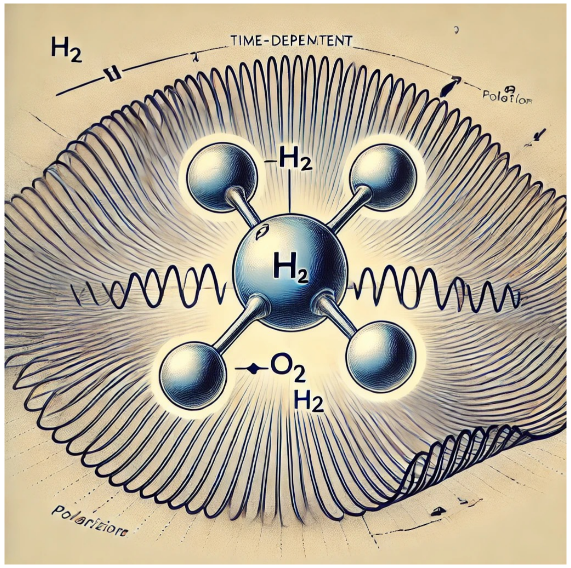
TDCIS: Real-Time Truncated Configuration Interaction Time Evolution Calculator

Lee M. Thompson



Indiana University (June 2024)

What is TDCIS

Introduction

- TDCIS propagates a truncated (nonorthogonal) configuration interaction wavefunction in time using specified initial conditions
- Time-dependence implies 'real' time evolution i.e. we see the evolution of the system as a function of time.
- Explicitly time-dependent approach required when:
  - We are far from equilibrium (stationary states no longer representative)
  - Fields are much stronger than can be accounted for in a perturbative approach (i.e. cannot use response theory)
  - Want to model e.g. intricate pulse shapes and pump-probe delay times
- Computationally, no difficult optimization problems, but need to simulate for a long time to extract accurate information

Tutorial Purpose

- Provide a quick and basic introduction to the theory the code is based on
- Highlight what the code can do and how to use the code for your own investigations
- If there is time — provide more details on code structure

TDCIS Theory

Time-Dependent Schrödinger Equation

- The time-dependent Schrödinger equation (TDSE) describes the time-evolution of the many-electron problem

$$i\hbar\frac{\partial\Psi(t)}{\partial t} = \hat{H}(t)\Psi(t)$$

- For a system in a TD external electric field, the TD Hamiltonian can be written as

$$\hat{H}(t) = \hat{H}_0 - \mathbf{F}(t)\cdot\hat{\boldsymbol{\mu}}$$

- The TD wavefunction can be constructed from a linear expansion of TI solutions

$$\Psi(t) = \sum_A c_A(t)\Psi_A(0)$$

- The TI wavefunctions are obtained from solutions of the TI Schrödinger equation

$$\hat{H}_0\Psi_A(0) = E_I\Psi_A(0)$$

TDCIS Theory

TDCI Equation of Motion

- Substituting time-dependent wavefunction into TDSE and left-projecting by time-independent wavefunction gives the equations of motion to determine the time-evolution of state coefficients

$$i\hbar\frac{\partial c_A(t)}{\partial t} = \sum_B \langle \Psi_A(0)|\hat{H}|\Psi_B(0)\rangle c_B(t)$$

- Solving for the time evolution of the state coefficients can make use of the split-operator approach (assumes commutation of operators)

$$\mathbf{c}(t+\Delta t) = \left[\prod_{q=x,y,z} \mathbf{U}_q^\dagger e^{iF_q(t)\tilde{\boldsymbol{\mu}}_q\Delta t}\mathbf{U}_q\right] e^{-i\mathbf{E}\Delta t}\mathbf{c}(t)$$

- Transformation from the diagonal dipole basis to the state basis performed using unitary transformations $\mathbf{U}_q = \mathbf{d}_q^\dagger \mathbf{D}$ where d and D are the eigenvectors giving the dipole and state eigenstates respectively

$$\Psi_A(0) = \sum_I D_{IA}|\Phi_I\rangle \qquad \Psi_{\tilde{A}}(0) = \sum_I d_{qI\tilde{A}}|\Phi_I\rangle$$

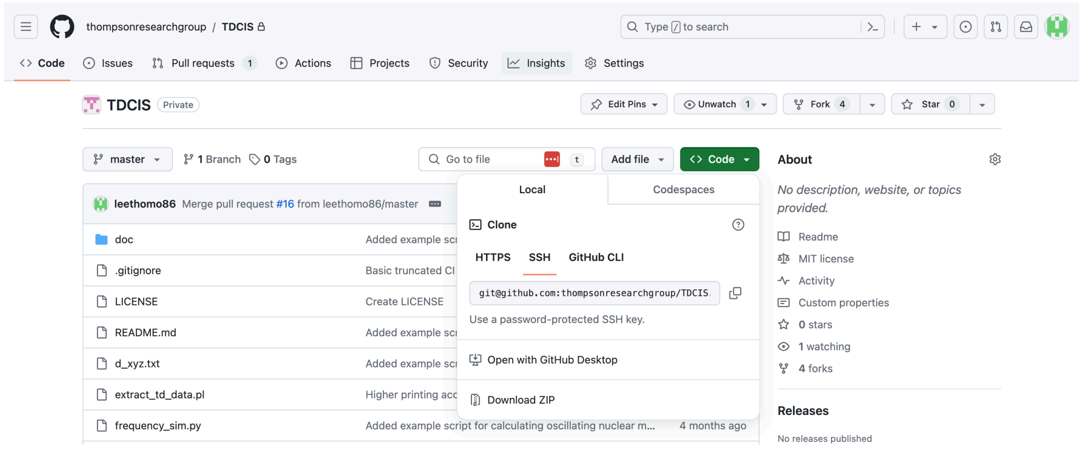
- The CI coefficients at any point in time are given by

$$\mathbf{D}(t) = \sum_A c_A(t)\mathbf{D}_A$$

Getting TDCIS Working

Where to Get TDCIS

- Go to https://github.com/thompsonresearchgroup/TDCIS
- Click on the green code button and download zip or use the address to clone with git



```
git@github.com:thompsonresearchgroup/TDCIS.
```

Use a password-protected SSH key.

No description, website, or topics provided.

leethomo86 Merge pull request #16 from leethomo86/master

doc	Added example scr
.gitignore	Basic truncated CI
LICENSE	Create LICENSE
README.md	Added example scr
d_xyz.txt	Added example scr
extract_td_data.pl	Higher printing acc
frequency_sim.py	Added example script for calculating oscillating nuclear m... 4 months ago

Compiling TDCIS

- TDCIS is built using MQCPack library (https://github.com/MQCPack/mqcPack)
- Edit the makefile

```
mqcroot = "/Path/To/MQCPack" #MQCPack installation directory
blalibs = mkl #mkl or blalap
FC = gfortran #gfortran or pgfortran
```

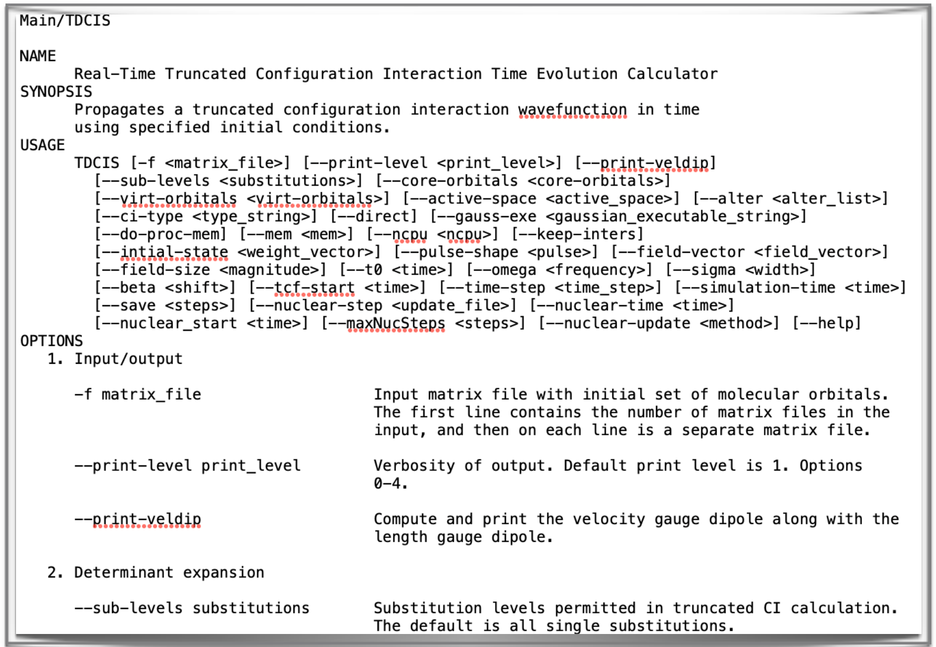
- Type ‘make’ on the command line

How to get started with TDCIS

Getting Help

- Once you have compiled TDCIS, you can view the help page

```
./tdcis.exe --help
```



```
Main/TDCIS

NAME
      Real-Time Truncated Configuration Interaction Time Evolution Calculator
SYNOPSIS
      Propagates a truncated configuration interaction wavefunction in time
      using specified initial conditions.
USAGE
      TDCIS [-f <matrix_file>] [--print-level <print_level>] [--print-veldip]
        [--sub-levels <substitutions>] [--core-orbitals <core-orbitals>]
        [--virt-orbitals <virt-orbitals>] [--active-space <active_space>] [--alter <alter_list>]
        [--ci-type <type_string>] [--direct] [--gauss-exe <gaussian_executable_string>]
        [--do-proc-mem] [--mem <mem>] [--ncpu <ncpu>] [--keep-inters]
        [--intial-state <weight_vector>] [--pulse-shape <pulse>] [--field-vector <field_vector>]
        [--field-size <magnitude>] [--t0 <time>] [--omega <frequency>] [--sigma <width>]
        [--beta <shift>] [--tcf-start <time>] [--time-step <time_step>] [--simulation-time <time>]
        [--save <steps>] [--nuclear-step <update_file>] [--nuclear-time <time>]
        [--nuclear_start <time>] [--maxNucSteps <steps>] [--nuclear-update <method>] [--help]
OPTIONS
   1. Input/output

      -f matrix_file                 Input matrix file with initial set of molecular orbitals.
                                     The first line contains the number of matrix files in the
                                     input, and then on each line is a separate matrix file.

      --print-level print_level      Verbosity of output. Default print level is 1. Options
                                     0-4.

      --print-veldip                 Compute and print the velocity gauge dipole along with the
                                     length gauge dipole.

   2. Determinant expansion

      --sub-levels substitutions     Substitution levels permitted in truncated CI calculation.
                                     The default is all single substitutions.
```

General Command Execution

- Commands are given as options on the command line, e.g.

```
./tdcis.exe -f input.txt --pulse-shape delta > output.txt
```

- All options and their usage can be found in the help document

How to use TDCIS

Basic Input Options

- TDCIS always requires an input file, where:
  - The first line is the number of files listed in the input
  - The remaining lines give the names of the matrix files providing integrals, molecular orbital coefficients etc.
  - Example

```
4
matrix_file_1.mat
matrix_file_2.mat
matrix_file_3.mat
matrix_file_4.mat
```

- Print level (0-4) changes output information
  - TDCIS can produce big files so be careful, but obviously make sure to print what you need!
- Matrix files can be obtained from Gaussian 16
  - Example

```
%chk=matrix_file_1.chk
#P hf/sto-3g geom=allcheck nosymm output=matrix scf=conven

matrix_file_1.mat
```

Determining the TDCIS wave function

Wavefunction Types

- TDCIS propagates the CI coefficients of multideterminant wave function expansions
- Orbital time propagation coming soon!
- Different types of wave functions can be selected (`--ci-type`)
  - Orthogonal CI — only one matrix file allowed in the input
    - Complete Active Space (`ocas`)
      - `--active-space [2,2]` specifies 2 electrons in 2 orbitals
      - `--alter-list [3a4,2b4:5a6]` swaps orbitals from input
    - Truncated CI (`oci`)
      - `--sub-levels [0,1,2]` specifies reference, singles and doubles substitutions
      - `--core-orbitals N` removes occupied orbitals in expansion
      - `--virt-orbitals N` removes virtual orbitals in expansion
  - Nonorthogonal CI — each determinant must be specified as file in input (`noci`)
    - Script to generate NOCIS matrix files included (nocis_generator.pl)
  - Hybrid CI (nonorthogonal coupled orthogonal expansions) coming soon!

Building TDCIS matrix elements

Conventional vs. Direct Matrix Elements

- The quickest way to calculate is to use conventional matrix elements (default)
  - Requires two-electron integrals to be stored on the first matrix file specified in input
  - If your molecule is large, two-electron integrals may take up a lot of space
- Direct matrix elements use Gaussian to compute two-electron integrals as needed so they are not stored on disk (`--direct`)
  - Requires modifications to Gaussian that are not distributed with TDCIS
  - Need to recompile Gaussian with modifications
  - Builds Gaussian input file to run calculation so additional options available

`--gauss-exe gaussian_executable`	Executable and command line flags
`--do-proc-mem`	Add %mem and %nproc lines
`--mem mem`	Amount of memory to add (8GB default)
`--ncpu ncpu`	Number of processors to add (2 default)
`--keep-inters`	Keep intemediate files for debugging

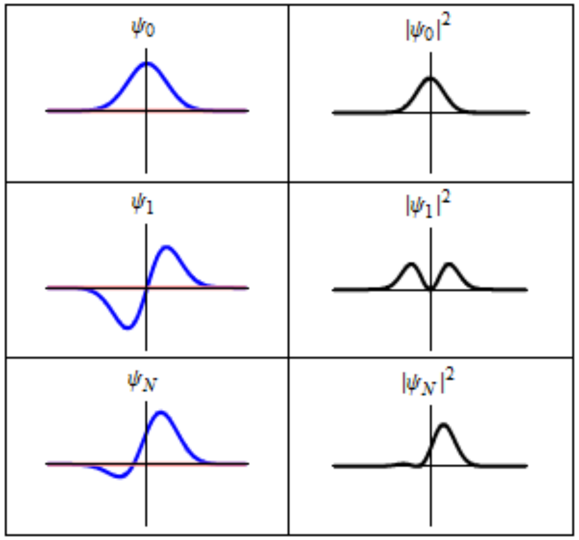
Initial TDCIS wavefunction

Setting initial conditions

- Initially all stationary states of wave function are computed
- Can specify whether to start in ground state (default), excited state, or superposition state

```
--initial-state [1,0.5:2,0.5]
```

- Note that only pure states can be studied with TDCIS (no mixed states)
  - Future plans to include mixed states and decoherence



Applying a field in TDCIS

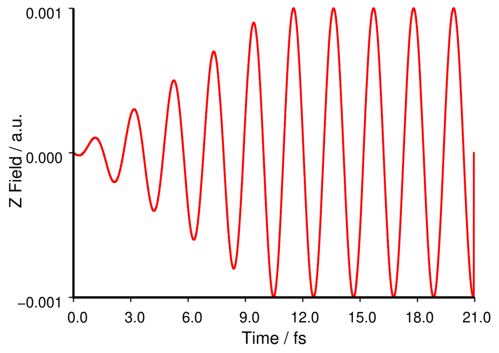
Pulse Types Available

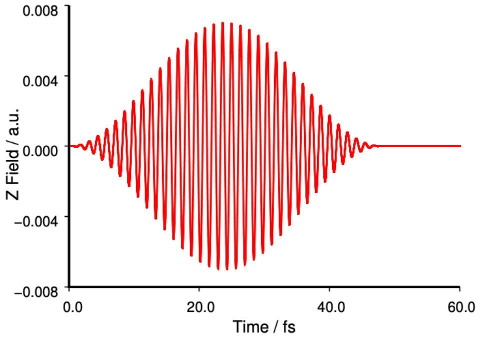
```
--pulse-shape 'transform limited'
```

Name	Equation
Rectangle	$\mathbf{E}(t) = \mathbf{E}_{max} H(t - t_0) H((t_0 + \sigma) - t)$
Delta	$\mathbf{E}(t) = \mathbf{E}_{max} \delta(t - t_0)$
Continuous	$\mathbf{E}(t) = \mathbf{E}_{max} \sin(\omega(t - t_0)) H(t - t_0) H((t_0 + \sigma) - t)$
Transform Limited	$\mathbf{E}(t) = \mathbf{E}_{max} \exp\left(\frac{-(t-t_0)^2}{2\sigma^2}\right) \sin(\omega(t - t_0))$
Chirped Pulse	$\mathbf{E}(t) = \mathbf{E}_{max} \exp\left(\frac{-(t-t_0)^2}{2\sigma^2}\right) \sin((\omega + \beta(t - t_0))(t - t_0))$

t_0 – pulse onset time, ω – pulse frequency, σ – pulse frequency,
β – chirp parameter, $H(x)$ – Heaviside function, $\delta(x)$ – delta function

- Rectangle is a homogeneous field
- Also a cos square pulse available
- Need to properly add option for ramp time





Pulse Parameters

`--field-vector [0.5,0,-0.5]`	Orientation of field vector (z-axis default)
`--field-size 0.0`	*E*E_{max} (default is 0.005 au = 25.7 MV/cm = 1.752 W/m^2)
`--t0 10.0`	t_0 Pulse onset or maximum (0.0 default)
`--omega 5.0`	ω Pulse frequency (10.0 default)
`--sigma 0.1`	σ Pulse width (0.5 default)
`--beta 2.0`	β Chirp parameter (3.0 default

Simulation Parameters in TDCIS

Simulation Parameters

- Time step for electron dynamics must be small enough for accurate numerical integration but larger is less computationally demanding

  `--time-step 0.1` (default 0.1 au = 2.42 as)

- Simulation time option gives total time of simulation

  `--simulation-time 100.0` (default is 100.0 au = 2.42 fs)

Additional Outputs

- Time correlation function can be calculated using option:

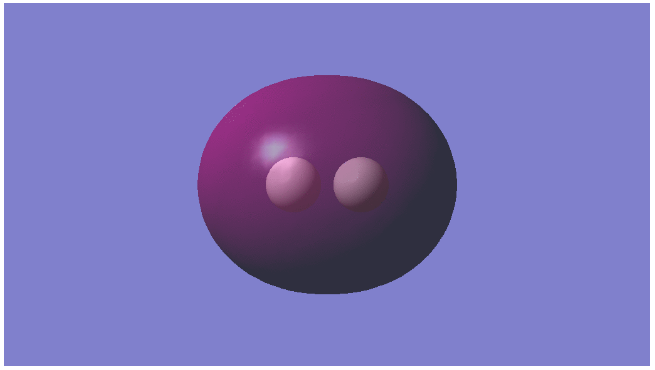
  `--tcf-start time`

  The Fourier transform of this function when the start time is the time step after the delta pulse provides the absorption spectrum.

- The density matrix can be saved to a matrix file every N steps (default 0 means no matrix files are saved) using option:

  `--save steps`

  This option allows for the density matrix to be analyzed/ visualized



10 au movie; $A_z \sin(\omega t)$ pulse, A_z = 1 au, ω = 10 au and field applied for 0.5 au.

Changing External Potential in TDCIS

Updating the Nuclear Potential

- Changing the position of nuclei can be performed semi-classically
- Requires recalculating integrals and the stationary state wavefunctions so expensive (generally this is the most expensive part of the calculation)
- All changes in external potential require update file to be written

```
--nuclear-step file_name
```

- Gaussian is called to recompute integrals. Parameters to control how often Gaussian is called are:

`--nuclear-time time`	Time interval for update (never updated default)
`--nuclear-start 10.0`	Time to start nuclear updates (0.0 default)
`--maxNucSteps N`	Maximum number of steps to apply nuclear update (default until end of simulation)

- Whatever is in the nuclear update file is the Δ applied at each time step
  - A python script (or any language in fact) can be called to change the nuclear update file given nuclear step filename and current simulation time as inputs (`--function-script filename`)
  - Additionally, a file with state populations can be written that the file can read – we can call have the script call Gaussian to get nuclear forces for each electronic state and update the Δ file with the weighted average to do Ehrenfest dynamics (need to push changes to main repository)

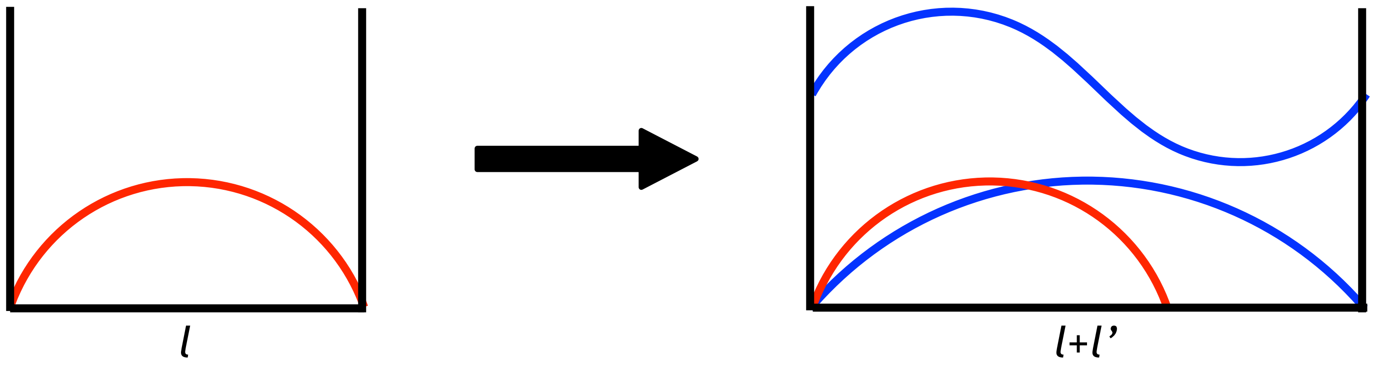
Changing External Potential in TDCIS

Example Script

- An example function script is included in TDCIS (frequency_sim.py) which can be used to simulate normal mode vibrations during the electron dynamics
- For electron detachment simulated by a point charge ejected with constant frequency the script is not required but could be used to modify the velocity depending on the calculated interaction with the molecule

Wavefunction Update Approximations

- When the nuclear potential is updated, there are two options for determining how the wavefunction before the update maps onto the wavefunction after the update (`--nuclear-update nuclear_update`):
  - Sudden (default) Wavefunction does not respond to perturbation
  - Adiabatic Wavefunction adapts to perturbation



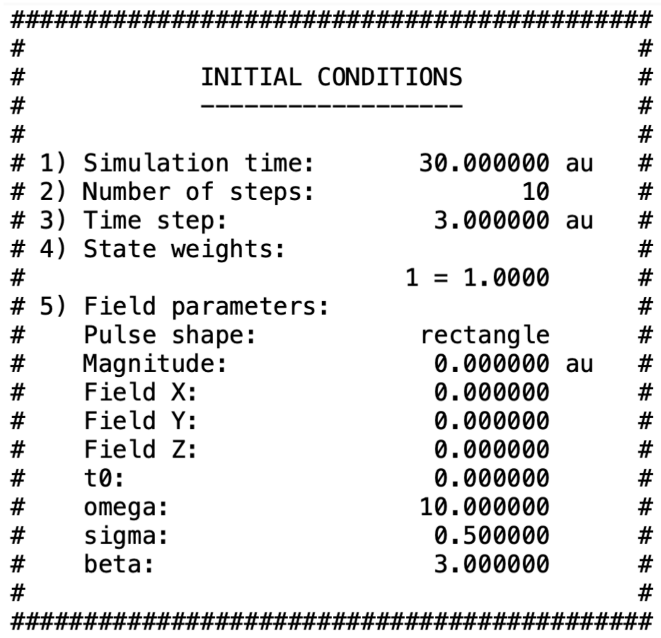
TDCIS Output

Printing Level

- The printing level will significantly changed what is printed. You should use a small molecule to play around and see what printing level is required to print what you need. Here we focus on the basic output

Parameter Information

- At the top of the output file you will find a table of the simulation parameters



```
############################################
#                                          #
#            INITIAL CONDITIONS            #
#            ------------------            #
#                                          #
# 1) Simulation time:       30.000000 au   #
# 2) Number of steps:              10      #
# 3) Time step:              3.000000 au   #
# 4) State weights:                        #
#                          1 = 1.0000      #
# 5) Field parameters:                     #
#    Pulse shape:           rectangle      #
#    Magnitude:              0.000000 au   #
#    Field X:                0.000000      #
#    Field Y:                0.000000      #
#    Field Z:                0.000000      #
#    t0:                     0.000000      #
#    omega:                 10.000000      #
#    sigma:                  0.500000      #
#    beta:                   3.000000      #
#                                          #
############################################
```

TDCIS Output

Example Time Step

- By default, the time, energy, field vector applied, and the molecular dipole are printed at each step
- The state density matrix is printed only at the end of the simulation by default

```
Time step:     0.000000 au

Applied field vector
      1      0.0000000000
      2      0.0000000000
      3      0.0000000000

Energy (au) =  -101.98561625

Total dipole
      1     -0.0000000000
      2     -0.0000000000
      3     -0.0000000000
```

- A script to extract field vector and dipole moment at each time point included in TDCIS files (extract_td_data.pl)

Example Simulation

Running TDCIS Simulation of Point Charge Ejection

1. Get the MOs of the N-electron system

```
%chk=ethene1.chk
#P uhf/6-31G nosymm

Title Card Required

0 1
C                   0.00000000   -0.67759997  -0.00000000
H                   0.92414474   -1.21655197   0.00000000
H                  -0.92414474   -1.21655197   0.00000000
C                  -0.00000000    0.67759997  -0.00000000
H                  -0.92414474    1.21655197   0.00000000
H                   0.92414474    1.21655197  -0.00000000
```

Example Simulation

Running TDCIS Simulation of Point Charge Ejection

2. Build the N-1 electron system with the MOs of the N-electron system (i.e. remove an electron from the HOMO)

```
%oldchk=ethene1.chk
%chk=ethene2.chk
#P uhf/6-31G guess=(read) scf=skip nosymm

Title Card Required

1 2
C                    0.00000000   -0.67759997  -0.00000000
H                    0.92414474   -1.21655197   0.00000000
H                   -0.92414474   -1.21655197   0.00000000
C                   -0.00000000    0.67759997  -0.00000000
H                   -0.92414474    1.21655197   0.00000000
H                    0.92414474    1.21655197  -0.00000000
```

Example Simulation

Running TDCIS Simulation of Point Charge Ejection

3. Output a matrix file with the point charge at the initial position

```
%oldchk=ethene2.chk
%chk=ethene3.chk
#P uhf/6-31G guess=read scf=(skip,conven) int=noraff
output=matrix nosymm

Title Card Required

1 2
C                  0.00000000  -0.67759997  -0.00000000
H                  0.92414474  -1.21655197   0.00000000
H                 -0.92414474  -1.21655197   0.00000000
C                 -0.00000000   0.67759997  -0.00000000
H                 -0.92414474   1.21655197   0.00000000
H                  0.92414474   1.21655197  -0.00000000
Bq(znuc=-1.0)      0.0          0.0          0.0

ethene.mat
```

Example Simulation

Running TDCIS Simulation of Point Charge Ejection

4. Create the input file input.txt

```
1
ethene.mat
```

5. Create the nuclear update file nuc.txt

```
7

Update for point charge ejection from ethene

0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.00000000
0.0  0.00000000   0.00000000   0.01000000
```

File information

Number of atoms

Change in charge

Change in coordinates

Example Simulation

Running TDCIS Simulation of Point Charge Ejection

6. Run the simulation

```
~/Documents/TDCIS/tdcis.exe -f input.txt --print-level 1 --sub-
levels [1] --initial-state [1,1.0] --field-size 0.0 --simulation-
time 50.0 --nuclear-step nuc.txt --nuclear-time 0.5 > output.txt &
```

Example Analysis

Producing Density Difference Animations

1. Run TDCIS as before but saving the density at each point in time

```
~/Documents/TDCIS/tdcis.exe -f input.txt --print-level 1 --sub-
levels [1] --initial-state [1,1.0] --field-size 0.0 --simulation-
time 50.0 --nuclear-step nuc.txt --nuclear-time 0.5 --save 1 >
output.txt &
```

2. Create cube files with Gaussian with the same grid dimensions in all files

```
for i in {0..100}; do for j in {0..5}; do cubegen 1 density=scf MO-
matrix-$i-$j.fchk NEWCUBE/MO-matrix-$i-$j.cube -1 h MO-
matrix-100-0.cube; done; done &
```

3. Subtract cube file at each point in time from initial density to get density difference cube files

```
for i in $(ls *.cube); do echo -e
"su\ninitial.cube\ny\n$i\ny\nDIFFERENCE/Diff_$i\ny\n" | cubman;
done &
```

4. In Gaussview, open all cube files in a single molecule group. Press the Green button in the upper left of the window to run the animation through all time steps (the surface is only built on the frames you have viewed 🤷‍♂️). If you have a Bq atom and you want to visualize its position, you will need to change the atom number from zero in the cube file before loading (change it from 0 to e.g. 2, which is helium)

Example Analysis

Producing Density Difference Animations

5. Go to Results→Surfaces, click on the radio button 'Apply actions to molecule group' and then click 'Surface Actions' and then 'New Surface'
6. Wait for the surfaces to build (could take a while depending on number of frames and size of molecule) then save (better to use screen capture than Gaussview's 'Capture Video' option.