

# Assignment 4 – Computational Materials and Molecular Physics

## First hand-in

Eric Lindgren – CID: ericlin

March 10, 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  $\text{Na}_8$  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  $\text{Na}_8$  cluster seems to have its 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 unoccupied 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 interesting 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).

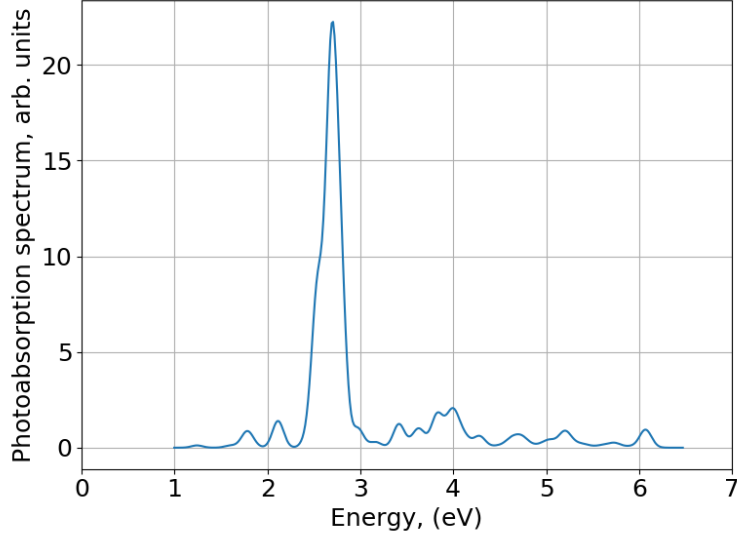


Figure 1: Photoabsorption spectrum as obtained with the Casida method in GPAW. Note that the most probable transition for  $\text{Na}_8$  seems to lie at around 3 eV.

## 2 Task 2: Casida from scratch

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

$$\mathbf{\Omega} \mathbf{F}_I = \Omega_I^2 \mathbf{F}_I \quad (1)$$

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

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

$n_p$  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

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

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

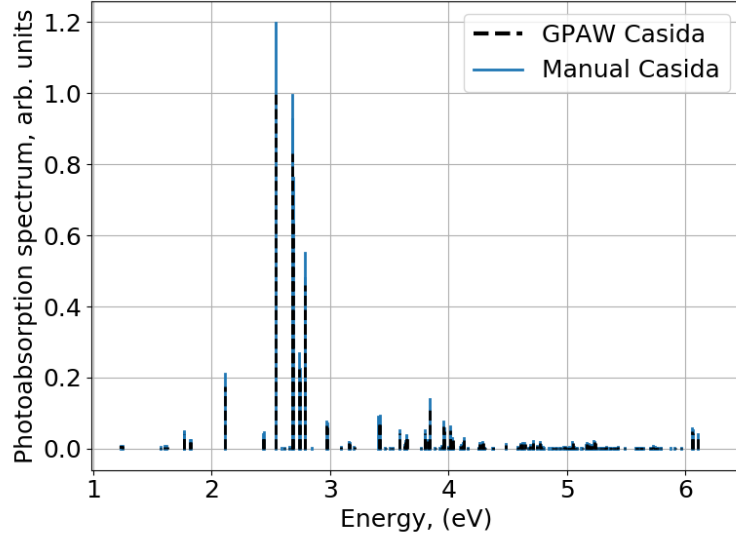


Figure 2: Discrete photoabsorption spectra, both as computed manually and with GPAW. Note that the spectra are very similar, barring small amplitude specific differences..

If we plot the transition energies  $\Omega_I$  against the corresponding oscillator strength  $f_I$  we obtain the discrete photoabsorption 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 some of the discrete amplitudes varies slightly. This difference is not noticeable though 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 continuous spectra. Again, we observe that both the manual method and GPAW yield very similar spectra in shape. The small differences between the spectra could be attributed to the difference in the grid spacing between GPAW and my manual implementation.

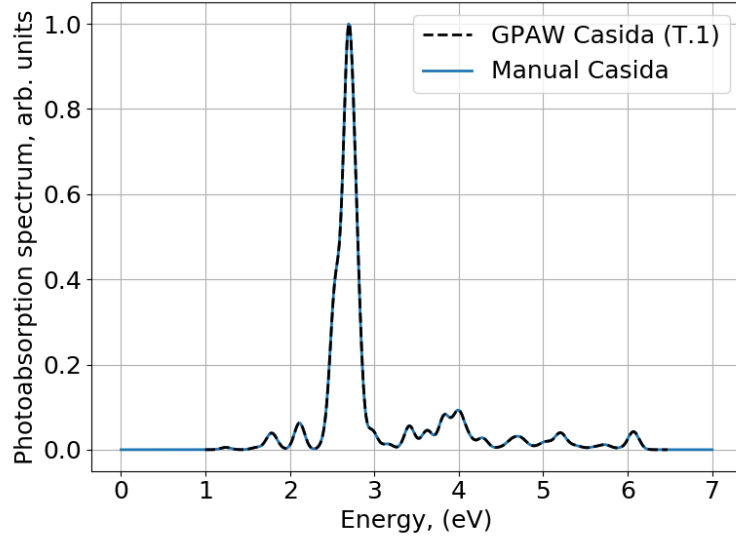


Figure 3: Folded version of the discrete photoabsorption spectra in figure 2, where the oscillator strengths have been convoluted with a gaussian.

### 3 Task 3: Kohn-Sham spectrum

If we set the  $\mathbf{K}$ -matrix, which includes all transition coupling, 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  $\mathbf{K}$ -matrix 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.

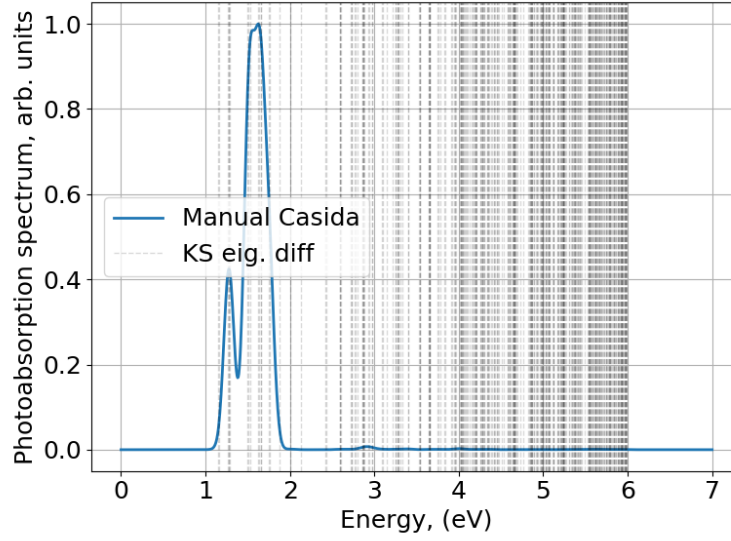


Figure 4: Kohn-Sham spectrum for  $\text{Na}_8$ . Notice the difference between this spectrum and the ones in figures 1 and 3; this is due to the coupling between the transitions being omitted when calculating the Kohn-Sham spectrum.

## 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.

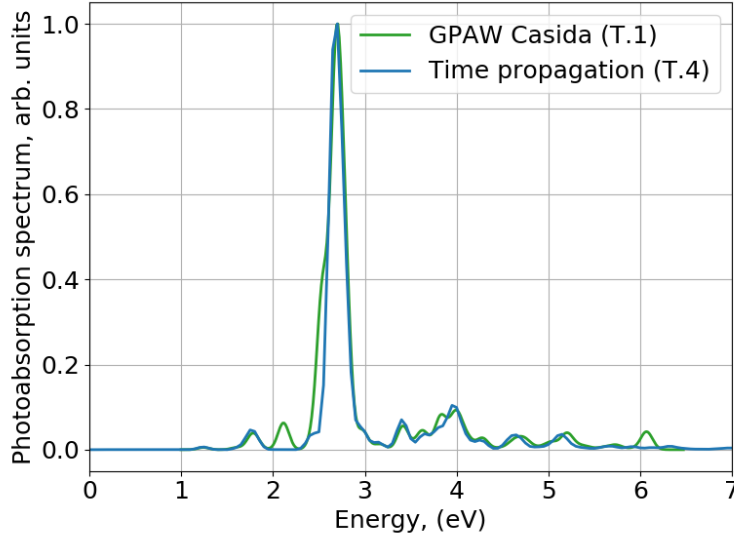


Figure 5: Photoabsorption spectra as obtained with time-propagation TDDFT and with Casida TDDFT.

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. either propagating the system for a longer time or using a shorter timestep 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 ( $E = 3.00$  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.

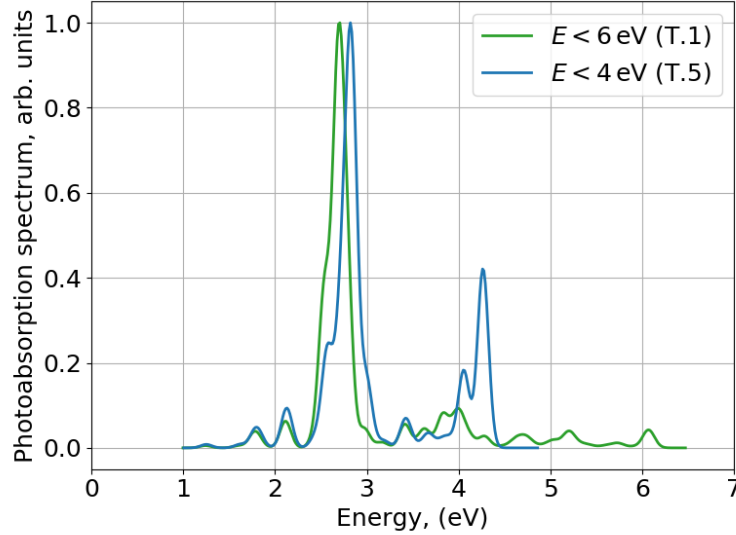


Figure 6: Casida photoabsorption spectra with two different cutoff energies. T.1 corresponds to Task 1 and T.4 to Task 5.

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 interesting 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".

```
1 # Built-in packages
2 import time
3
4 # ASE
5 from ase import Atoms
6 from ase.io import read, write
7 from ase.parallel import world
8
9 # GPAW
10 from gpaw import GPAW
11
12
13 # Load Na8
14 atoms = read('./Na-tddft/Na8.xyz')
15 atoms.center(vacuum=8.0) # Add 8 Angstrom of vacuum around the cluster
16
17 # Define calculator
18 calc = GPAW(
```



```

19     mode = 'fd',
20     xc = 'LDA',
21     setups = {'Na': '1'},
22     h = 0.3,
23     nbands = 0,
24     txt = 'conv.gpaw-out'
25 )
26 atoms.set_calculator(calc)
27
28 #----- Converge ground state -----
29 if world.rank==0:
30     print(f'----- Converge calculation for ground state -----')
31 start = time.time()
32
33 atoms.get_potential_energy()
34 # Save result for ground state
35 calc.write('groundCalc.gpw', 'all')
36
37 #----- Converge empty states -----
38 if world.rank==0:
39     print(f'----- Converge calculation for empty states -----')
40
41 calc.set(
42     nbands = 110,
43     convergence = {'bands': -10},
44     fixdensity = True,
45     eigensolver = 'cg'
46 )
47 atoms.get_potential_energy()
48
49 if world.rank==0:
50     print(f'Kohn-Sham eigenvalues: {calc.get_eigenvalues()}')
51
52 # Save results for empty states
53 calc.write('emptyCalc.gpw', 'all')
54
55 end = time.time()
56 if world.rank==0:
57     print(f'----- Converge calculation finished in: ' + f'{{(end-start):.2f}} s ←
58         -----'.rjust(34))
59     print('-----')

```

```

1 # Built-in packages
2 import time
3
4 # ASE
5 from ase.parallel import world
6
7 # GPAW
8 from gpaw import GPAW
9 from gpaw.lrtddft import LrTDDFT
10 from gpaw.lrtddft import photoabsorption_spectrum
11
12
13 if world.rank==0:
14     print(f'----- Extracting photoabsorption spectrum using TDDFT ←
15         -----')
16 start = time.time()
17
18 # Load calculator after relaxing empty structure

```

```

18 calc = GPAW('emptyCalc.gpw')
19 calc.set(
20     txt = 'spec.gpaw-out'
21 )
22
23 # Calculate and diagonalize Omega matrix
24 dE = 6 # Up to 6 eV transitions considered
25 lr = LrTDDFT(
26     calc,
27     xc='LDA',
28     energy_range=dE,
29 ) # Construct the omega matrix, parallelised over all available cores
30
31 lr.write(f'lr_dE={dE}eV.dat.gz') # Save the tdDFT calculator just in case
32
33 lr.diagonalize()
34 wd = 0.06
35 photoabsorption_spectrum(
36     lr,
37     f'spectrum_w{wd}.dat',
38     width = wd
39 )
40
41 # Save results for task 2
42 lr.write('TDDFT_Task1.dat')
43
44
45
46 end = time.time()
47 if world.rank==0:
48     print('----- Photoabsorption spectrum extracted in: ' + f'{{(end-start):.2f}} s ←
49           -----'.rjust(34))
49     print('-----')

```

```

1 # Imports
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Set plot params
6 plt.rc('font', size=18) # controls default text sizes
7 plt.rc('axes', titlesize=18) # fontsize of the axes title
8 plt.rc('axes', labelszize=18) # fontsize of the x and y labels
9 plt.rc('xtick', labelszize=18) # fontsize of the tick labels
10 plt.rc('ytick', labelszize=18) # fontsize of the tick labels
11 plt.rc('legend', fontsize=18) # legend fontsize
12
13
14
15 # Load data
16 spec_data = np.loadtxt('spectrum_w0.06.dat', skiprows=4)
17
18 # Plot
19 fig, ax = plt.subplots(figsize=(8,6))
20 ax.plot(spec_data[:,0], spec_data[:,1], linestyle='-')
21 ax.set_xlabel('Energy, (eV)')
22 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
23 ax.set_xlim(0,7)
24 ax.grid()
25 plt.tight_layout()
26 plt.savefig('task1_spectra.png')

```

```
27 plt.show()
```

## A.2 Task 2

Please run in the folder "task2".

```
1 # External imports
2 import helper as h
3
4 from gpaw import GPAW
5 from gpaw.lrtddft import LrTDDFT
6
7 # Retrieve the calculator from task 1
8 lr = LrTDDFT('../task1/TDDFT_Task1.dat')
9
10 # Dump the results from the calculation to file
11 h.dump_data(lr, fpath='dumpTask1.npz')
12
13 # Also generate the discrete spectrum from GPAW
14 h.discrete_spectrum(lr, 'GPAW_discrete.dat')
```

```
1 # External imports
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import helper as h
5
6
7 # Set plot params
8 plt.rc('font', size=18)           # controls default text sizes
9 plt.rc('axes', titlesize=18)      # fontsize of the axes title
10 plt.rc('axes', labelsiz=18)       # fontsize of the x and y labels
11 plt.rc('xtick', labelsiz=18)      # fontsize of the tick labels
12 plt.rc('ytick', labelsiz=18)      # fontsize of the tick labels
13 plt.rc('legend', fontsize=18)     # legend fontsize
14
15 # Load dumped data from task 1
16 d = np.load('dumpTask1.npz')
17 K = d['K_pp'] # K-matrix
18 omega = d['ediff_p'] # KS eigenvalue difference
19 n = d['fdiff_p'] # Occupation difference
20 mux_p = d['mux_p']
21 muy_p = d['muy_p']
22 muz_p = d['muz_p']
23 mu_p = [mux_p, muy_p, muz_p]
24
25 # Construct the Omega matrix
26 Omega = np.diag(v=omega**2, k=0) # First add diagonal
27 for p in range(len(Omega)):
28     for q in range(len(Omega)):
29         Omega[p,q] += 2*np.sqrt(n[p]*omega[p]) * K[p,q] * np.sqrt(n[q]*omega[q])
30
31 # Obtain eigenvalues and eigenvectors from the Omega matrix
32 eigVal, F = np.linalg.eig(Omega) # Eigenvalue, eigenvectors
33 sort = np.argsort(eigVal)
34 eigVal = eigVal[sort]
35 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] ) )**2
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='--', ←
58             , linewidth='3', zorder=3, label='GPAW Casida')
59         # Manual
60         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2', label='←
61             Manual Casida')
62     else:
63         # GPAW
64         d_e = disc_data[i,1]
65         I_e = disc_data[i,2]
66         ax.plot([d_e, d_e], [0, I_e] / max(disc_data[:,2]), color='k', linestyle='--', ←
67             , zorder=3, linewidth='2')
68         # Manual
69         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2')
70 ax.set_xlabel('Energy, (eV)')
71 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
72 ax.grid()
73 plt.tight_layout()
74 ax.legend(loc='best')
75 plt.savefig('task2_discrete_spectra.png')
76
77 # Finally, convolute my spectrum with a Gaussian and compare with task 1
78 task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)
79
80 energy = np.linspace(0, 7, 500)
81 f_fold = h.fold(x_t=energy, x_i=eps, y_i=f, width=0.06)
82 fig, ax = plt.subplots(figsize=(8,6))
83 ax.plot(task1[:,0], task1[:,1] / max(task1[:,1]), color='k', linestyle='--', ←
84     linewidth='2', zorder=3, label='GPAW Casida (T.1)') # Normalize both spectra ←
85     since arbitrary units
86 ax.plot(energy, f_fold / max(f_fold), color='C0', linestyle='-', linewidth='2', label←
87     ='Manual Casida')
88 ax.set_xlabel('Energy, (eV)')
89 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
90 ax.grid()
91 plt.tight_layout()
92 ax.legend(loc='best')
93 plt.savefig('task2_folded_spectra.png')
94 plt.show()

```

### A.3 Task 3

Please run in the folder "task3".

```
1  # External imports
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import helper as h
5
6
7  # Set plot params
8  plt.rc('font', size=18)           # controls default text sizes
9  plt.rc('axes', titlesize=18)      # fontsize of the axes title
10 plt.rc('axes', labelsiz=18)       # fontsize of the x and y labels
11 plt.rc('xtick', labelsiz=18)      # fontsize of the tick labels
12 plt.rc('ytick', labelsiz=18)      # fontsize of the tick labels
13 plt.rc('legend', fontsize=18)     # legend fontsize
14
15 # Load dumped data from task 1
16 d = np.load('../task2/dumpTask1.npz')
17 K = d['K_pp'] # K-matrix
18 omega = d['ediff_p'] # KS eigenvalue difference
19 n = d['fdiff_p'] # Occupation difference
20 mux_p = d['mux_p']
21 muy_p = d['muy_p']
22 muz_p = d['muz_p']
23 mu_p = [mux_p, muy_p, muz_p]
24
25 # Construct the Omega matrix
26 Omega = np.diag(v=omega**2, k=0) # First add diagonal
27 # Nothing more, since K=0
28
29 # Obtain eigenvalues and eigenvectors from the Omega matrix
30 eigVal, F = np.linalg.eig(Omega) # Eigenvalue, eigenvectors
31 sort = np.argsort(eigVal)
32 eigVal = eigVal[sort]
33 F = F[:,sort]
34
35 # Extract the excitations in Ha, and convert them to eV
36 eps = np.sqrt(eigVal) * 27.2
37
38 # Calculate the oscillator strength
39 f = np.zeros(len(eigVal))
40 for i, e in enumerate(eigVal):
41     for alpha, mua_p in enumerate(mu_p):
42         f_ia = 2 * np.abs( np.sum( mua_p * np.sqrt(n*omega) * F[:,i] ) )**2
43         f[i] += f_ia
44     f[i] /= 3 # Average over all dipole moments
45
46 # Compare with the discrete spectrum from GPAW
47 # Plot
48 # fig, ax = plt.subplots(figsize=(8,6))
49 # for i,e in enumerate(eps):
50 #     if i==0:
51 #         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2', label='↔
52 #             Casida (manual)')
53 #     else:
54 #         ax.plot([e,e], [0,f[i]], color='C0', linestyle='-', linewidth='2')
55 # ax.set_xlabel('Energy, (eV)')
56 # ax.set_ylabel(r'Oscillator strength, arb. units')
```

```

56 # ax.grid()
57 # plt.tight_layout()
58 # plt.savefig('task3_discrete_kohn_sham_spectra.png')
59
60
61 # Finally, convolute my spectrum with a Gaussian and compare with KS eigenvalue ←
    differences
62 # print('**** Kohn-Sham eigenvalue differnces ****')
63 KS_diff = np.sort(omega*27.2)
64 # print('*****')
65 # TODO: What do these KS transitions mean?
66 energy = np.linspace(0, 7, 1000)
67 f_fold = h.fold(x_t=energy, x_i=eps, y_i=f, width=0.06)
68 fig, ax = plt.subplots(figsize=(8,6))
69 ax.plot(energy, f_fold / max(f_fold), color='C0', linestyle='-', linewidth='2', label←
    ='Manual Casida')
70 for i,k in enumerate(KS_diff):
71     if i==0:
72         # GPAW
73         ax.axvline(k, color='k', linestyle='--', zorder=3, linewidth='1', alpha=0.15,←
            label='KS eig. diff')
74     else:
75         # GPAW
76         ax.axvline(k, color='k', linestyle='--', zorder=3, linewidth='1', alpha=0.15)
77 ax.set_xlabel('Energy, (eV)')
78 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
79 ax.grid()
80 ax.legend(loc='best')
81 plt.tight_layout()
82 plt.savefig('task3_kohn_sham_spectra.png')
83 plt.show()

```

## A.4 Task 4

Please run in the folder "task4".

```

1 # Built-in packages
2 import time
3
4 # ASE
5 from ase import Atoms
6 from ase.io import read, write
7 from ase.parallel import world
8
9 # GPAW
10 from gpaw import GPAW
11 from gpaw.tddft import *
12
13 # Load the ground state calculator from task1
14 td_calc = TDDFT('../task1/groundCalc.gpw')
15
16 # Set up time propagation DFT
17 time_step = 30 # 30 attoseconds
18 total_time = 45000
19 iterations = total_time/time_step
20 kick_strength = 1e-5 # Kick with a field of this strength in x, y and z directions
21
22 # Run calculation in each cartesian direction

```

```

23 coord = ['x', 'y', 'z']
24
25 if world.rank==0:
26     print(f'----- Task 4 TDDFT started -----')
27 start = time.time()
28
29 for i, c in enumerate(coord):
30     if world.rank==0:
31         print(f'----- Propagating {c}-direction -----')
32         start_c = time.time()
33         # Prepare kick
34         kick = [0.0]*3
35         kick[i] = kick_strength
36         # Kick with a delta kick in direction c
37         td_calc.absorption_kick(kick_strength=kick)
38         # Propagate system and save time dependent dipole moment to 'Na8_dm_{c}.dat' and
39         # use 'Na8_td_{c}.dat' as restart file
40         td_calc.propagate(time_step, iterations, f'Na8_dm_{c}.dat', f'Na8_td_{c}.dat')
41
42         # Calculate photoabsorption spectrum and save it
43         wd = 0.06
44         photoabsorption_spectrum(f'Na8_dm_{c}.dat', f'Na8_spectrum_{c}.dat', width = wd)
45         end = time.time()
46
47         if world.rank==0:
48             print(f'----- Propagating {c}-direction finished in: ' + f'{{(end-start):.2f}} s -----'
49                   f' s -----'.rjust(34))
50
51 end = time.time()
52 if world.rank==0:
53     print('----- TDDFT calculation finished in: ' + f'{{(end-start):.2f}} s -----'
54           '.rjust(34))
55     print('-----')
56 # Load all spectra, take the average and plot in a separate script.

```

```

1  # Imports
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # Set plot params
6  plt.rc('font', size=18)           # controls default text sizes
7  plt.rc('axes', titlesize=18)      # fontsize of the axes title
8  plt.rc('axes', labelsize=18)     # fontsize of the x and y labels
9  plt.rc('xtick', labelsize=18)     # fontsize of the tick labels
10 plt.rc('ytick', labelsize=18)     # fontsize of the tick labels
11 plt.rc('legend', fontsize=18)     # legend fontsize
12
13
14
15 # Load data
16 spec_x = np.loadtxt('Na8_spectrum_x.dat', skiprows=6)
17 spec_y = np.loadtxt('Na8_spectrum_y.dat', skiprows=6)
18 spec_z = np.loadtxt('Na8_spectrum_z.dat', skiprows=6)
19 energies = spec_x[:,0]
20 spec = np.sqrt(spec_x[:,1]**2 + spec_y[:,1]**2 + spec_z[:,1]**2)
21
22 # Plot
23 task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)
24

```

```

25 fig, ax = plt.subplots(figsize=(8,6))
26 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
27 ax.plot(energies, spec / max(spec), color='C0', linestyle='-', linewidth='2', label=↵
    'Time propagation (T.4)')
28 ax.set_xlabel('Energy, (eV)')
29 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
30 ax.set_xlim(0,7)
31 ax.grid()
32 ax.legend(loc='best')
33 plt.tight_layout()
34 plt.savefig('task4_spectra.png')
35 plt.show()

```

## A.5 Task 5

Please run in the folder "task5".

```

1 # Built-in packages
2 import time
3
4 # GPAW
5 from gpaw import GPAW
6 from gpaw.lrtddft import LrTDDFT
7 from gpaw.lrtddft import photoabsorption_spectrum
8
9
10 print(f'----- Extracting shortened photoabsorption spectrum -----')
11
12 start = time.time()
13
14 # Import LrTDDFT results from Task 1
15 lr = LrTDDFT('../task1/TDDFT_Task1.dat')
16
17 lr.diagonalize(energy_range=4) # Only include up to 4 eV
18
19 # Generate spectrum and save it
20 wd = 0.06
21 photoabsorption_spectrum(
22     lr,
23     f'spectrum_w{wd}.dat',
24     width = wd
25 )
26
27 # Extract all information about all transitions
28 print('** LrTDDFT.analyse() output **')
29 lr.analyse()
30 print('*****')
31
32
33 end = time.time()
34 print('----- Photoabsorption spectrum extracted in: ' + f'{(end-start):.2f} s ↵
    -----'.rjust(34))
35 print('-----')

```



```

1  # Imports
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # Set plot params
6  plt.rc('font', size=18)           # controls default text sizes
7  plt.rc('axes', titlesize=18)      # fontsize of the axes title
8  plt.rc('axes', labelsiz=18)       # fontsize of the x and y labels
9  plt.rc('xtick', labelsiz=18)      # fontsize of the tick labels
10 plt.rc('ytick', labelsiz=18)      # fontsize of the tick labels
11 plt.rc('legend', fontsize=18)     # legend fontsize
12
13
14
15
16 # Plot
17 task1 = np.loadtxt('../task1/spectrum_w0.06.dat', skiprows=4)
18 task5 = np.loadtxt('../task5/spectrum_w0.06.dat', skiprows=4)
19
20 fig, ax = plt.subplots(figsize=(8,6))
21 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
22 ax.plot(task5[:,0], task5[:,1] / max(task5[:,1]), color='C0', linestyle='-', ←
    linewidth='2', label=r'$E<4 \rm\, eV$ (T.5)')
23 ax.set_xlabel('Energy, (eV)')
24 ax.set_ylabel(r'Photoabsorption spectrum, arb. units')
25 ax.set_xlim(0,7)
26 ax.grid()
27 ax.legend(loc='best')
28 plt.tight_layout()
29 plt.savefig('task5_spectra.png')
30 plt.show()

```