Sample rst2pdf doc

version

TDAP

■■ 12, 2020

Contents

TDAP Help

Introduction

Introduction

Time-Dependent DFT

The Time-Dependent Ab-initio Package, or in its acronym 'TDAP', is aimed at carrying out real-time (rt) time-dependent density functional theory (TDDFT) calculations in efficient atomic bases. Such calculations allow first principles treatment of both electron and ion dynamics with non-adiabatic effects fully taken in account, by solving many-electron time-dependent Schr"odinger equation. It enables large-scale first principles simulations of quantum dynamics in finite and extended systems including atoms, clusters, molecules, nanoparticles, solids, defects, interfaces, and two-dimensional materials, significantly reducing the heavy computational cost of typical rt-TDDFT simulations.

Be aware that since the time-dependent Schr"odinger equation is a one-order partial differential equation, it is a initial value problem instead of an eigenvalue problem with specific boundary conditions. Thus, the solution method is to solve a time-evolving equation $\psi(t+dt) = \exp(-idtH)\psi(t)$, instead of diagonalizing H with $H\psi = E\psi$. Some important features of the current TDAP code include:

- Both finite and extended systems can be simulated.
- Both electronic and ionic degree of freedoms are evolved, therefore a complete information on electronic wavefunctions and ionic movements during real time evolution can be provided for simulations of complex materials and rich phenomena far from equilibrium.
- Momentum-resolved real-time evolution of electrons can be simulated, which yields the important momentum resolution and a new degree of freedom, essential to describe key quantities and important physics in photoexcited molecules and condensed matter materials. It also significantly improves the computational efficiency of rt-TDDFT calculations for solids.
- Different from approaches using real space grids and all-electron full-potential linearized augmented-planewaves, the adoption of local atomic basis reduces the number of required basis functions, making rt-TDDFT computation of realistic large systems plausible. In addition, the bases can be defined by users. With a relatively small real-space cutoff for atomic bases, the order-N linear scaling with respect to the system size can be achieved.

Compilation

Compilation

Rather than using the top-level Src directory as building directory, the user has to use an ad-hoc building directory (by default the top-level **Obj** directory, but it can be any (new) directory in the top level). The building directory will hold the object files, module files, and libraries resulting from the compilation of the sources in **Src**. The **VPATH** mechanism of modern **make** programs is used. This scheme has many advantages. Among them:

- The Src directory is kept pristine.
- Many different object directories can be used concurrently to compile the program with different compilers or optimization levels.

The compilation of TDAP is similar to siesta. But there are some tips:

- Use **gcc + Open MPI/MVAPICH** . In some computer compatibility problems will come out if you use Intel compiler. It's a safely strategy by using **Open MPI/MVAPICH** .
- We recommend that you use the ScaLAPACK from Netlib instead of MKL.
- TDAP is parallel version. Don't use a serial compiler.

- If you use Intel's compiler, please remember to change 299th line's code **probe=rand() ** into **call random number(probe)** in decay_test.F90. Different compilers use different standard functions. Nevertheless, incompatibility happens in some machine using Intel compiler.

If you just want to compile the program, go to **Obj** and issue the command:

```
sh ../Src/obj_setup.sh
```

Then configure the install environment:

```
../Src/configure --enable-mpi
```

to populate this directory with the minimal scaffolding of makefiles, and then make sure that you create or generate an appropriate arch.make file. Specify the libraries in parallel mode and make sure you are using the right compiler in arch.make file:

```
FC=mpif90
BLAS_LIBS=~/librefblas.a
LAPACK_LIBS=~/libreflapack.a
BLACS_LIBS=
SCALAPACK_LIBS=~/libscalapack.a
```

Then, type

```
make
```

Now the TDAP program is compiled. To compile utility programs (those living in Util), you can just simply use the provided makefiles, typing

```
make
```

as appropriate.

Input File Description

Input File Description

Time Evolution

Propagation

TDAP is aimed at carrying out real time simulations of electron and nuclear dynamics from first principles. The solution method is to solve a time-evolving Schr"odinger equation $\psi(t+dt) = \exp(-idtH)\psi(t)$, instead of diagonalize H with $H\psi = E\psi$. For this purpose, a set of parameters specified below are needed for real time evolution of quantum states.

SolutionMethod (String)

Character string to chose between diagonalization (**diagon**) or Order-N (**OrderN**) solution of the LDA Hamiltonian or the TDDFT solver (**evolve**).

> ***Note***
>
> Always use **evolve** to do the TDDFT calculation, otherwise all the parameters listed below will not be read.

Default value: diagon

TD.NewInv (Logical)

If set this parameter to be true, the TD evolve operator is $\frac{S - iHdt/2}{}$, else $\frac{1 - iS^{-1}Hdt/2}{}$. The old version of TDAP use $\frac{1 - iS^{-1}Hdt/2}{}$. Now you can use the new faster method by set it to be true. Default value: False

TD.FinalTimeStep (Integer)

Final time step of the TD simulation.

> ***Note***
>
> The number of BOMD steps is max(MD.FinalTimeStep - TD.FinalTimeStep , 1)

Default value: 1

TD.LengthTimeStep (Real Time)

Length of the time step of the TD simulation.

> ***Note***
>
> Best to be the same as MD.LengthTimeStep. Unit in fs.

Default value: 1.0 fs

TD.EmptyBands (Integer)

The number of unoccupied bands to be considered in density matrix calculation, etc.

Default value: 5

Population switch

To prepare an initial excited state to start a time-dependent simulation, one can choose in which states electrons are pumped to another (empty) states. The population switch method allows such electronic transitions from a given set of initial and final states to occur in the simulated systems, with features specified by the user.

TD.PopulationTransition (data block)

Includes arbitrary number of lines. Each line with the format :

```
Start Band -- End band -- Jumped Electrons
```

Start Band is The band from which the electron is excited. If $\searrow$ 0, no excitation occurs. End band is The band to which the electron is excited. Jumped Electrons is the number of electrons pumped from Start Band to End band.

For instance:

```
%block TD.PopulationTransition
-1 2  0.5
-1 1  0.5
%endblock TD.PopulationTransition
```

Means that 0.5 electrons are pumped each from the HOMO band to the LUMO band and the band higher than LUMO.

Default value: No default values

Tuned electric field

The electric field can be tuned to vary across z direction of the cell or to change with time. For basic concepts and cautions in applying electric field in the simulations, please refer to the **ExternalElectricField** tag, which control the value of field intensity E_0. You also can apply vector field by **TD.GaugeField** tag.

For the time-dependent changes, a Gaussian form wavepackage is implemented as: $E(t) = E_0 \cos(2\pi f t)\exp(-\frac{(t-t_0)^2}{\sigma^2})$

ExternalElectricField (data block)

It specifies an external electric field. For instance:

```
%block ExternalElectricField
0.000  0.000  0.0500  V/Ang
%endblock ExternalElectricField
```

Default value: No default values

TD.GaugeField (data block)

It specifies an external vector gauge field. For instance:

```
%block TD.GaugeField
0.000  0.000  0.0500  V/Ang
%endblock TD.GaugeField
```

> ***Warning***
>
> This function is highly experimental. Do not use it now.

Default value: No default values

TD.DeltaElectricField (Logical)

Specify a delta electric field. Then gaussian electric field will be ignored.

Default value: .False.

TD.LightFrequency (Real Frequency)

The frequency f of the electric field. Default value: 0.5 fs

TD.LightInitialTime (Real time)

The initial time t_0 to introduce the electric field.

Default value: 100.0 fs

TD.LightTimeScale (Real time)

The peak width σ of the wavepackage.

Default value: 25.0 fs

Spatially nonuniform electric field

For spatially nonhomogeneous fields, only a function with nonzero values in the selected slab z to z is implemented. Therefore,

$E(t, z) = E(t)\theta(z - z_{\cdot})(z_{\cdot\cdot} - z)$

TD.EfZmin (Real)

The lower boundary of the electric field. Unit in the percentage of the cell vector.

Default value: 0.333

TD.EfZmax (Real)

The upper boundary of the electric field. Unit in percentage of the cell vector.

Default value: 0.666

Analysis

Band projection

TD.WriteDMOfSelectiveOrbitals (Logical)

Whether to output the partial DM of centain band and orbital projection.

Default value: false

TD.PartialDMBand (data block)

Specify the band to output and orbital projection. See also **TD.PartialDMSumOrbitals**.

Multiple lines with each of them include one band index. The corresponding charge density will be projected to the selected orbital blocks, and output to file: chg*Band Index*.txt

For instance:

```
%block TD.PartialDMBand
1
2
%endblock TD.PartialDMBand
```

will analysis the band 1-2 and projected them to the orbital blocks.

Default value: No default values

TD.PartialDMSumOrbitals (data block)

Specify orbital blocks for projection. See also **TD.PartialDMBand**.

Multiple lines with each of them include the start index and the end index of the orbital. The corresponding charge density will be projected to the selected orbital blocks, and output to file: chg*Band Index*.txt

For instance:

```
%block TD.PartialDMSumOrbitals
1 10
11 22
%endblock TD.PartialDMSumOrbitals
```

will analysis the selected band (using **TD.PartialDMBand**) and projected them to the two orbital blocks: 1 to 10 and 11 to 22.

Default value: No default values

Restart a TDDFT-MD calculation

To restart a calculation, the wavefunction and the atomic positions are needed. The atomic positions can be obtained in the file systemLabel.XV, and the wavefunction should be output using the following tag,

If TD.WriteWaveFunctionStep = x, it means to output the wavefunction every x step to the file **systemLabel.step.TDWFSX**. **step** is the certain step at which the wavefunction is output.

Default value: 1000

Whether to read wavefunction from the file **systemLabel.TDWFSX**.

Default value: false

> ***Note***
>
> Note: the systemLabel is not followed by **step**.

For instance:

```
mkdir restartdir
cp H.psf  O.psf  siesta.EIG  siesta.PAR  input.fdf  structure.fdf  siesta.XV \
siesta.VERLET_RESTART siesta.TDWFSX  restartdir
```

Add this in restartdir/input.fdf

```
TD.ReadWaveFunction              T
MD.UsesaveXV T
```

Example

Contributors

Contributors

- Sheng Meng (Institute of Physics, Chinese Academy of Sciences)
- Efthimios Kaxiras (Harvard University)
- Chao Lian (Institute of Physics, Chinese Academy of Sciences)
- Peiwei You (Institute of Physics, Chinese Academy of Sciences)
- Daqiang Chen (Institute of Physics, Chinese Academy of Sciences)
- Jin Zhang (Institute of Physics, Chinese Academy of Sciences)
- Wei Ma (Institute of Physics, Chinese Academy of Sciences)
- Yang Jiao (Institute of Physics, Chinese Academy of Sciences)
- Yi Gao (Institute of Physics, Chinese Academy of Sciences)
- Junhyeok Bang (Korea Basic Science Institute)
- Shengbai Zhang (Rensselaer Polytechnic Institute)
- **genindex**
- **modindex**
- **search**