Computer Exercises for Lecture "Computational Photonics" **(Date: 2022-05-31)**

Exercise 04 - The nonlinear Schrödinger equation

Part I

Before you start working on the assignments and before you hand in a solution, read the instructions below.

- You can work on these assignments in groups of up to 3 students. Note, however, that everybody has to hand in his/her own solution.
- Please personalize your completed script `main_ex04_part1_ID.py` by replacing the handle "`ID`" by your student-id. For redundancy, make sure to also include your name and student-id in the completed scripts that you hand in. We therefore provided three metadata variables

```
__STUDENT_NAME__   = "YOUR_NAME_HERE"
__STUDENT_ID__     = "YOUR_STUDENT_ID_HERE"
__GROUP_MEMBERS__ = "YOUR_TEAM_HERE"
```

  at the beginning of the script.
- To complete the code, look out for TODO statements, they will provide further hints!
- Make sure to hand in the completed script into the assignment folder `Exercise04_part1/Solutions`.

Problem 1: Numerical solution of the nonlinear Schrödinger equation (1 + 2 + 2 = 5 points)

The computational problem that we will study in this exercise is an initial value problem with periodic boundary conditions, consisting of the propagation of a complex-valued field $A(z,t)$ along the propagation coordinate z on a periodic t-domain, subject to the nonlinear Schrödinger equation (NSE)

$$\begin{cases} \partial_z A(z,t) = -\frac{i}{2}\beta_2 \partial_t^2 A(z,t) + i\gamma |A(z,t)|^2 A(z,t), & z \geq 0, \quad -t_{\max} \leq t \leq t_{\max}, \\ A(z,-t_{\max}) = A(z,t_{\max}), & z \geq 0, \\ A(z,t)|_{z=0} = A_0(t), & -t_{\max} \leq t \leq t_{\max}. \end{cases} \tag{1}$$

The field $A(z,t)$ describes the complex envelope of an optical pulse and Eq. (1) models its z-propagation through a single-mode fiber governed by the group-velocity dispersion (GVD) parameter β_2, and the nonlinear parameter γ. Therein, the pulse dynamics is resolved in a frame of reference moving at the group velocity of the pulse, see slide 40 of lecture 6. Equation (1) has an exact solution of the form

$$A_{\text{exact}}(z,t) = \sqrt{P_0}\ \text{sech}\left(\frac{t}{t_0}\right) e^{i\frac{\gamma P_0}{2} z}, \tag{2}$$

with $P_0 = |\beta_2|/\gamma\, t_0^2$, defining the fundamental soliton illustrated on slide 41 of lecture 6. Throughout this exercises, Eq. (2) will form the basis of our initial conditions $A_0(t)$ in Eq. (1).

A popular and convenient numerical method for solving Eq. (1) is the split-step Fourier method (SSFM). This approach relies on a Fourier spectral method to account for the linear part on the right-hand-side of the NSE and naturally accounts for the periodicity of the field $A(z,t)$ in t. We here opt for a simple splitting scheme for which we update a field $A(z,t) \to A(z+\Delta z,t)$ by first solving the nonlinear part in the time-domain, and then solving the linear part in the associated Fourier-domain, reading

$$A(z+\Delta z,t) = \mathcal{F}^{-1}\left[e^{i\beta_2\omega^2\Delta z/2}\mathcal{F}\left[A(z,t)e^{i\gamma|A(z,t)|^2\Delta z}\right]\right]. \tag{3}$$

Note that in both cases we used exact solutions for the individual sub-steps, similar to what you have seen on slide 27 of lecture 6. Above, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, ω is the angular frequency conjugate to t and Δz is the stepsize along z. This splitting scheme, discussed, e.g., here, is slightly simpler than the one shown on slide 27 of lecture 6 and its global error is $\mathcal{O}(\Delta z)$.

Download the zip-archive `Ex04_part1.zip` from folder `Exercise04_part1` on StudIp. The unzipped archive contains the following files:

```
Ex04_part1/
  split_step_solver.py - module implementing different split-step Fourier solver
  helper_functions.py - module defining various helper functions
  figures.py - module defining functions that generate figures
  main_ex04_part1_ID.py - script that assembles full simulation runs
```

To keep this small code-project clearly structured, maintainable and reusable, we opted for a modularized implementation. The aim was to keep the main scripts short and readable and to outsource individual software units to modules. The splitting scheme given by Eq. (3) is implemented in module `split_step_solver.py` as function `SSFM_NSE_simple()`. Look up the function and make sure you understand how Eq. (3) is segmented into two sub-steps. Now, consider the fragmented `Python` script `main_ex04_part1_ID.py`. Wade through the code, and consider the problems below.

(a) Quality control is of key importance for all numerical simulations. One way to assure quality is to check conservation laws that are expected to hold for a problem at hand. In general, the NSE given by Eq. (1) exhibits an infinite number of conservation laws. In particular, conservation of energy for the NSE reads

$$\frac{d}{dz}\int |A(z,t)|^2 \, \mathrm{d}t = 0, \tag{4}$$

wherein the integration is performed for an entire period of the considered field pulse. In terms of the field energy $E(z) = \int |A(z,t)|^2 \, \mathrm{d}t$, Eq. (4) simply requires that $E(z+\Delta z) = E(z)$. An immediate way to judge whether your simulation run seems to be reliable is to check the fractional change in energy between the initial and final field configuration of your simulation run, given by

$$f_E = \frac{|E(z_{\max}) - E(0)|}{E(0)}. \tag{5}$$

Consider the function `main_a()`. It specifies simulation parameters for the numerical solution of the NSE, using a fundamental soliton as initial condition. Here, the NSE is solved in terms of the above simple split-step Fourier method. There is nothing to implement for this subproblem, simply run the script. As soon as the propagation routine terminates, the script prints the fractional energy change f_E to the standard output. What does the numerical value on display tell about the completed simulation run? Include your answer in the string-variable `ANSWER_a`.

Hints:

- The limited precision of the floating-point representation of numbers is an issue! Usually, single-precision numbers are reliable up to 6-7 decimal places, and double-precision numbers are reliable up to 15-16 decimal places.

(b) Activate the function `main_b()`. It implements a performance test for the two splitting schemes `SSFM_NSE_simple()` and `SSFM_NSE_symmetric()`, defined in module `split_step_solver.py`. At the moment, the second scheme is merely a re-implementation of the first one and your task is to amend it (see below).

The performance test is based on the propagation of the fundamental soliton given by Eq. (2). As you know from the previous homework (where you considered the dimensionless NSE), this function is an exact solution of the NSE. In other words, it presents an analytical function to which you can compare your numerical solution of the complex envelope $A(z,t)$ for all z and t.

The script will run as is. For a fixed propagation length $[0, z_{\max}]$ it propagates the initial field $A_0(t) = A_{\mathrm{exact}}(z=0,t)$ using the two splitting schemes for a sequence of decreasing step-sizes. At each stepsize it determines the root-mean-squared (RMS) error between the exact solution and the numerical solution that accumulated from 0 through $z_{\max}$. This provides a measure for the *global* error of both splitting schemes. The script will take about 5 sec. to complete and it will generate a plot that compares the RMS error for both splitting schemes.

Now, consider the function `SSFM_NSE_symmetric()` in module `split_step_solver.py`. Amend the function so that it implements the symmetric splitting scheme discussed on slides 27 of lecture 6. It should update a field $A(z,t) \to A(z+\Delta z, t)$ by proceeding in three steps: (i) propagate the linear part for a half z-step, (ii) propagate the nonlinear part for a full z-step, and (iii) propagate the linear part for a further

half z-step. In the notation used above, and using the two intermediate solutions A' and A'', these three steps can be written in the form

$$A'(z,t) = \mathcal{F}^{-1}\left[e^{i\beta_2\omega^2\Delta z/4}\mathcal{F}\left[A(z,t)\right]\right], \tag{6}$$

$$A''(z,t) = A'(z,t)e^{i\gamma|A'(z,t)|^2\Delta z}, \tag{7}$$

$$A(z+\Delta z,t) = \mathcal{F}^{-1}\left[e^{i\beta_2\omega^2\Delta z/4}\mathcal{F}\left[A''(z,t)\right]\right]. \tag{8}$$

This splitting scheme, referred to as symmetric splitting, discussed, e.g., here, has a global error of $\mathcal{O}(\Delta z^2)$. Thus, it outperforms the simple splitting scheme by far!

After you amended the script, observe the generated plot! What do you find? Include a short answer in the string variable `ANSWER_b`.

Hints:

- When you interpret the results, be aware that it shows the simulation data in terms of a log-log plot.

(c) Activate the function `main_c()`. It implements the propagation of a fundamental soliton similar to `main_a()`. Instead of the simple splitting scheme, it uses your implementation of `SSFM_NSE_symmetric()`, i.e. the "better" one. The script runs as is. Upon termination it produces a png-figure in your current working directory.

Review the lecture material of lecture 6 and figure out how you can change the initial condition to realize a soliton of order N. Then, observe the propagation of solitons of higher order.

The propagation distance is fixed to a value that, for the given fiber and pulse parameters, allows to observe the evolution of the initial condition over approximately two soliton periods. Generate a plot showing a soliton of order $N = 6$ and adjust the simulation parameters `Nt` and `Nz` so that it remains periodic. Adjust the bounds of the t-axis and ω-axis, provided as parameter-tuples `tLim` and `wLim` for the figure-generating function `figure_1c()`, so that the generated plot focuses on the relevant parts of the shown intensities. Hand in the generated figure to the folder `Exercise04_solutions`.

Hints:

- You might, e.g., systematically increase N from 1 to, say, 10. Make sure to also consider a case where the soliton order is not an integer number. However, note that it is unlikely that you will observe a perfectly periodic soliton of high order $N = 10$ within a reasonable computing time.
- When you attempt to reproduce the propagation of solitons of order $N = 3$ and $N = 4$, illustrated on slides 44 and 46 of lecture 6, be aware that the scale of the colormap is different. Here, to allow you to assess even slight nonperiodic artifacts of the intensities $|A(z,t)|^2$ and $|A_\omega(z)|^2$, we opted for a logarithmic color-scale.
- Since we use the Fast Fourier-Transform (FFT) as technical basis to realize $\mathcal{F}$ and $\mathcal{F}^{-1}$, make sure that `Nt` is set to an integer power of 2. This will ensure an optimal performance of the underlying algorithm.