TDSE Solver Based on Numerical Basis Method

Shaohao Chen

June 12, 2012

This program is for solving time-dependent Schrodinger equation (TDSE) using numerical basis. It can be applied to atoms in laser fields. The related formalism is described in [1]. This document will be focused on how to use the program. All program files are located at */p3data2/common/program-basis*.

1 Program Structure and Compilation

Before solving the TDSE, a numerical basis should be obtained. The program for obtaining the numerical basis is written in Fortran77. The program for solving the TDSE is developed based on the Nonlinear Physics Strong Field (NPSF) lib and the structure is similar. The core part is written in C++ functions or classes and complied to a lib. Users should write their own *main.cpp* file and call appropriate functions for specific problems.

1.1 Numerical Basis

The basis functions could be eigen functions of an atomic Hamiltonian. An exemplary basis is given here. This part of program is in the folder named *obtain-basis*.

nrscf.f : to calculate Hartree-Fock potential for any atom by self-consistent field (SCF) method. (NRSCF=NonRelativistic Self-Consistent Field).

nrwf-inp.f : to generate an input file of basis information for *nrwf.f*.

nrwf.f : to solve the eigen problem of the atomic Hamiltonian with any single-active-electron (SAE) approximation under a certain boundary condition, using the Numerov method. (NRWF=NonRelativistic WaveFunction)

The source files can be compiled using any Fortran compiler, e.g.,

ifort -O2 -o name name.f

1.2 Core Library

The core part of the program is in the folder named *npsflib-basis*. The source files related to the basis method are located at *libbasis*, and the head files are located at *include*.

BasisPropagate.cpp : including most important functions, e.g. calculating the Hamiltonian matrix, propagation of the wavepacket, projecting the wavepacket onto Coulomb wave, etc.

BasisLaser.cpp : to construct the laser pulses.

PETSc.cpp : to define some functions for the convenience of using PETSc.

GNU makefiles are used for creating the lib. All common variables used in Makefiles are defined in *common.mk* in the root directory. The working compiler is g++. To start, one should set the correct path of the package in *common.mk* and then run *make*. The created lib file named *libbasis.a* should be located in the folder *lib*.

1.3 Specific Jobs

After creating *libbasis.a*, one should write a *main.cpp* file and create an executable file for a specific job. Examples of *main.cpp* files and related Makefile and input files are in the folder named *run-basis*.

main-dme.cpp : to obtain the matrix of the time-independent dipole moment.

main-tdse.cpp : to solve the TDSE.

main-cwf.cpp : to project the wavepacket onto Coulomb wavefunctions.

In order to compile the main-*.cpp files, one should first modify the object name and source file name in the Makefile, e.g.,

object.name: source.name.cpp libbasis.a ,

and then run

make object.name .

The executable file named as the *object.name* should be created.

1.4 External Libraries

Some external libraries are used in this program. Corresponding paths have been defined in *common.mk*, which are set to be the correct directories on the photon machines. They should be probably changed on other machines (e.g. yotta).

gsl: to calculate analytical eigen functions of hydrogen atom, including bound states (Legendre functions) and continuum states (Coulomb waves).

PETSc: to parallelize the matrix-vector operations using Crylov sub-space methods based on Message Passing Interface (MPI) scheme. See details at http://www.mcs.anl.gov/petsc/ .

OpenMP: to parallelize some iterations based on OpenMP scheme. It is included in most compilers (e.g. g++) by default.

FFTW: to implement Fast Fourier Transform.

Matlab: for output of wavefunctions in Matlab format. Unnecessary if using normal text format output.

2 Run the Program

A complete run of the program as well as the input/output files will be described in this section.

2.1 Create Basis Functions

In order to create the basis functions, execute the following steps. Run them only once for one kind of atom.

2.1.1 Step 1: run nrscf

In this step, the Hartree-Fock potential for any atom will be calculated by SCF method.

Input files:

nrscf.in : input atomic parameters.

try.pot : initially estimated potential, usually calculated from Fermi distribution.

Specifically for hydrogen atom, the input parameters in the two files can be any values, because the program will output $V(r) = -1/r$ anyway.

Output file:

nadata.pot : storing the Hartree-Fock potential for multielectron atom and $V(r) = -1/r$ for hydrogen atom.

2.1.2 Step 2: run nrwf-inp

In this step, an input file of basis information for step 3 will be generated.

Input files:

inp-nrwf : input parameters of basis functions. The format is as following.

ZE	! nuclear charge
$RLARG$	! the largest distance of the grid (in a.u.)
n_{max}	! maximum number of principle quantum number n
l_{max}	! maximum number of angular momentum l

Output file:

nrwf.in : an input file for step 3, containing the values of n, l and initially estimated energy for each basis function, and total number of basis functions, etc.

2.1.3 Step 3: run nrwf

In this step, eigen energies and functions of the atomic Hamiltonian will be calculated. They will be used as basis functions for solving the TDSE.

Input files:

nadata.pot : obtained from step 1.

nrwf.in : obtained from step 2.

Output files:

inp-orbit : information for basis functions on a nonuniform grid. The format is as following.

column 1:	principle quantum number n.
column 2:	angular momentum l.
column 3:	the largest distance (in a.u.) of the grid for each n and l.
column 4:	number of grids for each n and l.
column 5:	eigen energies E_{nl} (in a.u.).

nrwf.out : basically the same as *inp-orbit*, just in another format. Not used below.

wavefunction : storing the basis functions. The format is as following.

column 1: distance r (in a.u.).
column 2: radial eigen wavefunction $u_{nl}(r)$.

2.2 Obtain Dipole-moment Matrix

In this subsection, the matrix of the time-independent radial dipole moment $< \psi_j|r|\psi_i >$ will be obtained by running the program *main-dme*. Run it only once for one kind of atom. In order to perform the multiplication of two basis wavefunctions, all basis functions are interpolated onto the same uniform grid by spline method.

Input files:
inp-orbit and *wavefunction* : obtained from above.
inp-dme : input parameters. The format is as following.

n_{basis}	! total number of basis, equal to the number of lines in *inp-orbit*.
r_{max}^{new}, dr	! the largest distance and spatial step (in a.u.) of the uniform grid.
wf_folder	! the path to where *inp-orbit* are located.
dme_folder	! the path to where you want to save the ouput files.

Output files:
dme.dat : storing the time-independent radial dipole moment.
wf_new.dat : storing basis wavefunctions on the new uniform grid

2.3 Solve TDSE

In this subsection, the TDSE will be solved and the expansion coefficients at the end of the pulse will be saved, by running *main-tdse*. Run it separately for different laser parameters.

Input files:
inp-orbit and *dme.dat*: obtained from above.
inp-tdse : input parameters. The format is as following.

n_{basis}	! total number of basis.
dt_1, dt_2	! time steps (in a.u.) during and after the laser pulse.
$lambda, i_0, n_{cycle}, cep, t_{wait}$	! wavelength (in nm), intensity (in W/cm^2),
	! number of cycles and CEP (in rad),
	! time (in a.u.) of free propagation after the pulse.
n_{front}, n_{back}	! parameters for shaped envelop of the pulse:
	! sin^{n_f} and sin^{n_b} for front and back parts respectively.
wf_folder	! the path to where *inp-orbit* is located
dme_folder	! the path to where *dme.dat* is located
output_folder	! the path to where you want to put the output files.

Output files:
E_eigen_vec.dat : storing eigen energies E_{nl} (in a.u.).
laser.dat : storing the laser field. The format is as follows.

column 1: time (in a.u.)
column 2: field strength E (in a.u.)
column 3: vector potential A (in a.u.)

psi_vec.dat : storing the coefficients c_{nl} at the end of the pulse.

cn2_vec.dat : storing the populations in each state labeled by n and l (i.e. $|c_{nl}|^2$) at the end of the pulse.

Each line of *E_eigen_vec.dat*, *psi_vec.dat* and *cn2_vec.dat* corresponds to a unique set of n and l. They are sorted in the same order as that in *inp-orbit*.

2.4 Obtain Momentum Distribution

By running *main-cwf*, the wavepacket at the end of the pulse will be projected onto Coulomb wave functions to obtain the momentum distribution of photoelectron.

Input files:

inp-orbit, psi_vec.dat and *wf_new.dat* : obtained from above.

inp-cwf : input parameters. The format is as following.

$n_{basis}, l_{max}^{prj}, n_0$	! total number of basis,
	! maxmun l of Coulomb wavefunctions,
	! number of bound states unnecessary for integrations.
r_{max}^{new}, dr	! the largest distance and spatial step (in a.u.).
k_{max}, dk	! the maximum radial momentum and the momentum step.
n_{theta}	! number of angles equally distributed from 0 to π.
wf_folder	! the path to where *inp-orbit* is located.
dme_folder	! the path to where *dme.dat* is located.
cn_folder	! where *E_eigen_vec.dat* and *cn2_vec.dat* are located.
output_folder	! the path to where you want to put the output files.

Output files:

k_end.dat : storing radial momentum k.

thetak_end.dat : storing angles of momentum distribution θ_k.

P_mom_end.dat : storing momentum distribution calculated by Eq. (11) in [1].

Pl_mom_end.dat : storing momentum distribution for each l.

dP_dk_drho_end.dat : storing doubly differential momentum distribution calculated by Eq. (12) in [1].

The data in *P_mom_end.dat* and *dP_dk_drho_end.dat* are sorted by increasing k first and increasing θ_k later, while that in *Pl_mom_end.dat* is sorted by increasing θ_k first and increasing k later.

References

[1] Shaohao Chen, Xiang Gao, Jiaming Li, Andreas Becker, and Agnieszka Jaron Becker, Phys. Rev. A.