FYS-4096 Computational Physics, exercise 6

Return your solution to project `exercise6` under your GitLab group for this course by Friday 5 AM.

Tag the final version with `final` keyword, and make sure to include a file `problems_solved` in the repository. The `problems_solved`-file should be a comma separated list of problems you have solved.

Exercise: Time-dependent quantum mechanics

1. Introduction (1 XP)

In exercise 5, problem 1 you calculated the ground state of our 1D Hydrogen model. Now we'll have a look on how the electron density evolves when exposed to a laser pulse.

Your simulation code will be separated into two parts

1. the supporting modules and methods (place these inside `quantum_dynamics`-module/directory)
2. the main simulation script/executable [write this to `scripts/qdyn_laser` (no `.py` suffix, we'll make this an executable!)]

Begin `scripts/qdyn_laser` with the following code to be able to adjust simulation parameters from the command line.

```
#!/usr/bin/env python3

import argparse

parser = argparse.ArgumentParser(description="""Simulation of TDSE for 1D
                                 Hydrogen electron under laser electric
                                 field in the dipole approximation""")

parser.add_argument('--delta-t', '-dt', type=float, default=5e-1,
                    help="Time-step")
parser.add_argument('--max-t', '-T', type=float, default=1000,
                    help="Maximum simulated time")
parser.add_argument('--grid-length', '-L', type=float, default=500,
                    help="Length of coordinate grid: [-L/2,L/2]")
parser.add_argument('--grid-pts', '-N', type=int, default=2000,
                    help="Number of gridpoints")
parser.add_argument('--save-interval', type=int, default=1,
                    help="How many time-steps to save. 1 = each one, 2= every 2nd, 3 = every
parser.add_argument('--expm-tolerance', type=float, default=1e-5,
```

```
                    help="Relative error for the operation of the matrix exponential")
parser.add_argument('--krylov-subspace-maxdim', type=int, default=50,
                    help="Maximum dimension of the Krylov Subspace to use \
                    (warns if propagator not converged within this)")
parser.add_argument('--pulse-amplitude', type=float, default=0.08,
                    help="Amplitude of the laser electric field in atomic units")
parser.add_argument('--pulse-frequency', type=float, default=0.0569,
                    help="Carrier frequency of the laser electric field")
parser.add_argument('--pulse-duration', type=float, default=1000,
                    help="Duration of the laser electric field (sin^2\
                    envelope)")
parser.add_argument('--savefile', type=str, default="qdyn.h5",
                    help="Path where to save the simulation data")
parser.add_argument('--cap-width', type=float, default=10,
                    help="Width of the complex absorbing potential")
parser.add_argument('--cap-height', type=float, default=0.5,
                    help="Height of the complex absorbing potential")

args = parser.parse_args()

delta_t = args.delta_t
max_t = args.max_t
grid_length = args.grid_length
grid_pts = args.grid_pts
save_every_nth_timestep = args.save_interval
expm_tolerance = args.expm_tolerance
krylov_subspace_maxdim = args.krylov_subspace_maxdim
pulse_amplitude = args.pulse_amplitude
pulse_frequency = args.pulse_frequency
pulse_duration = args.pulse_duration
cap_width = args.cap_width
cap_height = args.cap_height
savefile = args.savefile
```

If you run your simulation from the root of the git repository, you should be able to launch the above script with command(s)

```
$ export PYTHONPATH=.:$PYTHONPATH
$ chmod +x scripts/qdyn_laser
$ scripts/qdyn_laser --delta-t 1e-1 --max-t ...
```

Note: You probably want to start using `nose2` for tests. To do this, you modify the following arguments in `setup.py`:

```
   test_suite='nose2.collector.collector',
   tests_require=['nose2'],
```

2. Initial state and time-independent Hamiltonian (2 XPs)

To simulate a time-dependent problem, we first need the initial state '$\psi(x, t = 0)$'. You've already calculated this in ex5.1. Extract the code to `quantum_dynamics.tise.get_initial_state` as a function which returns

1. the ground state wavefunction '$\psi(x)$', i.e., the array ' $[\psi(x_0), \psi(x_0 + \Delta x), \ldots, \psi(x_0 + N\Delta x)]$ ' on a computer and
2. the *time-independent* Hamiltonian matrix as a sparse matrix

Remember to write a few unit-tests: Test the numerically obtained '$\psi(x)$' against some analytically solvables results such as particle in a box and harmonic oscillator:

'$V(x) = 0, x \in [0, 5] \Rightarrow \psi(x) = \sqrt{\frac{2}{5}} \sin\left(\frac{\pi}{5}x\right)$ '

'$V(x) = \frac{1}{2}x^2, x \in]-\infty, \infty[\Rightarrow \psi(x) = \frac{1}{\sqrt[4]{\pi}} \exp\left(-\frac{x^2}{2}\right)$ '

In the end you should use this function in `scripts/qdyn_laser` to obtain the initial state and the time-independent Hamiltonian matrix. Something along the lines

```
from quantum_dynamics.tise import get_initial_state

def potential(x):
    return -1.0/np.sqrt(x**2+1)

coordinate_grid = np.linspace(-grid_length/2, grid_length/2, grid_pts)
psi0, H0 = get_initial_state(coordinate_grid, potential)
```

3. Electric field of the laser pulse (0.5 XP)

In addition, we need a Python function that returns the laser electric field at time 't'. Our laser electric field is given by

'$E(t) = E_{\max} \sin^2\left(\frac{\pi}{T}t\right) \cos(\omega t)\, \Theta(t)\Theta(T - t),$ '

where 'T' is the pulse duration, '$E_{\max}$' the electric field amplitude, 'ω' the pulse (carrier) frequency, and '$\Theta(\cdot)$' the Heaviside step function.

Implement the function

```
def laser(t, amplitude, duration, frequency):
    ...
```

in `scripts/qdyn_laser`.

4. Time-dependent part of the Hamiltonian matrix (0.5 XP)

Your `scripts/qdyn_laser` should now look something like

```
#!/usr/bin/env python3

"""Docstring here"""

import this
import that

parser = argparse blabla

# Comment here
def potential(x):
    return -1.0/np.sqrt(x**2+1)

# Comment there
coordinate_grid = np.linspace(-grid_length/2, grid_length/2, grid_pts)
psi0, H0 = initial_state(coordinate_grid, potential)

def laser(t, amplitude, duration, frequency):
    ...
```

Next, write a function

```
def VL_electric_field_interaction(grid, t, amplitude, duration, frequency):
    ...
```

which returns the laser-electron interaction matrix ‘$V_{ij}^{\mathrm{L}}(t) = \delta_{i,j} x_i E(t)$‘ at time ‘$t$‘ as sparse matrix. Here ‘$x_k = x_0 + k\Delta x$‘ and ‘δ_{ij}‘ is the Kroenecker delta.

5. Complex absorbing potential (0.5 XP)

Implement a function `complex_absorbing_potential`in `quantum_dynamics.tdse`. It should return the imaginary potential

$$\text{‘CAP}(x, w, C_{\max}) = \begin{cases} 0, & x \in]-L+w, L-w[\\ -\mathrm{i}C_{\max} \cos^2\left[\frac{\pi}{2w} \min(|x-L|, |x+L|)\right], & x \in [-L, -L+w] \cup [L-w, L], \end{cases}\text{‘}$$

evaluated on a the coordinate grid. Here ‘L‘ is half the grid length, ‘w‘ is the complex absorber width, and ‘$C_{\max}$‘ is the strenght/height of the absorbing potential.

6. Time-evolution from '$\psi(x,t)$' to '$\psi(x,t+\Delta t)$' (1 XP)

Implement the mid-point rule time-evolution operator

'$U(t+\Delta t,t) \approx \exp\left[-\mathrm{i}\Delta t\mathbf{H}(t+\Delta t/2)\right] \doteq \exp\left(-\mathrm{i}\Delta t\mathbf{H}_{\mathrm{mid}}\right)$'

using the Krylov-subspace matrix exponential from `ex5.2`. Implement the evolution operator in `quantum_dynamics.tdse` with call syntax

```
def evolve_timestep(psi, Hmid, delta_t, expm_tolerance, krylov_subspace_maxdim):
    ...
```

This function should calculate and return '$\psi(x,t+\Delta t) = U(t+\Delta t,t)\psi(x,t)$'.

You can extract whatever code you wish from exercise 5.2 (your own or the reference solution). Note that when you embed your solution from `ex5.2` inside the `quantum_dynamics`-package, you might need to change the imports of type `from krylov._krylov_subspace import *` to relative imports like `from ._krylov_subspace import *`.

7. Prepare the time-independent part of the Hamiltonian matrix (0.5 XP)

The time-independent part of the Hamiltonian matrix is

'$\mathbf{H}_{\mathrm{ti}} = \mathbf{H}_0 + \mathbf{V}_{\mathrm{CAP}}$',

where '$\mathbf{H}_0$' is the Hamiltonian from step 2 above and 'V_{CAP}' is the complex absorbing potential (diagonal matrix).

Next, write a section to `scripts/qdyn_laser` calculating '$\mathbf{H}_{\mathrm{ti}}$'.

8. Time-evolution loop (2 XP)

Next, write to `scripts/qdyn_laser` a loop for calculating the time-evolution of the wavefunction. The adjustable parameters were already defined in step 1. Use them here appropriately.

The time-evolution loop should be something like

```
... initialize storage for saving $\psi(x,t)$, pre-calculate H_ti, ...

times = np.arange(...)

for iteration, time in enumerate(times):
    Hmid = H_ti + VL_electric_field_interaction(...)

    psi = evolve_timestep(...)
```

```
    # Save only every `save_interval` time step
    if we_should_save_now:
        save_this_timestep_to_file

save_other_data_needed_for_plotting
```

Remember that the field-interaction matrix should be evaluated at ‘$t + \Delta t/2$‘.

What the parameters adjustable from command line should do:

- `--delta-t` :: time-step ‘Δt‘
- `--max-t` :: final propagation time. The propagation steps should be (‘$t = 0, \Delta t, \ldots, T_{\max}$‘)
- `--save-interval` :: save wavefunction at every nth time-step
- `--grid-length` :: Size of the coordinate grid, ‘$-\frac{L}{2} \leq x \leq \frac{L}{2}$‘
- `--grid-pts` :: Number of points in the coordinate grid
- `--mask-width` :: width of the mask absorber
- `--expm-tolerance` :: relative tolerance for the matrix exponential
- `--krylov-subspace-maxdim` :: maximum dimension of the Krylov subspace for the matrix exponential
- `--pulse-amplitude` :: amplitude of the laser pulse
- `--pulse-frequency` :: frequency of the laser pulse
- `--pulse-duration` :: duration of the laser pulse (can be different from ‘$T_{\max}$‘)
- `--cap-width` :: width of the complex absorbing potential
- `--cap-height`:: strength of the complex absorbing potential
- `--savefile` :: filename for output

9. Visualize the time-dependent electron density (2 XP)

Finally, visualize the electron density ‘$|\psi(x,t)|^2$‘ as a density plot with logarithmic scale for the electron density. Run the simulation with parameters

- ‘$t = 0 \ldots 1000$‘ a.u. with

- time-step ‘$\Delta t = 0.5$‘ a.u.,
- coordinate grid length ‘500‘ a.u.,
- number of grid points ‘$N = 2000$‘,
- matrix exponential tolerance ‘10^{-5}‘, and
- maximum dimension for the Krylov subspace: ‘50‘
- complex absorber width ‘$w = 10$‘ a.u. and height ‘$C_{\max} = 0.5$‘
- with pulse amplitude ‘0.08‘ a.u., duration ‘$T = 1000$‘ a.u., and carrier frequency ‘$\omega = 0.0569$‘ a.u.

Write this visualization script to `scripts/plot_time_evolution` (again, no `.py` suffix). Save the resulting figure as `scripts/demo_time_evolution_plot.pdf`.

You should get something like

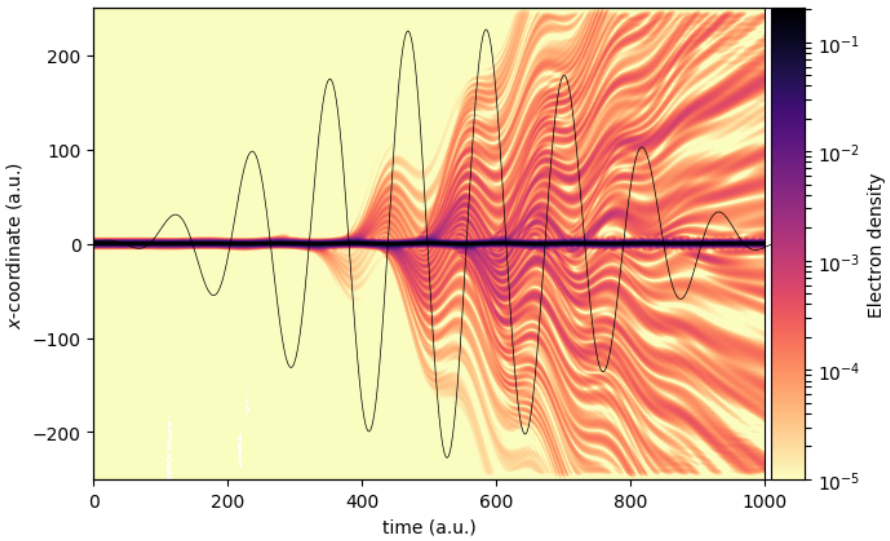

Figure 1: Example result