_ECP *integer*	*default = 0* Controls ECP verbose levels.
fock_ECP_read *logical*	*default = .false.* Enables restart read in ECP.

Table 7.5: Effective Core Potentials

Variable	Description
fock_ECP_write *logical*	*default = .false.* Enables restart write in ECP.
fullTimer_ECP *logical*	*default = .false.* Enables full timers in ECP.

Table 7.5: Effective Core Potentials

7.6 Keywords - DFTB Embedding

Variable	Description
dftb_calc *logical*	*default = .false.* Activates the TB embedding of the system.
MTB *integer*	*default = 0* TODO Size of the two tight-binding subatrices.
end_bTB *integer*	*default = 0* TODO Index matrix size.
start_tdtb *integer*	*default = 0* TODO Initial time step for evolution of diagonal TB terms (????).
end_tdtb *integer*	*default = 0* TODO Final time step for evolution of diagonal TB terms (????).
alfaTB *double precision*	*default = UNSET* Manually sets the on-site energies (diagonal values) for the TB part of the Hamiltonian.
betaTB *double precision*	*default = UNSET* Manually sets the hopping terms for the TB part of the Hamiltonian (ie, the non-diagonal nearest neighbour terms for TB - TB interactions).
gammaTB *double precision*	*default = UNSET* Manually sets the hopping terms for the interaction between TB atoms and DFT atoms.
Vbias_TB *double precision*	*default = UNSET* Sets a bias for the on-site energies to simulate electrodes.
TBload *logical*	*default = .false.* TODO.
TBsave *logical*	*default = .false.* TODO.

Table 7.6: DFTB Embedding

7.7 Keywords - Fields and Biases

Variable	Description
field *logical*	*default = .false.* Use an external field (perturbation in TD).
a0 *double precision*	*default = 1.0d3* A dividing factor in electric field calculations.
epsilon *double precision*	*default = 1.0d0* Relative permitivity of the medium.
Fx, Fy, Fz *double precision*	*default = 0.05d0* The value of the external electric field in the x, y and z directions.
nfields_iso *integer*	*default = 0* Number of shape-isotropic fields.
field_iso_file *character*20*	*default = 'field.in'* Isotropic fields input file.
nfields_aniso *integer*	*default = 0* Number of shape-anisotropic fields.
field_aniso_file *character*20*	*default = 'field.in'* Anisotropic fields input file.
fockbias_is_active *logical*	*default = .false.* TODO.
fockbias_is_shaped *logical*	*default = .false.* TODO.
fockbias_readfile *character*80*	*default = 'atombias.in'* Atomic bias input file.
fockbias_timeamp0 *double precision*	*default = UNSET* TODO.

Table 7.7: Fields and Biases

Variable	Description
fockbias_timefall *double precision*	*default = UNSET* TODO.
fockbias_timegrow *double precision*	*default = UNSET* TODO.

Table 7.7: Fields and Biases

7.8 Keywords - Self Consistent Field

Variable	Description
initial_guess *integer*	*default = 0* Method for generating the initial guess for the SCF.
nMax *integer*	*default = 100* Maximum number of SCF steps.
told *double precision*	*default = 1.0d-6* Tolerance threshold for density matrix convergence.
Etold *double precision*	*default = 1.0d0* Tolerance threshold for energy convergence.
DIIS *logical*	*default = .true.* Use DIIS convergence accelerator if true, or damping convergence accelerator if false.
nDIIS *integer*	*default = 30* Number of DIIS convergence iterations.
gold *double precision*	*default = 1.0d1* Proportion of old matrix to use when using the damping mixture (gold = X means that it will use an 1:X new to old proportion).
hybrid_converg *logical*	*default = .false.* Use Hybrid convergence accelerator.
good_cut *double precision*	*default = 1.0d-5* Tolerance threshold for damped convergence, switch to DIIS afterwards.
vcinp *logical*	*default = .false.* Reads the molecular orbital coefficients from *frestart* and uses that as the starting guess for the first SCF cycle.
rst_dens *integer*	*default = 0* rst_dens = 1 reads a density matrix restart, while rst_dens = 2 both reads and writes a density matrix restart.

Table 7.8: Self Consistent Field

Variable	Description
frestartin *character*20*	*default = 'restart.in'* Filename for the input containing the molecular orbital coefficients to be used as starting guess when *vcinp = .true.*.
frestart *character*20*	*default = 'restart.out'* Filename for the output containing the molecular orbital coefficients.
restart_freq *integer*	*default = 0* Indicates the frequency for writing the restart: it will do so every set number of calls to LIO (that is, number of steps of nuclear moves performed by the MD-engine.

Table 7.8: Self Consistent Field

7.9 Keywords - Geometry Optimization

Variable	Description
steep *logical*	*default = .false.* Activate steepest descent algorithm for geometry optimization.
Force_cut *double precision*	*default = 5.0d-4* Convergence criteria in forces (Hartree/bohr) for geometry optimization.
Energy_cut *double precision*	*default = 1.0d-4* Convergence criteria in energy (Hartree) for geometry optimization.
minimzation_steep *double precision*	*default = 0.05d0* Initial distance steep (bohr).
n_min_steeps *integer*	*default = 500* Maximum number of geometry optimization steps.
lineal_search *logical*	*default = .true.* Enable lineal search algorithm.
n_points *integer*	*default = 5* Number of points scaned for lineal search.
number_restr *integer*	*default = 0* Number of distance restraints used.

Table 7.9: Geometry Optimization

7.10 Keywords - Real Time TD-DFT

Variable	Description
timeDep *integer*	*default = 0* Use RT-TD-DFT when timeDep = 1.
tdStep *double precision*	*default = 2.0d-5* Timestep for TD-DFT (in atomic units).
ntdStep *integer*	*default = 0* Total number of TD-DFT steps.
propagator *integer*	*default = 1* RT-TD-DFT propagator (1 = Verlet, 2 = Magnus).
NBCH *integer*	*default = 10* Number of [ρ. Fock^{n}] commutators in Magnus.
tdrestart *logical*	*default = .false.* Reads an input restart for TD (named td_in.restart).
td_rst_freq *integer*	*default = 500* Write the TD restart every td_rst_freq steps.
td_do_pop *integer*	*default = 0* Number of step stride in which the pop will be written. (0 means it is never written).
writeDens *logical*	*default = .false.* Writes electronic density to an output file after having finished the calculations.

Table 7.10: Real Time TD-DFT

7.11 Keywords - Transport

Variable	Description
transport_calc *logical*	*default = .false.* TODO.
generate_rho0 *logical*	*default = .false.* TODO.
gate_field *logical*	*default = .false.* TODO.
driving_rate *double precision*	*default = UNSET* TODO.
pop_drive *integer*	*default = UNSET* TODO.
save_charge_freq *integer*	*default = UNSET* TODO.
nbias *integer*	*default = UNSET* TODO.

Table 7.11: Transport

7.12 Keywords - Ehrenfest

Variable	Description
ndyn_steps *integer*	*default = 0* Number of nuclear movement steps.
rsto_nfreq *integer*	*default = 0* Frequency (in steps) in which the restart is printed. (A value of 0 means only written in the end)
rsto_saves *logical*	*default = .false.* TODO.
rsti_loads *logical*	*default = .false.* TODO.
nullify_forces *logical*	*default = .false.* Returns 0 for all forces to the MD engine.
nullify_forces *logical*	*default = .false.* Returns 0 for all forces to the MD engine.
eefld_on *logical*	*default = .false.* TODO.
eefld_timegih *logical*	*default = .false.* TODO.
eefld_timegfh *logical*	*default = .false.* TODO.
eefld_ampx/ampy/ampz *double precision*	*default = 2.D-5* TODO.
eefld_timeamp *double precision*	*default = 2.D-5* TODO.
eefld_timepos *double precision*	*default = 2.D-5* TODO.

Table 7.12: Ehrenfest

Variable	Description
eefld_wavelen *double precision*	*default = 2.D-5* TODO.

Table 7.12: Ehrenfest

7.13 Keywords - CubeGen

Variable	Description
cube_dens *logical*	*default = .false.* Prints the electronic density.
cubeGen_only *logical*	*default = .false.* Avoid running SCF, only do cubeGen from a restart.
cube_res *integer*	*default = 40* Number of voxels per dimension (resolution).
cube_sel *integer*	*default = 0* Select only a particular orbital for printing (0 = all).
cube_dens_file *character*20*	*default = 'dens.cube'* File containing the electronic density.
cube_orb *logical*	*default = .false.* Prints orbital shapes.
cube_sqrt_orb *logical*	*default = .false.* Prints the orbitals square root.
cube_orb_file *character*20*	*default = 'orb.cube'* File containing the orbital shapes.
cube_elec *logical*	*default = .false.* Prints the electric field.
cube_elec_file *character*20*	*default = 'field.cube'* File containing the electrical field.
write1Drho *logical*	*default = .false.* Prints the electronic density integrated in 2 dimentions.
write_int_rho *character*	*default = " "* Selects 1 variable to NOT integrate. Available options are x,y,z,r

Table 7.13: CubeGen

Variable	Description
w_rho_xmin, **w_rho_ymin,** **w_rho_zmin** *double precision*	*default = -5.0* Minimun value of x,y,z in integration
w_rho_rmin *double precision*	*default = 0.0* Minimun value of r in integration
w_rho_xmax, **w_rho_ymax,** **w_rho_zmax,** **w_rho_rmax** *double precision*	*default = 5.0* Maximun value of x,y,z in integration
w_rho_dx, **w_rho_dy,** **w_rho_dz,** **w_rho_dr,** **w_rho_dtheta,** **w_rho_dphi** *double precision*	*default = 0.1* step in x,y,z,r, θ, ϕ

Table 7.13: CubeGen

References

[1] Luis R. Kahn, Paul Baybutt, and Donald G. Truhlar. "Ab initio effective core potentials: Reduction of all-electron molecular structure calculations to calculations involving only valence electrons". In: *The Journal of Chemical Physics* 65.10 (1976), pp. 3826–3853.

[2] Brett M. Bode and Mark S. Gordon. "Fast computation of analytical second derivatives with effective core potentials: Application to Si8C12, Ge8C12, and Sn8C12". In: *The Journal of Chemical Physics* 111.19 (1999), pp. 8778–8784.

[3] N. O. Foglia. "Simulación computacional de dinámica electrónica y reactividad química". PhD thesis. Universidad de Buenos Aires, 2019.

[4] A. Szabo and N.S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. 1st. Mineola: Dover Publications, Inc., 1996.

[5] W. Stevens, H. Basch, and M. Krauss. "Compact effective potentials and efficient shared–exponent basis sets for the first–and second–row atoms". In: *J. Chem. Phys.* 81.12 (1984), pp. 6026–6033.

[6] W. Stevens et al. "Relativistic compact effective potentials and efficient, shared–exponent basis sets for the third–, fourth–, and fifth–row atoms". In: *Can. J. Chemistry* 70.2 (1992), pp. 612–630.

[7] T. Cundari and W. Stevens. "Effective core potential methods for the lanthanides". In: *J. Chem. Phys.* 98.7 (1993), pp. 5555–5565.

[8] L. Fernandez Pacios and P. Christiansen. "*Abinitio* relativistic effective potentials with spin–orbit operators. I. Li through Ar". In: *J. Chem. Phys.* 82.6 (1985), pp. 2664–2671.

[9] M. M. Hurley et al. "Abinitio relativistic effective potentials with spin–orbit operators. II. K through Kr". In: *J. Chem. Phys.* 84.12 (1986), pp. 6840–6853.

[10] L. A. LaJohn et al. "*Abinitio* relativistic effective potentials with spin–orbit operators. III. Rb through Xe". In: *J. Chem. Phys.* 87.5 (1987), pp. 2812–2824.

[11] R. B. Ross et al. "*Abinitio* relativistic effective potentials with spin–orbit operators. IV. Cs through Rn". In: *J. Chem. Phys.* 93.9 (1990), pp. 6654–6670.

[12] P. Hay and W. Wadt. "*Ab initio* effective core potentials for molecular calculations. Potentials for the transition metal atoms Sc to Hg". In: *J. Chem. Phys.* 82.1 (1985), pp. 270–283.

[13] W. Wadt and P. Hay. "*Ab initio* effective core potentials for molecular calculations. Potentials for main group elements Na to Bi". In: *J. Chem. Phys.* 82.1 (1985), pp. 284–298.

[14] P. Hay and W. Wadt. "*Ab initio* effective core potentials for molecular calculations. Potentials for K to Au including the outermost core orbitals". In: *J. Chem. Phys.* 82.1 (1985), pp. 299–310.

[15] M. Dolg et al. "Energy–adjusted *abinitio* pseudopotentials for the first row transition elements". In: *J. Chem. Phys.* 86.2 (1987), pp. 866–872.