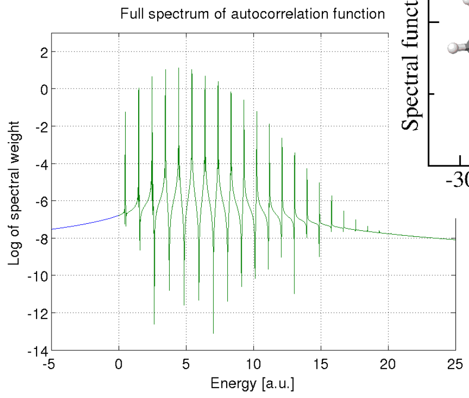
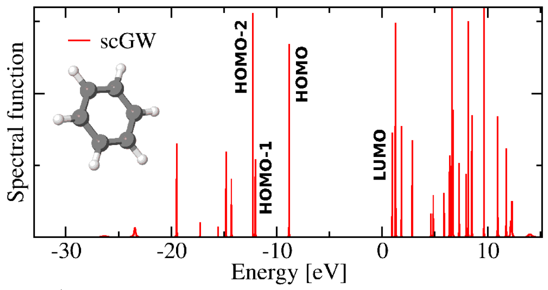
Hands-On Tutorial on *Ab Initio* Molecular Simulations

Berlin, July 12 – 21, 2011





Tutorial VI: Theoretical Spectroscopy and Electronic Excitations

Manuscript for Exercise Problems

Prepared by Fabio Caruso, Patrick Rinke, and Heiko Appel
Fritz-Haber-Institut der Max-Planck-Gesellschaft, Berlin
July 19, 2011

Practical Session VI - Theoretical Spectroscopy and Electronic Excitations

This tutorial is divided into two parts. The first part is concerned with the description of charged electronic excitations. In the second part we turn our attention to real-time propagations of electronic wavefunctions and the description of neutral electronic excitations.

Charged electronic excitations

The fist part of this tutorial is intended to show the adequacy and accuracy of Density-Functional Theory (DFT), Hartree-Fock (HF), and Many-Body Perturbation Theory (MBPT) in the GW approximation for the calculation of electronic excitations. In this tutorial, most of the calculation will be performed on the same system, the nitrogen dimer N_2, with the purpose of comparing the performance of different theoretical approaches with experimental references. Hence we invite you to organize the results of each exercise in a text file, composing a table similar to the following one:

N_2	Experiment	DFT (PBE)	Hartree-Fock	Δ-SCF	G_0W_0@PBE	G_0W_0@HF	scGW
HOMO	15.580 eV						
HOMO-1	16.926 eV			***			
HOMO-3	18.751 eV			***			

Optional:

H_2O	Experiment	DFT (PBE)	Hartree-Fock	Δ-SCF	G_0W_0@PBE	G_0W_0@HF	scGW
HOMO	12.615 eV						
HOMO-1	14.729 eV			***			
HOMO-2	18.550 eV			***			

A list of experimental ionization energies for N_2 and H_2O can be found for instance in reference [1]. Note that since the HOMO-1 and HOMO-2 levels are degenerate in N_2, the third experimental ionization energy has to be compared with the HOMO-3.

Exercise 1: inadequacy of DFT eigenvalues for describing charged electronic excitations

[*Total time for this exercise: 10 minutes. Total CPU time:* < 1 *minute.*]

In this first exercise, you will perform a Kohn-Sham DFT calculation with the PBE exchange-correlation functional and Tier 2 basis (*tight* settings) set for the nitrogen dimer N_2. The quantities of interest in this case are the Kohn-Sham eigenvalues. You can proceed as follows:

- Create the directory ∼`/exercise1`
- Create the `geometry.in` file with the experimental geometry of N_2:

```
atom 0.00000 0.00000 0.00000 N
atom 0.00000 0.00000 1.098 N
```

- Following the template for the `control.in` file available in the path `/pub/tutorial6/exercise_1_DFT_eigenvalue` run a spin-unpolarized DFT calculation using a *pbe* exchange correlation functional (flag: `xc pbe`).
- Compare the first three non-degenerate KS eigenvalues – corresponding to the highest occupied molecular orbital (HOMO), HOMO-1 and HOMO-3 – with the first three experimental ionization energies in [1].

Exercise 2: Electron removal energies from Hartree-Fock.

[*Total time for this exercise: 10 minutes. Total CPU time:* <1 *minute.*]

Modifying the input files of exercise 1, set up a Hartree-Fock calculation by using the flag

```
xc      hf
```

in the `control.in` file and compare the Hartree-Fock eigenvalues with the experimental ionization energies and with previous results from PBE and delta-SCF. Note that already for a small molecules such as N_2, the different treatment of exchange and correlation (the latter absent in HF) may lead to differences in the orbital energy ordering between DFT and HF.

Exercise 3: Electron removal energies from delta-SCF

[*Total time for this exercise: 10 minutes. Total CPU time:* <1 *minute.*]

In this exercise, the ionization energies (I) of N_2 will be evaluated with the delta-SCF approach [2]. This means that the ionization energy is obtained from two separate DFT PBE and HF calculation through total energy differences between the neutral and positively charged species using the formula:

$$I = E_{tot}^{PBE}(N-1) - E_{tot}^{PBE}(N),$$

and similarly for Hartree-Fock, where N is number of electrons of the neutral molecule, $E_{tot}^{PBE}(N)$ is the PBE total energy of the neutral molecule and $E_{tot}^{PBE}(N-1)$ is the PBE total energy of the molecule with a removed electron. Analogously, one can use the delta-SCF method to evaluate the electron affinity (A) as:

$$A = E_{tot}^{PBE}(N) - E_{tot}^{PBE}(N+1).$$

To evaluate Eq.(1), $E_{tot}^{PBE}(N)$ can be extracted from the output file of exercise 1; in addition we need to compute $E_{tot}^{PBE}(N-1)$, which requires an additional DFT calculation. You can proceed according to the following guidelines:

- Create the directory ∼`/exercise3` and copy the input files from Exercise 1.
- Modify the `control.in` file and set the flag for performing a spin-polarized calculation. Specify the initial spin of the calculation and the charge of the molecule:

```
xc          pbe
spin         collinear
```

```
default_initial_moment  1
charge                 +1
```

Compute the ionization energy and the electron affinity of N_2 using Eq.1.

- How does this values compare to the bare PBE eigenvalue and to experiments?
- What is the origin of the difference between the Hartree-Fock eigenvalue and Δ-SCF value for the ionization energy?

Exercise 4: perturbative G_0W_0 and the quasi-particle correction

An improved description of charged electronic excitations is obtained by perturbative inclusion of many-body effects through the many-body self-energy Σ. In the *GW* approximation [2] the self-energy is calculated as:

$$\Sigma^{GW}(\mathbf{r},\mathbf{r}',\omega) = \frac{i}{2\pi}\int d\omega' G(\mathbf{r},\mathbf{r}',\omega')W(\mathbf{r},\mathbf{r}',\omega'+\omega), \tag{1}$$

where $G(\mathbf{r},\mathbf{r}',\omega)$ is the one-particle Green's function and $W(\mathbf{r},\mathbf{r}',\omega)$ is the screened Coulomb interaction (see e.g. [3] for details). The *GW* self-energy can be used to perturbatively correct the DFT or HF eigenvalues by means of the linearized quasi-particle equation:

$$\epsilon_i^{QP} = \epsilon_i^{KS} - \left\langle \psi_i^{KS} \right| \hat{V}_{xc}^{KS} - \hat{\Sigma}^{GW}(\epsilon_i^{QP}) \left| \psi_i^{KS} \right\rangle. \tag{2}$$

This approximation is known as G_0W_0 or one-shot GW, because the self-energy is calculated only once, whereas a more rigorous approach would require a fully self-consistent evaluation of Σ. Since the quasi-particle energies in Eq.(2) are evaluated *perturbatively* on top of a preliminary single-particle calculation (generally DFT or Hartree-Fock),the G_0W_0 approach strongly depends on the starting point. In the following we refer to PBE and Hartree-Fock based G_0W_0 as G_0W_0@PBE and G_0W_0@HF respectively to distinguish between the different starting points. In numerical implementations, the treatment of the full frequency dependence of the self-energy $\Sigma(\mathbf{r},\mathbf{r}',\omega)$ is faced by introducing an additional discrete grid for the frequency ω. Although the default parameters in FHI-aims are safe in most cases, the frequency grid can be adjusted by explicitly setting the number of frequency points N_ω (flag: `frequency_points`) and the extension of the frequency grid ω_{max} (flag: `maximum_frequency`) to check the appropriate convergence of the calculation[1].

Exercise 4.1: G_0W_0 @PBE quasiparticle energies

[*Total time for this exercise: 10 minutes. Total CPU time:* <1 *minute.*]

The purpose of this exercise is to perform a G_0W_0 calculation for the quasi-particle energies of the nitrogen dimer.

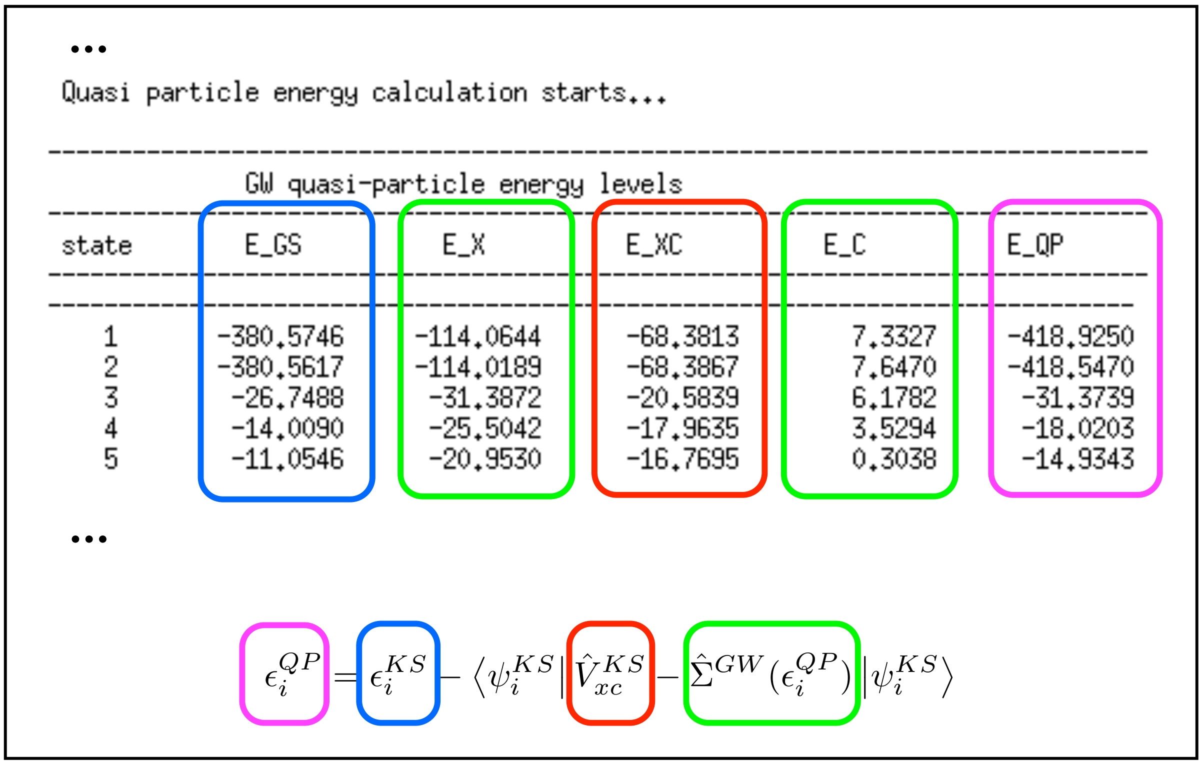
For this exercise, proceed according to the following steps:

- Create the directory ∼`/tutorial6/exercise4/N2` ;
- Copy input files from exercise 1;
- Modify the `control.in` file including the following flags:

```
xc          pbe
qpe_calc    gw
KS_method   lapack
```

[1]We refer to the FHI-aims manual for additional information.

Figure 1: Sample output of G_0W_0 quasiparticle calculation in FHI-aims. The different colors relate terms in the output file to the corresponding quantities in the linearized quasi-particle equation (2).



```
...
Quasi particle energy calculation starts...

---------------------------------------------------------------------------
                  GW quasi-particle energy levels
---------------------------------------------------------------------------
state           E_GS            E_X            E_XC            E_C          E_QP
---------------------------------------------------------------------------
---------------------------------------------------------------------------
   1        -380.5746      -114.0644       -68.3813         7.3327     -418.9250
   2        -380.5617      -114.0189       -68.3867         7.6470     -418.5470
   3         -26.7488       -31.3872       -20.5839         6.1782      -31.3739
   4         -14.0090       -25.5042       -17.9635         3.5294      -18.0203
   5         -11.0546       -20.9530       -16.7695         0.3038      -14.9343

...
```

$$\boxed{\epsilon_i^{QP}} = \boxed{\epsilon_i^{KS}} - \langle \psi_i^{KS} | \boxed{\hat{V}_{xc}^{KS}} - \boxed{\hat{\Sigma}^{GW}(\epsilon_i^{QP})} | \psi_i^{KS} \rangle$$

In the species settings at the end of the control.in file modify the following flags:

```
cut_pot            6.0  2.0  1.0
basis_dep_cutoff   0
```

In addition to the output of the DFT PBE calculation, the output file will contain a table – similar to that in Fig.1 – with the quasi-particle corrections to the single-particle eigenvalues. Extract the quasi-particle energies for the HOMO, HOMO-1 and HOMO-3 levels and compare them with previous results.

Exercise 4.2: G_0W_0 basis set convergence

[*Total time for this exercise: 15 minutes. Total CPU time:* ~ 6 *minutes.*]

Plot the convergence of the first G_0W_0@PBE quasi-particle energy (i.e. the G_0W_0@PBE HOMO level) for the nitrogen dimer using the Tier 1, Tier 2, Tier 3 and Tier 4 basis sets. Calculations with the Tier 4 basis set will require the following additional settings in the control.in file to overcome ill-conditioning of the overlap matrix between basis functions due to the large basis set:

```
basis_threshold 1.e-4
override_illconditioning .true.
```

- How does the G_0W_0@PBE ionization energy convergence compare to the PBE HOMO level?
- What is the origin of the qualitative differences between the PBE and G_0W_0@PBE convergence trends?
- Optional: plot the convergence of the first ionization energy of N_2 for cup_pot= $1.0, 2.0, \ldots, 6.0$ with a Tier 2 basis set[2].

Exercise 4.3: G_0W_0@HF and G_0W_0@PBE0: Dependence on the starting point

[*Total time for this exercise: 10 minutes. Total CPU time:* < 1 *minute.*]

The purpose of this exercise is to show one of the limitations of the perturbative G_0W_0 method, i.e. the dependence on the starting point. Following the steps given in the previous part of this exercise, perform a G_0W_0 calculation using a Hartree-Fock and a PBE0 starting point (i.e. set `xc hf` and `xc pbe0` respectively in the control.in file), and compare the HOMO, HOMO-1 and HOMO-3 quasi-particle energy of the N_2 dimer with G_0W_0@PBE.

Exercise 5: Self-consistent GW

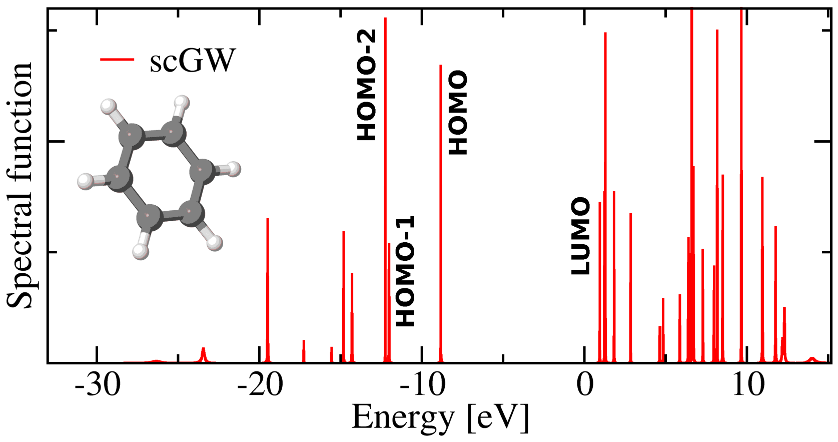
In this exercise, you will perform a fully self-consistent GW calculation. Differently from G_0W_0, the Green's function is calculated by solving the Dyson's equation self-consistently:

$$G = G_0 + G_0 \Sigma G \tag{3}$$

where G_0 is the Green's function of a non-interacting reference system (e.g. DFT, HF, etc.) and G is the Green's function of the interacting system, see e.g. ref.[4] for an introduction.

[2] Note, that in FHI-aims the default settings for basis sets and integration grids are tuned to optimize the performance of LDA and GGA density functional calculations. Calculations beyond plain DFT, may require an additional tuning of such settings.

Figure 2: Example of the spectral function calculated from a self-consistent *GW* Green's function for the Benzene molecule. Since the Green's function has poles at the addition/removal energies of electrons, the position of each peak in the spectral function can be associated with these addition and removal energies.



Exercise 5.1: Spectral function from the self-consistent Green's function

[*Total time for this exercise: 15 minutes. Total CPU time:* ∼ 3 *minutes.*]

To perform a self-consistent GW calculation for N_2, create a new directory and copy the input files from exercise 1. Modify the first part of the *control.in* file including the flags:

```
xc                pbe
sc_self_energy    scgw
spin              none
KS_method         lapack

output cube total_density
cube origin 0.0 0.0 0.55
cube edge 1 0.01 0.0 0.0
cube edge 400 0.0 0.005 0.0
cube edge 500 0.0 0.0 0.005
```

and **choose Tier 1** settings at the bottom of the `control.in` file. The last 5 lines will enable the total density plot which will be used in the next part of the exercise.

After running FHI-aims, in addition to the usual output file, the file `spectrum_sc.dat` will be created [3]. The file `spectrum_sc.dat` contains the spectral function calculated from the self-consistent *GW* Green's function according to the formula:

$$A(\omega) = -\frac{1}{\pi}\int \lim_{\mathbf{r}'\to\mathbf{r}} ImG(\mathbf{r},\mathbf{r}',\omega)d\mathbf{r}. \tag{4}$$

where G has been determined self-consistently from Eq.3. Fig.2 reports an example of a self-consistent *GW* spectral function. You can visualize the spectral function using `xmgrace`, `gnuplot` or any other available plotting tool[4]. The first three ionization energies of N_2 must be extract directly from the position of the peaks in the spectral function.

[3] Note, that the choice of a spin polarized calculation will produce spin-resolved spectral function –identical for closed shell systems, such as N_2 and H_2O – named `spectrum_sc_up.dat` and `spectrum_sc_do.dat` for the each component of the spin moment.

[4] The gnuplot script `/pub/tutorial6/exercise_5_scGW/N2/plot.plt` (run the command `gnuplot plot.plt` after copying the script in your working directory) permits to automatically generate a plot of the spectral function.

Exercise 5.2: GW density vs PBE density

[*Total time for this exercise: 20 minutes. Total CPU time: ∼ 3 minutes.*]
In addition to the spectral function, the self-consistent GW calculation of the previous exercise has produced a file named `cube_001_total_density.cube`. This file contains the total density of the system calculated from the self-consistent Green's function using the relation:

$$\rho(\mathbf{r}) = -\int_{-\infty}^{\mu} \frac{d\omega}{\pi} \lim_{\mathbf{r}' \to \mathbf{r}} ImG(\mathbf{r}, \mathbf{r}', \omega), \tag{5}$$

where μ is the chemical potential. To compare the density at the self-consistent GW level with the PBE density, perform an additional calculation following these steps:

- Rename the file `cube_001_total_density.cube` to `scGW_dens.cube` to avoid to overwrite it.
- Comment the flag `sc_self_energy scgw` in the `control.in` and run FHI-aims.
- Rename the file `cube_001_total_density.cube` to `PBE_dens.cube`
- Copy the script `/pub/tutorial6/exercise_5_scGW/N2/subtract_cubes.py` to your working directory
- Run the command:

```
python subtract_cubes.py  scGW_dens.cube  PBE_dens.cube  scGW-PBE_dens.cube
```

At this point, the file `scGW-PBE_dens.cube` contains the difference between the self-consistent GW density and the PBE density. Following the prescription of Tutorial 1 visualize `scGW-PBE_dens.cube` cube file, or in alternative use the script `/pub/tutorial6/exercise_5_scGW/N2/plot-density.sh` (run the command `./plot-density.sh <filename.cube>`) to generate automatically a plot of the density difference between sc-GW and PBE.

- What is the effect of self-consistent GW on the electronic density?

Optional

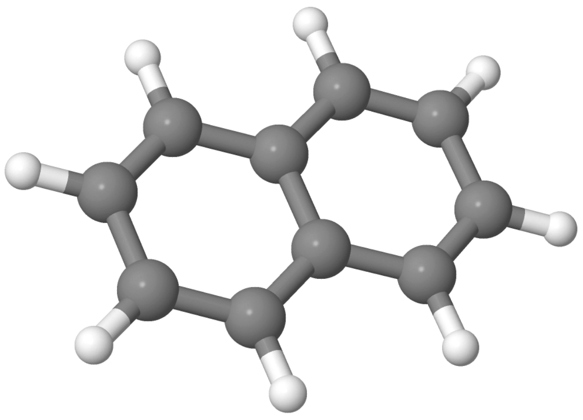
- Visualization: Using the data collected in the previous exercises for the three highest occupied states (HOMO, HOMO-1 and HOMO-3) of N_2, visualize in a plot the deviation from the experimental ionization energy for DFT, delta-SCF, Hartree-Fock, G_0W_0@PBE and self-consistent GW.
- Independence on the starting point at self-consistency: in a different folder, perform a second self-consistent GW calculation for N_2 choosing Hartree-Fock as starting point of the calculation and compare the spectral functions obtained in the previous exercise, in which a PBE starting point was used.

Bonus exercise: GW and the self-interaction error

Several pathologies of (semi-)local functionals, such as LDA and PBE, arise from an approximate treatment of the Fock (or exact exchange) operator in the exchange-correlation functional, which leads to the so called self-interaction error. The self-interaction error is particularly large in localized states, whereas it plays a minor effect in delocalized states. Molecules presenting both localized and delocalized state energetically close (as for instance aromatic molecules), are particularly problematic for LDA and PBE. In such systems, the self-interaction error affects differently the localized and delocalized states, leading to a wrong energetic ordering of the single particle orbitals. In this exercise we want to show how G_0W_0 is able to establish the correct energetic ordering by mean of a proper treatment of the Fock operator in the GW self-energy in Eq.(1).

Perform a G_0W_0 calculation based on an LDA calculation for the Naphtalene molecule ($C_{10}H_8$) molecule

Figure 3: The Naphtalene molecule.



using the `geometry.in` and control.in available in the directory `/pub/tutorial6/bonus_exercise_Naphtalene/`.

- Compare the energetic ordering of DFT orbitals and the G_0W_0 quasi-particle energies of Naphtalene.
- Plot the orbitals 27 and 28 using the following settings in the control.in file:

  ```
  output cube eigenstate  27
  output cube eigenstate  28
  ```

  then run the script `/pub/tutorial6/bonus_exercise_Naphtalene/plot-eigenstate.sh` in your working directory to produce `png` images of the Kohn-Sham eigenstates.
- How many orbitals are energetically swapped in G_0W_0@LDA compared to LDA?
- Which orbital is more localized: the 27 or the 28?
- Is the different localization of orbitals 27 and 28 consistent with the removal of the self-interaction error and the new energetic ordering?

Real-time evolution and optical absorption spectroscopy

This part of the tutorial is covering neutral electronic excitations which arise when a system under consideration is exposed to e.g. a time-dependent external electromagnetic field. The tutorial consists of two parts: the first part introduces several real-time evolution algorithms which allow to propagate electronic wave packets in real-time. The purpose of the first exercise is to familiarize you with wavepacket propagations in real-time and to compare the stability and efficiency of different propagation algorithms. To that end we employ a few `Matlab`/Octave[5] scripts which easily allow to investigate the structure of the propagation algorithms. Once we are familiar with real-time propagations we move to the second part of the exercise where we propagate the time-dependent Kohn-Sham equations in real-time in order to compute the density response of the test molecules from the previous sections (N_2 and H_2O).

Comparison of algorithms for electronic real-time evolution

[*Total time for this exercise: 50 minutes. Total CPU time:* ~ 30 *minutes.*]

The time evolution of a quantum mechanical state vector $|\Psi(t)\rangle$ can be expressed in terms of the time-ordered evolution operator

$$\hat{U}(t+\Delta t, t) = \hat{T} \exp\left(-i \int_t^{t+\Delta t} \hat{H}(\tau) d\tau\right), \tag{6}$$

where $\hat{H}(t)$ denotes the (possibly) time-dependent Hamiltonian of the system. With help of $\hat{U}(t+\Delta t, t)$ the state $|\Psi(t)\rangle$ can be propagated forward in time according to

$$|\Psi(t+\Delta t)\rangle = \hat{U}(t+\Delta t, t)|\Psi(t)\rangle. \tag{7}$$

Practical propagation schemes are therefore concerned with approximate representations of $\hat{U}(t+\Delta t, t)$, or to be more precise, with approximate ways to compute the action of $\hat{U}(t+\Delta t, t)$ on a state vector (with or without explicitly constructing the operator $\hat{U}(t+\Delta t, t)$).

Magnus expansion

The first complication that arises in approximating $\hat{U}(t+\Delta t, t)$ is the time-ordering operator $\hat{T}$ in Eq. (6). The Magnus series [6,7] provides an exact expression for the time-evolution operator Eq. (6) as a *time-unordered* exponential of so called Magnus operators Ω_j in the form

$$\hat{U}(t+\Delta t, t) = \exp\left(\hat{\Omega}\right) = \exp\left(\hat{\Omega}_1 + \hat{\Omega}_2 + \hat{\Omega}_3 + \cdots\right), \qquad \hat{\Omega} = \sum_{j=1}^{\infty} \hat{\Omega}_j, \tag{8}$$

where the $\hat{\Omega}_j$ are given in terms of time-integrals over nested commutators of the Hamiltonian at different points in time

$$\begin{aligned} \hat{\Omega}_1 &= -i \int_t^{t+\Delta t} \hat{H}(\tau) d\tau \\ \hat{\Omega}_2 &= \int_t^{t+\Delta t} \int_t^{\tau_1} [\hat{H}(\tau_1), \hat{H}(\tau_2)] d\tau_2 d\tau_1 \\ &\vdots \end{aligned} \tag{9}$$

The time-integrals can be evaluated numerically with e.g., a Gauss-Legendre quadrature. In the simplest case, which is accurate up to second order in the time-step Δt, one arrives at the exponential midpoint

[5]Matlab is a powerful commerical package tailored for numerical computations. The package uses a own scripting language which has similarities to Fortran and C. GNU Octave is a open source implementation that can be used to execute Matlab codes

rule

$$\begin{aligned} \hat{U}^{(2)}(t+\Delta t, t) &= \exp\left(\hat{\Omega}_1\right) + O(\Delta t^3) \\ \hat{\Omega}_1 &= -i\hat{H}(t+\Delta t/2)\Delta t + O(\Delta t^3). \end{aligned} \tag{10}$$

Higher order truncations of the Magnus series can easily be taken into account but require the computation of additional commutators of the Hamiltonian at different points in time. For example, the fourth order Magnus propagator takes the form

$$\begin{aligned} \hat{U}^{(4)}(t+\Delta t, t) &= \exp\left(\hat{\Omega}_1 + \Omega_2\right) + O(\Delta t^5) \\ \hat{\Omega}_1 &= -i(\hat{H}(\tau_1) + \hat{H}(\tau_2))\frac{\Delta t}{2} + O(\Delta t^5). \\ \hat{\Omega}_2 &= -i[\hat{H}(\tau_1), \hat{H}(\tau_2)]\frac{\sqrt{3}\Delta t^2}{12} + O(\Delta t^5). \\ \tau_{1,2} &= t + (\frac{1}{2} \pm \frac{\sqrt{3}}{6})\Delta t \end{aligned} \tag{11}$$

Exponential of a matrix

When a discretized representation of the state vectors and the Hamiltonian is introduced (e.g. basis set, or finite difference approximation) the propagation step, Eq. (7) in combination with the Magnus expansion, Eq. (8), amounts to compute the action of a matrix exponential on a vector. In the following we discuss different ways to compute this operation.

- *Taylor expansion.* The simplest approach is to perform a Taylor expansion of the exponential

  $$\exp(\hat{\Omega}) = 1 + \hat{\Omega} + \hat{\Omega}^2/2 + \hat{\Omega}^3/6 + \hat{\Omega}^4/24 + O(\hat{\Omega}^5). \tag{12}$$

  Acting with this approximation on a state vector involves only matrix vector products which can be computed efficiently. From a stability analysis, it turns out that an expansion up to fourth order gives the best compromise of computational cost and stability.

- *Matrix diagonalization.* If we have all eigenvalues of the matrix representation of $\hat{\Omega}$ at hand, we can simply take the exponential of the eigenvalues and transform back to the original non-diagonal frame of the matrix. This approach gives the most accurate way to compute the matrix exponential. The price to pay is the large computational cost for the dense matrix diagonalization and the required matrix-matrix products.

- *Padé approximation.* The lowest order approximation of the exponential by rational functions takes the form

  $$\exp(\hat{\Omega}) \approx \frac{1 + \hat{\Omega}/2}{1 - \hat{\Omega}/2}. \tag{13}$$

  This leaves us with an operator (or in discretized form with a matrix) in the denominator. However, since we are only interested in the action of the approximated exponential on a state vector this can be written as

  $$(1 - \hat{\Omega}/2)|\Psi(t+\Delta t)\rangle = (1 + \hat{\Omega}/2)|\Psi(t)\rangle. \tag{14}$$

  For a given point in time t, we can explicitly construct the right hand side of Eq. (14). The state vector at time $t + \Delta t$ can then be found by solving a linear system of equations. This scheme is also known as Crank Nicholson or Cayley approximation.

Standard ODE solver

In case we only consider a discretization of the state vectors and the Hamiltonian, we can regard the time-dependent Schrödinger equation as set of coupled first order ordinary differential equations in time

$$\frac{d}{dt}\Psi_p(t) = \sum_q -iH_{p,q}(t)\Psi_q(t). \tag{15}$$

For each given index p this corresponds to one ordinary differential equation (ODE) which is coupled to the other ODEs through the matrix elements $H_{p,q}(t)$ of the Hamiltonian. In this form standard ODE solver can be employed. Particularly simple ODE time-stepping schemes are Runge-Kutta methods. A frequently employed discretization is the Runge-Kutta method of fourth order

$$\begin{align}
y' &= f(t,y), \qquad y(t_0) = y_0 \tag{16}\\
k_1 &= hf(t_n, y_n) \tag{17}\\
k_2 &= hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \tag{18}\\
k_3 &= hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \tag{19}\\
k_4 &= hf(t_n + h, y_n + k_3) \tag{20}\\
y_{n+1} &= y_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \tag{21}\\
t_{n+1} &= t_n + h \tag{22}
\end{align}$$

which in terms of our TDSE notation can be written as

$$\begin{align}
\Phi_p^{(1)} &= \sum_q -i\Delta t H_{p,q}(t_n)\Psi_q^{(n)} \tag{23}\\
\Phi_p^{(2)} &= \sum_q -i\Delta t H_{p,q}(t_n + \frac{1}{2}\Delta t)[\Psi_q^{(n)} + \frac{1}{2}\Phi_q^{(1)}] \tag{24}\\
\Phi_p^{(3)} &= \sum_q -i\Delta t H_{p,q}(t_n + \frac{1}{2}\Delta t)[\Psi_q^{(n)} + \frac{1}{2}\Phi_q^{(2)}] \tag{25}\\
\Phi_p^{(4)} &= \sum_q -i\Delta t H_{p,q}(t_n + \Delta t)[\Psi_q^{(n)} + \Phi_q^{(3)}] \tag{26}\\
\Psi_p^{(n+1)} &= \Psi_p^{(n)} + \frac{1}{6}[\Phi_p^{(1)} + 2\Phi_p^{(2)} + 2\Phi_p^{(3)} + \Phi_p^{(4)}] \tag{27}\\
t_{n+1} &= t_n + \Delta t \tag{28}
\end{align}$$

Although such a Runge-Kutta algorithm is easy to implement it does not numerically exhibit the time-reversal symmetry which is a property of the exact solution of the TDSE (provided magnetic fields are absent).

Autocorrelation function

One possibility to compute the spectrum of the Hamiltonian $\hat{H}$ via real-time propagation relies on the autocorrelation function of the propagated state vector. The free time-evolution (without external field) of a state vector is given as

$$|\Psi(t)\rangle = \sum_q a_q|\Psi_q\rangle \exp(-iE_q t), \tag{29}$$

where $|\Psi_q\rangle$ and E_q are the solutions of the static Schrödinger equation

$$\hat{H}|\Psi_q\rangle = E_q|\Psi_q\rangle \tag{30}$$

and a_q are time-independent expansion coefficients. Next, let us consider the time-dependent correlation function

$$P(t) = \langle \Psi(t=0) | \Psi(t) \rangle. \tag{31}$$

Using Eq. (29) this can be written as

$$P(t) = \sum_q |a_q|^2 \exp(-iE_q t). \tag{32}$$

By computing the Fourier Transform of the correlation function $P(t)$ we arrive at a spectrum which has peaks at the eigenenergies of the system

$$P(E) = \sum_q |a_q|^2 \delta(E - E_q). \tag{33}$$

The peaks in the spectrum appear infinitely sharp since we assumed a infinite-length time-record of the autocorrelation function $P(t)$. If only a finite-length time-record is available, then a broadening of the peaks occurs.

Exercise 6: comparison of different real-time propagation algorithms

In this exercise we compare the efficiency and accuracy of the different propagation schemes that have been introduced in the previous section.

In the folder

```
        /pub/tutorial6/exercise_6_propagation_algorithms
```

you can find a few Matlab/Octave [6] scripts which contain a simple implementation of the discussed algorithms. Copy these files to your current working directory

```
> cp -a /pub/tutorial6/exercise_6_propagation_algorithms/*.m .
> ls -1 *.m
autocorrelation_spectrum.m
double_well_hamiltonian.m
matrix_exponential_taylor.m
oscillator_hamiltonian.m
propagation_algorithms.m
real_space_grid.m
td_hamiltonian.m
```

For simplicity, the implementation of our example is using a one-dimensional finite-difference representation of Hamiltonians and state vectors. The main program is contained in the file `propagation_algorithms.m`. The files `double_well_hamiltonian.m` and `oscillator_hamiltonian.m` contain different definitions of the system Hamiltonian (asymmetric double well and harmonic oscillator), the file `real_space_grid.m` contains the definition of the employed real-space grid and `td_hamiltonian.m` allows to construct explicitly time-dependent Hamiltonians. Take a look at all files to familiarize yourself with the propagation algorithms.

To start the program you have to execute

```
> octave propagation_algorithms.m
```

or alternatively you can use the octave shell

[6] Matlab is a powerful numerical computing environment. The package allows for easy matrix and vector manipulations and provides extensive plotting routines for numerical data. GNU Octave is an open source implementation of the Matlab computing environment and programming language.

```
> octave
octave:1> propagation_algorithms
```

The numerical settings can be adapted by editing the script `propagation_algorithms.m`. In the file you can find the parameters for the grid spacing, number of time steps, etc.

```
max_time_steps = 4096; % maximum number of time steps
dt = 0.024;            % time step
nx = 100;              % number of grid points
dx = 0.16;             % grid spacing
```

You can switch between different propagation schemes by adjusting the flags for the propagators

```
% propagation algorithms
rti_cn             = true;   % Crank-Nicholson/Cayley propagator
rti_rk4            = false;  % Runge-Kutta 4th order
rti_taylor         = false;  % Taylor expansion of exponential
rti_expm           = false;  % Exact matrix expoential
rti_etrs           = false;  % Enforced time-reversal symmetric propagator
rti_magnus2        = false;  % Second order Magnus expansion with exact matrix expoential
rti_magnus2_taylor = false;  % Second order Magnus expansion with Taylor expansion of exponential
```

Only one `rti_*` variable should be set to true for a given run.

In the following steps we compare two different propagation schemes (Crank-Nicholson/Cayley and Runge-Kutta 4th order). The initial Hamiltonian corresponds to a finite-difference discretization of a one-dimensional harmonic oscillator.

1. Run the script `propagation_algorithms.m` with the default parameters. Once the run finished, inspect the generated PNG files by typing in the shell

   ```
   > qiv *.png
   ```

   Press the space bar to switch to the next image. You will see among others the spectrum, Eq. (33), of the autocorrelation function, Eq. (31), the lowest four eigenstates of the Hamiltonian plotted together with the potential, also the sparsity pattern of the Hamiltonian matrix (zero elements are white, non-zero elements are blue) and the sparsity pattern of the exponential of the Hamiltonian.

2. Compute the autocorrelation function for a different initial condition. For example, change the parameter `xmin` in line 106 of the file `propagation_algorithms.m` from 1.0 to a value of 4.0. How many spectral lines can be resolved in both cases? What is causing the difference?

3. Following the formulas of Eqs. (23) - (28), try to add the fourth-order Runge-Kutta time-stepping to the provided file `propagation_algorithms.m`. As a hint you can find Eq. (23) and Eq. (24) as example in the script

   ```
       rk1 = -sqrt(-1) * dt * Hm_t     * (psi);
       rk2 = -sqrt(-1) * dt * Hm_t_dt2 * (psi + rk1/2);
   % >>>>>>>> Add missing lines
   %   rk3 =
   %   rk4 =
   % <<<<<<<< Add missing lines
       psi = psi + (rk1 + 2*rk2 + 2*rk3 + rk4)/6;
   ```

   Run the Runge-Kutta time-stepping by adjusting the corresponding flags:

   ```
   % propagation algorithms
   rti_cn             = false;   % Crank-Nicholson/Cayley propagator
   rti_rk4            = true;    % Runge-Kutta 4th order
   ```

Can you still reproduce the same spectrum for the autocorrelation function? Which algorithm runs faster, Runge-Kutta, or Crank-Nicholson/Cayley? Is the norm of the wavefunction still preserved when you increase the time-step for Runge-Kutta from `dt = 0.027` to `dt = 0.028`? What can you say about the stability of the Runge-Kutta and Crank-Nicholson/Cayley schemes for larger time steps with `dt > 0.027`?

4. By changing the parameters `dt` for the time-step and `nx` for the grid-spacing, try to investigate for the Crank-Nicholson/Cayley and the Runge-Kutta propagation scheme the following questions:

   - Which method allows for the largest time step?
   - Which method allows for the fastest time stepping?
   - How is the grid spacing related to the time step?

   The stability of a given propagator can be monitored by plotting the energy and norm of the state as function of time. If the norm is not approximately unity (e.g. starts to blow up and eventually shows `NaN` (not a number)), the numerical propagation is not usable anymore.

5. Optional: If time permits also compare to other propagation schemes, e.g. the exact matrix exponential (Option `rti_expm`).

6. Enable the frame output by setting

   ```
   td_frame_output = true;       % output frames for wave packet movie
   ```

   If this is enabled, the script `propagation_algorithms.m` also generates a movie of the computed wavepacket propagation. Also set

   ```
   tdmod = 8;
   xmin = 1.0;
   ```

   and run the script. You can display the time-evolving wavefunction by typing in the shell e.g.

   ```
   > mplayer wave_packet.avi
   ```

   Modify the initial condition of the time-propagation by changing the center and the "squeezing" parameter of the Gaussian initial state.

   ```
   % initial state ((shifted and squeezed) Gaussian at minimum of potential)
   % coherent state (xmin != 0.0, squeeze = 1.0)
   % squeezed state (xmin != 0.0, squeeze != 1.0)
   xmin = 0.5;
   squeeze = 2.0;
   psi = sqrt(dx)*(mass*(omega/squeeze) / pi)^(1/4)*exp(-0.5*mass*(omega/squeeze)*(x+xmin).^2);
   ```

   How does the observed wave-packet propagation compare to a classical oscillator?

7. Switch to a Hamiltonian for an asymmetric double well by commenting the line containing `oscillator_hamiltonian` and uncommenting the line containing `double_well_hamiltonian`

   ```
   [Hm, vpot] = double_well_hamiltonian(nx, dx, x, B, w0, beta);
   %[Hm, vpot] = oscillator_hamiltonian(nx, dx, x, mass, omega);
   ```

   Run the script and inspect again the spectrum of the autocorrelation function and the movie of the wavepacket propagation.

Exercise 7: Optical absorption spectra from TDDFT real-time propagations

[*Total time for this exercise: 40 minutes. Total CPU time:* ~ 30 *minutes.*]

In this exercise we compute optical absorption spectra for H_2O (and optional N_2) using a real-time propagation of the time-dependent Kohn-Sham equations.

To enable the real-time propagation in FHI-aims, you have to add the keyword `TDDFT_run` to the input file. This keyword takes as argument the total propagation time, the time step and two further parameters which determine the amount of information that is printed on the screen (`write info`) and the output frequency (`output every`). Setting `output every` to zero disables file output whereas a value of ten would result in file output after every ten time steps.

```
#
# Time propagation
#
#  mode      total time    dt    write info   output every
  TDDFT_run     2000      0.05       1             0
```

Besides the modification of the `control.in` file, there is also a modification of the `geometry.in` file required:

```
atom                -1.551007  -0.114520   0.000000    O
atom                -1.934259   0.762503   0.000000    H
atom                -0.599677   0.040712   0.000000    H
homogeneous_field  0.000000   0.100000   0.000000
```

By adding a homogeneous field to the coordinate file, FHI-aims computes a ground state in the presence of a static electric field. The Kohn-Sham orbitals of this perturbed ground-state are then propagated in time with a Kohn-Sham Hamiltonian that does not contain the static electric field anymore (`homogeneous_field` is set to zero during `TDDFT_run`). In other words, we propagate a perturbed initial state with a free Hamiltonian (free in the sense of no applied external fields). Since the initial state is perturbed this propagation exhibits density fluctuations. Fourier transforming the time-dependent dipole of these density oscillations allows then to compute the frequency-dependent polarizability of the system

$$\alpha_{ij}(\omega) = -\frac{2}{E}\int \rho_1^{(i)}(\mathbf{r},\omega) r_j \, d^3r \qquad i,j = x,y,z, \tag{34}$$

or likewise the photoabsorption cross section

$$\sigma_{ij} = \frac{4\pi\omega}{c}\Im\alpha_{ij}(\omega). \tag{35}$$

In this notation the component (i) denotes the direction in which the homogeneous field has been applied and (j) denotes the component of the dipole moment which is monitored in the propagation. By performing three independent propagations with perturbations (i) along the x, y, and z coordinates respectively, we can compute the full tensor of the frequency-dependent polarizability.

Run FHI-aims for this input by executing

```
> mpirun -np 4 aims.hands-on-2011.scalapack.mpi.x > optical_absorption_H2O.out &
> tail -f optical_absorption_H2O.out
```

After the run finished sucessfully, the dipole moments and the energy of the system can be extracted from the output of FHI-aims via e.g.

```
> grep 'Electronic dipole moment' optical_absorption_H2O.out    \
    | awk '{print NR" "$3" "$10" "$11" "$12" "$5}'              \
    | head -n 16384 > multipoles.out
```

To compute the photoabsorption cross section you can perform the following steps

```
> cat multipoles.header multipoles.out > multipoles
> cross-section
```

This will create a file `cross_section_vector` which contains the different components of the cross section vector for the chosen perturbation direction.

References

[1] Chong, Gritsenko and Baerends, *J. Chem. Phys.* **116**, 1760 (2002);
[2] Bagus, *Phys. Rev.* **139**, A619 (1965);
[3] Hedin, *Phys. Rev.* **139**, A796 (1965);
[4] Aryasetiawan and Gunnarsson, *Rep. Prog. Phys.* **61**, 237 (1998).
[5] R.D. Mattuck, A guide to Feynman diagrams in the Many-Body problem, Dover Books.
[6] W. Magnus, Comm. Pure Appl. Math. 7 (1954) 649.
[7] A. Castro et. al., J. Chem. Phys. 121, 3425 (2004)