# Assignment 4 – Computational Materials and Molecular Physics
# First hand-in

Eric Lindgren – CID: ericlin

February 28, 2020

## 1 Task 1: Casida with GPAW

I implemented the Casida algorithm for TDDFT in accordance with the problem description and the GPAW example [1]. I used the $Na_8$ cluster as given in the course git repository. For implementation specific details, see appendix A.1. The acquired photoabsorption spectrum, calculated with a Gaussian fold width of 0.06, is given in figure 1. We note that the $Na_8$ cluster seems to have it's most probable transition at slightly less than 3 eV.

Studying the Kohn-Sham eigenvalues for the in total 110 bands of the system, we identify the highest occupied band, the fourth, to have eigenvalue $-3.1297868$ eV. In the ground state only the first four bands are occupied because of the system having eight valence electrons, thus leading to four spin-paired states [1]. We also find the highest unnoccupied band to be the 100th, with eigenvalue 2.95714849 eV. Here we omit the 10 highest band due to them not having been fully converged, and thus their eigenvalues are not fully reliable. This yields the largest Kohn-Sham eigenvalue difference to be 6.08693529 eV. This value corresponds to the "largest" Kohn-Sham transition, i.e. a transition when the electrons of the system are interpreted as being non-interacting. Thus we don't expect these transitions to match the photoabsorption spectrum in figure 1, since the whole point of TDDFT is to correct this simplified Kohn-Sham picture. An intersting observation is that we see a small peak in the photoabsorption spectrum at around the highest Kohn-Sham transition energy of $\sim 6$ eV, even though we only considered transitions up to 6 eV when computing the spectrum. However, I consider this a fluke; a more probable cause of the peak could be due to errors when convoluting the Casida transitions with gaussians in order to go from the discrete spectrum to the continuous spectrum in figure 1 (see task 2).

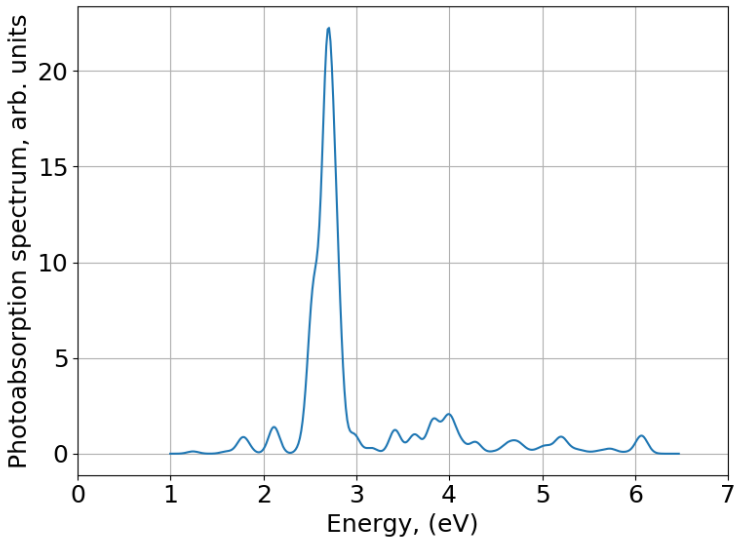


Figure 1: Photoabsorption spectrum as obtained with the Casida method in GPAW. Note that the most probable transition for $Na_8$ seems to lie at around 3 eV.

## 2 Task 2: Casida from scratch

The Casida method boils down to solving the following eigenvalue equation [2]

$$\mathbf{\Omega F_I} = \Omega_I^2 \mathbf{F_I} \tag{1}$$

The matrix $\mathbf{\Omega}$ contains information of all transitions of the system, and is thus in theory infinitely large. In practice however one must limit the matrix to only consider transitions up to some maximum transition energy. $\Omega_I$ corresponds to the transition energy for transition $I$. The $\mathbf{\Omega}$ matrix can be computed as

$$\Omega_{pq} = \omega_p \delta_{pq} + 2\sqrt{n_p \omega_p} K_{pq} \sqrt{n_q \omega_q}. \tag{2}$$

$n_p$ and $\omega_p$ are the occupancy number difference and the Kohn-Sham eigenvalue difference for transition $p$, and $\mathbf{K}$ is the so called Hartree xc-kernel, which contains information about how the Kohn-Sham excitations, i.e. non-interacting electron transitions, couple together to yield the true transitions for the system.

Solving the eigenvalue problem in equation (1) we can calculate the oscillator strength for each transition, which corresponds to the "height" of the photoabsorption spectrum for that transition. See appendix A.2 for code implementation details. The oscillator strength $f_I^\alpha$ in the $\alpha$-direction ($\alpha$ is one of the cartesian directions) is calculated as

$$f_I^\alpha = 2\left|\sum_p \mu_p^\alpha \sqrt{n_p \omega_p} F_p^I\right|^2 \tag{3}$$

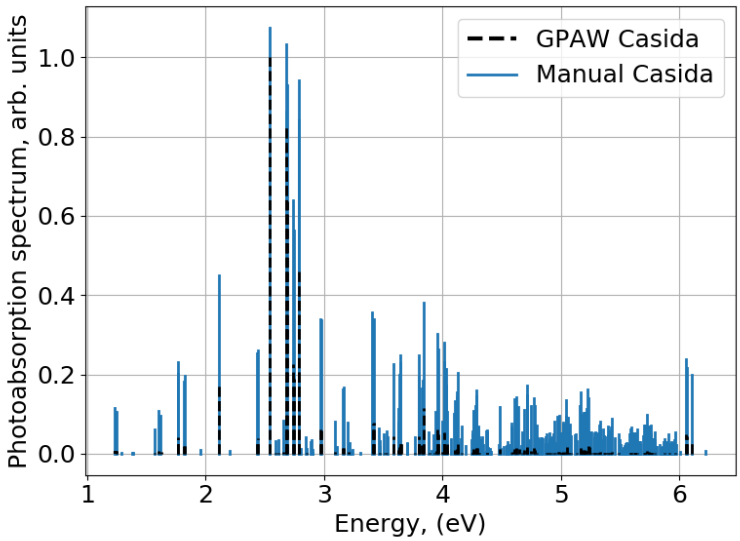


Figure 2: Discrete photoabsorption spectra, both as computed manually and with GPAW. Note the that there is a difference in the amplitude for all transitions except the most likely one between the spectra.

where $\mu_p^\alpha$ is an element of the dipole matrix for transition $I$ with regards to direction $\alpha$ [1].

If we plot the transition energies $\Omega_I$ against the corresponding oscillator strength $f_I$ we obtain the discrete photoaborption spectrum. This manually computed spectrum is compared with the discrete spectrum from GPAW from task 1 in figure 2. As we can see, the spectra are very similar regarding the position of the transitions, but the relative amplitudes between the transitions differs between the spectra. This can more easily be seen in figure 3, where each discrete peak has been folded with a gaussian of the same width as in task 1, in order to obtain a continous spectra. Again, we observe that both the manual method and GPAW yield very similar spectra in shape, but that there is a difference in the amplitude for all transitions except for the most likely one between the spectra. The differences between the GPAW spectrum and the Manual spectrum could be attributed to choosing a different grid size according to the problem description [1], however I don't see how that could explain the relative amplitude differences between the two spectra. If it had been Time-propagation TDDFT the grid spacing could have possibly affected the amount of aberrations from the Fourier transformation of the linear system response, but in Casida TDDFT I can't see how the grid spacing would come into play.

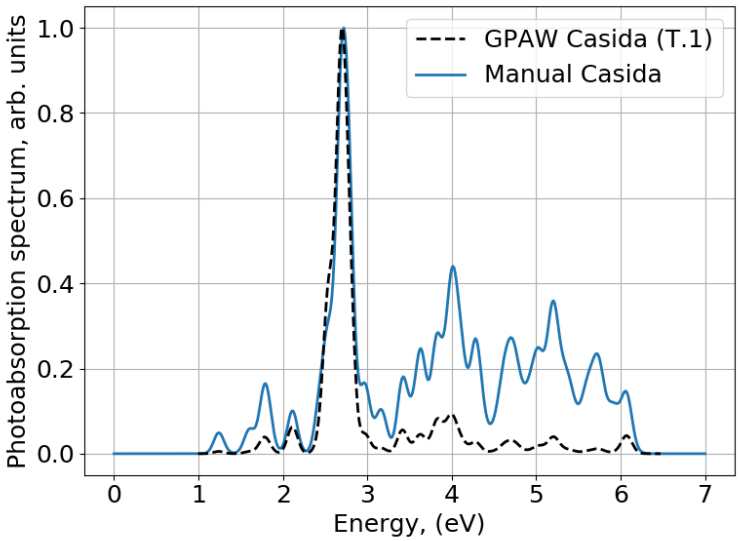


Figure 3: Folded version of the discrete photoabsorption spectra in figure 2, where the oscillator strengths have been convoluted with a gaussian.

## 3 Task 3: Kohn-Sham spectrum

If we set the xc-kernel **K** to zero in equation (2) and solve equation (1) we can obtain the Kohn-Sham spectrum, since we remove all the corrections added by TDDFT to the Kohn-Sham picture. This spectrum is given in figure 4. Comparing the spectrum with all the Kohn-Sham eigenvalue differences which are plotted as vertical lines in the same figure we see that the differences matches the transitions of the spectrum perfectly. This is expected, since the calculated spectrum with the xc-kernel set to zero is precisely the photoabsorption spectrum of the Kohn-Sham system. Hence this spectrum describes the transitions of the system had the electrons been non-interacting as well as in an effective potential, which is assumed in the Kohn-Sham picture. The Kohn-Sham picture only yields the correct system density, not necessarily the correct transitions, which is why we turn to TDDFT to correct these transitions.

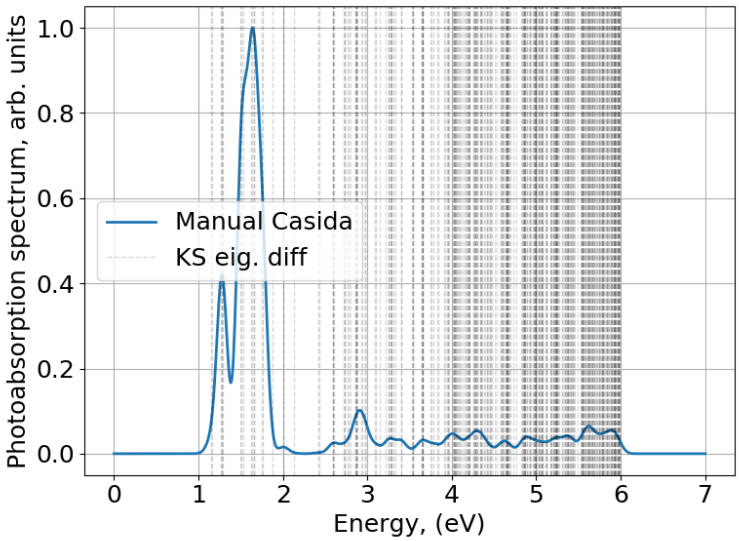


Figure 4: Kohn-Sham spectrum for $Na_8$. Notice the difference between this spectrum and the ones in figures 1 and 3; this is due to the coupling between the transitions being omitted when calculating the Kohn-Sham spectrum.

# 4 Task 4

Another approach to obtain the photoabsorption spectrum via TDDFT is by using the Time-propagation method. It is a linear response theory, in which the system is initially exposed to a weak perturbation, the response to which is studied by propagating the system in time [2]. A nice trick for extracting the photoabsorption spectrum is to subject the system to a delta-function kick; since the fourier transform of a Dirac-delta is one, this corresponds to the system being subjected to a pulse of (in this case) light containing an equal part of every frequency [2]. The response to this signal is exactly the photoabsorption spectrum, and thus one doesn't need to propagate all frequencies of interest individually.

The system was subjected to such a delta-kick and then propagated for a total of 45 fs. This propagation time determines the numerical convergence of the method; since we are interested in the frequency response of the system we must Fourier transform the response, which in theory requires the signal to be integrated over all time. To minimise aberrations we must thus propagate the system for so long so that the response has practically decayed to zero.

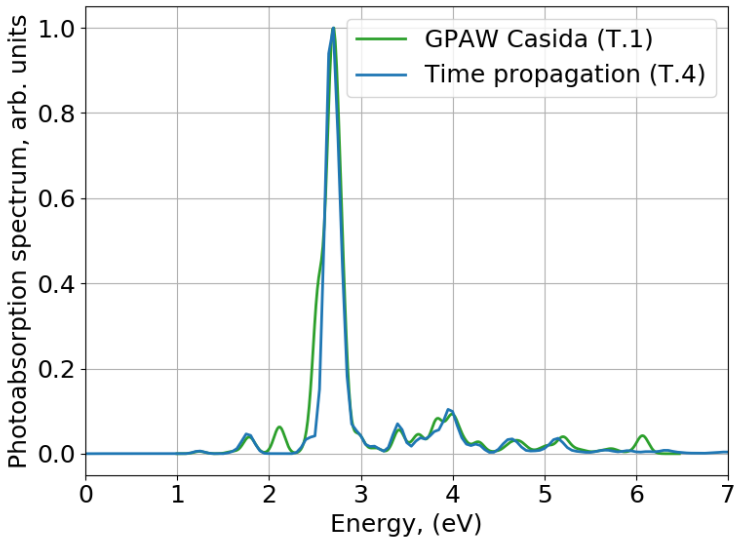


Figure 5: Photoabsorption spectra as obtained with time-propagation TDDFT and with Casida TDDFT.

The resulting photoabsorption spectrum using Time-propagation TDDFT is given in figure 5. We note that it is very similar to what was obtained in Task 1 with Casida TDDFT, but that there are some slight differences between the two spectra. This runs for the two methods not having converged to the same degree. In order for the spectra to be truly the same one would need to converge them further, i.e. propagating the system for a longer time in Time-propagation TDDFT and using a larger cut-off energy in the case of Casida TDDFT [2].

# 5 Task 5

Finally, we recompute the photoabsorption spectrum from task 1, i.e. Casida TDDFT, with a lower cutoff-energy of 4 eV instead of 6 eV. The resulting spectrum is given in figure 6. We observe that the spectra are fairly similar, disregarding aberrations above the cutoff energies, but that the peaks for the case of a lower cutoff energy seem to be somewhat lower in energy. By running the LrTDDFT analyzer we see that the coupling for a transition to a certain band is largest to transitions between lower bands. For instance, the transitions to band 36 ($E = 3.00\,\text{eV}$) couple the largest to the band $1 \rightarrow 11$ transition, with a fraction of $\sim 0.59$. But, we also see that it is coupled to other transitions, in the category `rest`, with a fraction of $\sim 0.06$. If I've interpreted the output of the analyze module correctly (I couldn't find any documentation on it, unfortunately), this rest category could include all other transitions between bands, including transitions between bands above band 36. Thus, by limiting the cutoff energy to 4 eV for our transitions we could miss out on the coupling between lower energy transitions below 4 eV and higher energy ones. The size of this error, which would be on the order of a few percent (the size of the `rest`-category) could possibly explain the small energy differences between the spectra in

figure 6.

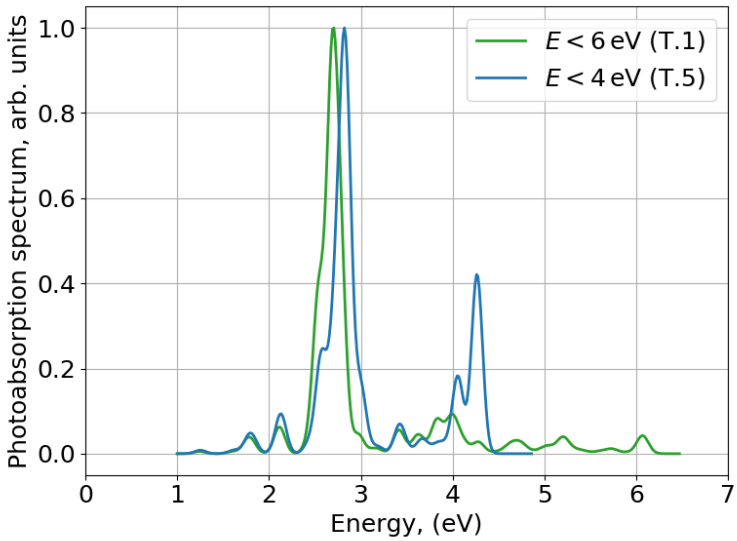


Figure 6: Casida photoabsorption spectra with two different cutoff energies. T.1 corresponds to Task 1 and T.4 to Task 5.

We now turn to the provided article. If we compare the computed photoabsorption spectrum with TDDFT in the adiabatic approximation (row **d)**) with the experimental result (row **e)**) we see a striking similarity between them. However, there are some slight differences. The authors state that the calculations have been properly converged, which rules out convergence errors such as a too low cutoff energy, grid spacing or number of bands as the source of the difference. This leaves larger systematic errors. One such error could be that the authors used the LDA xc-functional (TDLDA), which is fairly inexpensive but not always too accurate. We used the same functional in this assignment. One more advanced functional which could yield better results (atleast in general among other systems) is PBE, but to find the best functional one would have to test different functionals for the system at hand. If one wants results that are almost perfectly in line with experiments then one could use semi-empirical functionals, however that may be seen as not rigorous enough from a first-principles perspective. Another systematic error source could be that the Casida method relies on linear response theory, which is an approximation valid for weak perturbations. I assume that the experimental results in the paper also corresponds to weak electromagnetic fields, but the approximation still introduces some error. Thirdly, in TDDFT the motion and interaction of the nuclei with light are generally omitted [2], which could impact the behaviour of the electron density and thus the electronic transitions.

One example of a real-life scenario in which the photoabsorption spectrum of a simulated atomic system could be interasting could be in the design of solar panels. Specifically, solar panels which utilizes the plasmonic behaviour of nanoparticles, such as described in this article [3]. Plasmonic behaviour is when the electron densities in the system couple to an external electric field, such as light, which can lead to the system exhibiting effects such as Localized

Surface Plasmon Resonance (LSPR) [4]. In this application it would be interesting to study the photoabsorption spectrum in conjunction with the plasmonic resonance spectrum in order to find nanoparticle structures that yields both the desired plasmonic and absorption properties for the solar panel.

The system in my example is a bit larger than what has been studied in this home assignment, with nanoparticles ($\sim$ 100 atoms) instead of clusters ($\sim$ 10 atoms). However, the methods would probably be the same with the difference that it would be much more computationally expensive.

## References


[1] Nicklas Österbacka. *Home Assignment 4: TDDFT on a sodium cluster*. Accessed: 2020-02-27. 2020. URL: %5Curl%7Bhttps://chalmers.instructure.com/courses/8949/files?preview=390331%7D.

[2] Tuomas Rossi. *Lecture 9 – TDDFT*. Accessed: 2020-02-27. 2020. URL: %5Curl%7Bhttps://chalmers.instructure.com/courses/8949/files?preview=392414%7D.

[3] Emanuel Kymakis Emanuel Stratakis. *Nanoparticle-based plasmonic organic photovoltaic devices*. Accessed: 2020-02-28. 2013. URL: %5Curl%7Bhttps://www.sciencedirect.com/science/article/pii/S1369702113001065%7D.

[4] Christopher Langhammar. *Lecture 1, Plasmonic nanospectroscopy, handouts*. Accessed: 2020-02-28. 2019. URL: %5Curl%7Bhttps://chalmers.instructure.com/courses/7761/files/135923?module_item_id=28354%7D.


# A Python scripts

## A.1 Task 1

Please run both scripts in the folder ”task1”.

```
# Built-in packages
import time

# ASE
from ase import Atoms
from ase.io import read, write
from ase.parallel import world

# GPAW
from gpaw import GPAW


# Load Na8
atoms = read('../Na-tddft/Na8.xyz')
atoms.center(vacuum=8.0) # Add 8 Angstrom of vacuum around the cluster

# Define calculator
calc = GPAW(
    mode = 'fd',
```

```
    xc = 'LDA',
    setups = {'Na': '1'},
    h = 0.3,
    nbands = 0,
    txt = 'conv.gpaw-out'
)
atoms.set_calculator(calc)

#------------ Converge ground state ------------
if world.rank==0:
    print(f'------------   Convergence calculation for ground state   ------------')
start = time.time()

atoms.get_potential_energy()
# Save result for ground state
calc.write('groundCalc.gpw', 'all')

#------------ Converge empty states ------------
if world.rank==0:
    print(f'------------   Convergence calculation for empty states   ------------')

calc.set(
    nbands = 110,
    convergence = {'bands': -10},
    fixdensity = True,
    eigensolver = 'cg'
)
atoms.get_potential_energy()

if world.rank==0:
    print(f'Kohn-Sham eigenvalues: {calc.get_eigenvalues()}')

# Save results for empty states
calc.write('emptyCalc.gpw', 'all')

end = time.time()
if world.rank==0:
    print('-------- Convergence calculation finished in: ' + f'{(end-start):.2f} s ↩
        --------'.rjust(34))
    print('------------------------------------------------------------------------')
```

```
# Built-in packages
import time

# ASE
from ase.parallel import world

# GPAW
from gpaw import GPAW
from gpaw.lrtddft import LrTDDFT
from gpaw.lrtddft import photoabsorption_spectrum


if world.rank==0:
    print(f'------------   Extracting photoabsorption spectrum using TDDFT   ↩
        ------------')
start = time.time()

# Load calculator after relaxing empty structure
calc = GPAW('emptyCalc.gpw')
```

```
calc.set(
    txt = 'spec.gpaw-out'
)

# Calculate and diagonalize Omega matrix
dE = 6  # Up to 6 eV transitions considered
lr = LrTDDFT(
    calc,
    xc='LDA',
    energy_range=dE,
)  # Construct the omega matrix, parallelised over all available cores

lr.write(f'lr_dE={dE}eV.dat.gz')  # Save the tdDFT calculater just in case

lr.diagonalize()
wd = 0.06
photoabsorption_spectrum(
    lr,
    f'spectrum_w{wd}.dat',
    width = wd
)


 # Save results for task 2
lr.write ('TDDFT_Task1.dat')


end = time.time()
if world.rank==0:
    print('-------- Photoabsorption spectrum extracted in: ' + f'{(end-start):.2f} s ↩
        --------'.rjust(34))
    print('-------------------------------------------------------------------------')
```

```
# Imports
import numpy as np
import matplotlib.pyplot as plt

# Set plot params
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)     # fontsize of the x and y labels
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize



# Load data
spec_data = np.loadtxt('spectrum_w0.06.dat', skiprows=4)

# Plot
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(spec_data[:,0], spec_data[:,1], linestyle='-')
ax.set_xlabel('Energy, (eV)')
ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
ax.set_xlim(0,7)
ax.grid()
plt.tight_layout()
plt.savefig('task1_spectra.png')
plt.show()
```

## A.2 Task 2

Please run in the folder ”task2”.

```python
# External imports
import helper as h

from gpaw import GPAW
from gpaw.lrtddft import LrTDDFT

# Retrieve the calculator from task 1
lr = LrTDDFT('../task1/TDDFT_Task1.dat')

# Dump the results from the calculation to file
h.dump_data(lr, fpath='dumpTask1.npz' )

# Also generate the discrete spectrum from GPAW
h.discrete_spectrum(lr, 'GPAW_discrete.dat')
```

```python
# External imports
import numpy as np
import matplotlib.pyplot as plt
import helper as h


# Set plot params
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)     # fontsize of the x and y labels
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize

# Load dumped data from task 1
d = np.load('dumpTask1.npz')
K = d['K_pp'] # K-matrix
omega = d['ediff_p'] # KS eigenvalue difference
n = d['fdiff_p']  # Occupation difference
mux_p = d['mux_p']
muy_p = d['muy_p']
muz_p = d['muz_p']
mu_p = [mux_p, muy_p, muz_p]

# Construct the Omega matrix
Omega = np.diag(v=omega**2, k=0) # First add diagonal
for p in range(len(Omega)):
    for q in range(len(Omega)):
        Omega[p,q] += 2*np.sqrt(n[p]*omega[p]) * K[p,q] * np.sqrt(n[q]*omega[q])

# Obtain egivenalues and eigenvectors from the Omega matrix
eigVal, F = np.linalg.eig(Omega)  # Eigenvalue, eigenvectors
sort = np.argsort(eigVal)
eigVal = eigVal[sort]
F = F[:,sort]
```

```
36
37 # Extract the excitations in Ha, and convert them to eV
38 eps = np.sqrt(eigVal) * 27.2
39
40 # Calculate the oscillator strength
41 f = np.zeros(len(eigVal))
42 for i, e in enumerate(eigVal):
43     for alpha, mua_p in enumerate(mu_p):
44         f_ia = 2 * np.abs( np.sum( mua_p * np.sqrt(n*omega) * F[:,i] ) )
45         f[i] += f_ia
46     f[i] /= 3 # Average over all dipole moments
47
48 # Compare with the discrete spectrum from GPAW
49 # Plot
50 disc_data = np.loadtxt('GPAW_discrete.dat', skiprows=0)
51 fig, ax = plt.subplots(figsize=(8,6))
52 for i,e in enumerate(eps):
53     if i==0:
54         # GPAW
55         d_e = disc_data[i,1]
56         I_e = disc_data[i,2]
57         ax.plot([d_e, d_e], [0, I_e] / max(disc_data[:,2]), color='k', linestyle='--'↩
               , linewidth='3', zorder=3, label='GPAW Casida')
58         # Manual
59         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2', label='↩
               Manual Casida')
60     else:
61         # GPAW
62         d_e = disc_data[i,1]
63         I_e = disc_data[i,2]
64         ax.plot([d_e, d_e], [0, I_e] / max(disc_data[:,2]), color='k', linestyle='--'↩
               , zorder=3, linewidth='2')
65         # Manual
66         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2')
67 ax.set_xlabel('Energy, (eV)')
68 ax.set_ylabel(r'Oscillator strength, arb. units')
69 ax.grid()
70 plt.tight_layout()
71 ax.legend(loc='best')
72 plt.savefig('task2_discrete_spectra.png')
73
74 # Finally, convolute my spectrum with a Gaussian and compare with task 1
75 task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)
76
77 energy = np.linspace(0, 7, 500)
78 f_fold = h.fold(x_t=energy, x_i=eps, y_i=f, width=0.06)
79 fig, ax = plt.subplots(figsize=(8,6))
80 ax.plot(task1[:,0], task1[:,1] / max(task1[:,1]), color='k', linestyle='--', ↩
    linewidth='2', zorder=3, label='GPAW Casida (T.1)') # Normalize both spectra ↩
    since arbitrary units
81 ax.plot(energy, f_fold / max(f_fold), color='C0', linestyle='-', linewidth='2', label↩
    ='Manual Casida')
82 ax.set_xlabel('Energy, (eV)')
83 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
84 ax.grid()
85 plt.tight_layout()
86 ax.legend(loc='best')
87 plt.savefig('task2_folded_spectra.png')
88 plt.show()
```

## A.3 Task 3

Please run in the folder ”task3”.

```
# External imports
import numpy as np
import matplotlib.pyplot as plt
import helper as h


# Set plot params
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)     # fontsize of the x and y labels
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize

# Load dumped data from task 1
d = np.load('../task2/dumpTask1.npz')
K = d['K_pp'] # K-matrix
omega = d['ediff_p'] # KS eigenvalue difference
n = d['fdiff_p']  # Occupation difference
mux_p = d['mux_p']
muy_p = d['muy_p']
muz_p = d['muz_p']
mu_p = [mux_p, muy_p, muz_p]

# Construct the Omega matrix
Omega = np.diag(v=omega**2, k=0) # First add diagonal
# Nothing more, since K=0

# Obtain egivenalues and eigenvectors from the Omega matrix
eigVal, F = np.linalg.eig(Omega)  # Eigenvalue, eigenvectors
sort = np.argsort(eigVal)
eigVal = eigVal[sort]
F = F[:,sort]

# Extract the excitations in Ha, and convert them to eV
eps = np.sqrt(eigVal) * 27.2

# Calculate the oscillator strength
f = np.zeros(len(eigVal))
for i, e in enumerate(eigVal):
    for alpha, mua_p in enumerate(mu_p):
        f_ia = 2 * np.abs( np.sum( mua_p * np.sqrt(n*omega) * F[:,i] ) )
        f[i] += f_ia
    f[i] /= 3 # Average over all dipole moments

# Compare with the discrete spectrum from GPAW
# Plot
# fig, ax = plt.subplots(figsize=(8,6))
# for i,e in enumerate(eps):
#     if i==0:
#         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2', label='↩
    Casida (manual)')
#     else:
#         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2')
# ax.set_xlabel('Energy, (eV)')
# ax.set_ylabel(r'Oscillator strength, arb. units')
```

```
# ax.grid()
# plt.tight_layout()
# plt.savefig('task3_discrete_kohn_sham_spectra.png')


# Finally, convolute my spectrum with a Gaussian and compare with KS eigenvalue differences
print('**** Kohn-Sham eigenvalue differnces ****')
print(omega)
print('****************************************')
# TODO: What do these KS transitions mean?
energy = np.linspace(0, 7, 1000)
f_fold = h.fold(x_t=energy, x_i=eps, y_i=f, width=0.06)
fig, ax = plt.subplots(figsize=(8,6))

ax.plot(energy, f_fold / max(f_fold), color='C0', linestyle='-', linewidth='2', label='Manual Casida')
ax.set_xlabel('Energy, (eV)')
ax.set_ylabel(r'Oscillator strength, arb. units')
ax.grid()
plt.tight_layout()
plt.savefig('task3_kohn_sham_spectra.png')
plt.show()
```

## A.4 Task 4

Please run in the folder ”task4”.

```
# Built-in packages
import time

# ASE
from ase import Atoms
from ase.io import read, write
from ase.parallel import world

# GPAW
from gpaw import GPAW
from gpaw.tddft import *

# Load the ground state calculator from task1
td_calc = TDDFT('../task1/groundCalc.gpw')

# Set up time propagation DFT
time_step = 30  # 30 attoseconds
total_time = 45000
iterations = total_time/time_step
kick_strength = 1e-5  # Kick with a field of this strength in x, y and z directions

# Run calculation in each cartesian direction
coord = ['x', 'y', 'z']

if world.rank==0:
    print(f'------------   Task 4 TDDFT started   ------------')
start = time.time()

for i, c in enumerate(coord):
    if world.rank==0:
```

```
        print(f'------------   Propagating {c}-direction   ------------')
    start_c = time.time()
    # Prepare kick
    kick = [0.0]*3
    kick[i] = kick_strength
    # Kick with a delta kick in direction c
    td_calc.absorption_kick(kick_strength=kick)
    # Propagate system and save time dependent dipole moment to 'Na8_dm_{c}.dat' and
    # use 'Na8_td_{c}.dat' as restart file
    td_calc.propagate(time_step, iterations, f'Na8_dm_{c}.dat', f'Na8_td_{c}.dat')

    # Calculate photoabsorption spectrum and save it
    wd = 0.06
    photoabsorption_spectrum(f'Na8_dm_{c}.dat', f'Na8_spectrum_{c}.dat', width = wd)
    end = time.time()

    if world.rank==0:
        print(f'-------- Propagating {c}-direction finished in: ' + f'{(end-start):.2f} s --------'.rjust(34))


end = time.time()
if world.rank==0:
    print('-------- TDDFT calculation finished in: ' + f'{(end-start):.2f} s --------'.rjust(34))
    print('------------------------------------------------------------------------')
# Load all spectra, take the average and plot in a separate script.
```

```
 # Imports
import numpy as np
import matplotlib.pyplot as plt

# Set plot params
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize



# Load data
spec_x = np.loadtxt('Na8_spectrum_x.dat', skiprows=6)
spec_y = np.loadtxt('Na8_spectrum_y.dat', skiprows=6)
spec_z = np.loadtxt('Na8_spectrum_z.dat', skiprows=6)
energies = spec_x[:,0]
spec = np.sqrt(spec_x[:,1]**2 + spec_y[:,1]**2 + spec_z[:,1]**2)

# Plot
task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)

fig, ax = plt.subplots(figsize=(8,6))
ax.plot(task1[:,0], task1[:,1] / max(task1[:,1]), color='C2', linestyle='-', linewidth='2', label='GPAW Casida (T.1)') # Normalize both spectra since arbitrary units
ax.plot(energies, spec / max(spec),  color='C0', linestyle='-', linewidth='2', label='Time propagation (T.4)')
ax.set_xlabel('Energy, (eV)')
ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
```

```
ax.set_xlim(0,7)
ax.grid()
ax.legend(loc='best')
plt.tight_layout()
plt.savefig('task4_spectra.png')
plt.show()
```

## A.5 Task 5

Please run in the folder ”task5”.

```
# Built-in packages
import time

# GPAW
from gpaw import GPAW
from gpaw.lrtddft import LrTDDFT
from gpaw.lrtddft import photoabsorption_spectrum


print(f'------------   Extracting shortened photoabsorption spectrum   ------------')

start = time.time()

# Import LrTDDFT results from Task 1
lr = LrTDDFT('../task1/TDDFT_Task1.dat')

lr.diagonalize(energy_range=4) # Only include up to 4 eV

# Generate spectrum and save it
wd = 0.06
photoabsorption_spectrum(
    lr,
    f'spectrum_w{wd}.dat',
    width = wd
)

# Extract all information about all transitions
print('** LrTDDFT.analyse() output **')
lr.analyse()
print('********************************')


end = time.time()
print('-------- Photoabsorption spectrum extracted in: ' + f'{(end-start):.2f} s ↩
    --------'.rjust(34))
print('-------------------------------------------------------------------------')
```

```
 # Imports
import numpy as np
import matplotlib.pyplot as plt

# Set plot params
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)    # fontsize of the x and y labels
```

```
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize




# Plot
task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)
task5 = np.loadtxt('../task5/spectrum_w0.06.dat', skiprows=4)

fig, ax = plt.subplots(figsize=(8,6))
ax.plot(task1[:,0], task1[:,1] / max(task1[:,1]), color='C2', linestyle='-', ↩
    linewidth='2', label=r'$E<6 \rm\, eV$ (T.1)') # Normalize both spectra since ↩
    arbitrary units
ax.plot(task5[:,0], task5[:,1] / max(task5[:,1]),  color='C0', linestyle='-', ↩
    linewidth='2', label=r'$E<4 \rm\, eV$ (T.5)')
ax.set_xlabel('Energy, (eV)')
ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
ax.set_xlim(0,7)
ax.grid()
ax.legend(loc='best')
plt.tight_layout()
plt.savefig('task5_spectra.png')
plt.show()
```