Three level system

We start out with ordinary differential equations. The most basic approach to solve (integrate)

$$\dot{y} = f(y;t) \ , \tag{1}$$

is to employ the implicit Euler numerical integration scheme. The idea is to replace the time derivative with a finite time difference, $\dot{y} \approx [y(t+\Delta) - y(t)]/\Delta$, where Δ denotes the finite time step. This yields the recipe

$$y(t+\Delta) = \Delta f(y;t) + y(t) \ , \tag{2}$$

which allows us to evolve the function $y(t)$ in time t. The Equ. (1), as well as the discretization of the time derivative can be generalized to vectors, i.e. equations of the form

$$\dot{\mathbf{N}} = f(\mathbf{N};t) \ , \tag{3}$$

where $\mathbf{N} = (N_1, N_2, ..., N_d)$. This way we can evaluate transition rate equations for d-level systems.

Task 1

Compute the time evolution of the differential equation

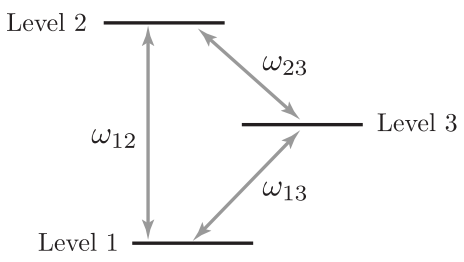
$$\dot{y} = -\gamma y \ , \tag{4}$$

by means of the implicit Euler scheme, where $\gamma > 0$. Discuss the solutions for the cases $\gamma\Delta \geq 2$, $\gamma\Delta \in (1,2]$ und $\gamma\Delta < 1$. Choose a Δ, which keeps the error small within the observed time frame. Think about how to test if your choice of Δ is appropriate – and check!

Task 2

Consider a three level system with the energies $E_1 < E_3 < E_2$. The energy difference between level i und j is given by $\omega_{ij} = |E_j - E_i|$ ($\hbar = 1$). The system is described by the transition rate equation

$$\begin{aligned} \dot{N}_1 &= -\omega_{12}N_1 + \omega_{13}N_3 \\ \dot{N}_2 &= -\omega_{23}N_2 + \omega_{12}N_1 \\ \dot{N}_3 &= -\omega_{13}N_3 + \omega_{23}N_2 \ . \end{aligned}$$



Implement the rate equation as matrix equation and compute the time evolution of the level occupancies N_1, N_2 und N_3 for different level spacings ω_{ij} and initial conditions $N_i(t=0)$. Determine the stationary state, i.e. $N_i^{\mathrm{s}} = N_i(t \to \infty)$, and compare against the analytical solution. Does N_i^{s} depend on initial conditions? Also measure $N(t) = \sum_i N_i(t)$ and explain why this quantity might be helpful to monitor in order to judge the fidelity of the evolution.

Use the python extension NumPy, which allows to define and perform operations with arrays (matrices). Here is an example to illustrate the functionality using the Pauli-matrices: $\boldsymbol{\sigma}_x \cdot \boldsymbol{\sigma}_y = \mathrm{i}\boldsymbol{\sigma}_z$,

```
>>> import numpy as np
>>> Sx = np.array( [[0,  1], [1, 0]] )
>>> Sy = np.array( [[0, -1j], [1j, 0]] )
>>> np.dot(Sx, Sy)
array([[ 0.+1.j,  0.+0.j],
       [ 0.+0.j,  0.-1.j]])
```

Also useful: Appending a NumPy-array to a NumPy-array:

```
>>> a = np.array([0,1])
>>> a = np.vstack((a,np.array([3,4])))
```

such that

```
>>> a
array([[0, 1],
       [3, 4]])
>>> a[0]
array([0, 1])
>>> a[1]
array([3, 4])
```

General remarks:

- Carefully read the instructions, perform *all* tasks, answer *all* questions, for *all* parameters requested!
- Your protocol should reflect your results and experience. Mention hurdles, problems and dead ends.
- Your results must be reproducible: make sure you have defined all simulation parameters in the main text or in the caption of plots.
- Before you submit your protocol: One more time, carefully read the instructions make sure you have performed *all* tasks, answered *all* questions, for *all* parameters required! Check your plots for their correct labelling and legends.