PDEs and numerical methods

Fall 2025

Lab sheet #2: Introduction to Finite Differences schemes

24.09.2025

Information: This worksheet provides the fundament for the next worksheets. After the lab session is over, make sure that you understand these concepts and that you can develop code using them. If you don't understand them, practice it! Otherwise, you will have more and more trouble in the next labs.

In this lab we will investigate the Finite Difference (FD) method. This method belongs to one of the oldest ways of discretizing PDEs and is still used in a wide range of applications from climate and weather simulation to full waveform inversion for seismology.

This worksheet is accompanied with **source files** which already contain some boilerplate code. Please download these source files from the website

`https://moodle.caseine.org/course/view.php?id=137`.

It also contains comments related to the locations where you are intended to extend the code:

```
##############################################
# Exercise a: START
##############################################
...

##############################################
# Exercise a: END
##############################################
```

You should first have a look at the source code of the first assignment to see what's provided.

1) Assignment: Error convergence tests

Test function

We like to run error convergence tests using a function where we know the derivative analytically. Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous differentiable function defined on an interval $\Omega = [0, 2]$ given by

$$f(x) = Re(\exp(i\pi x)). \tag{1}$$

Then, its p-th derivative given by

$$f^{(p)}(x) = Re\left((i\pi)^p \exp(i\pi x)\right). \tag{2}$$

In a more realistic application, this function would represent, e.g., the initial condition of the approximated current state of a simulation we use to investigate some fluid mechanics.

Discretization

Next, we use a discretization of the domain with nodal points given by $X = (x_0, \ldots, x_N)$. For sake of simplicity, we choose N equidistantly spaced points $x_n = a + n \cdot h$ with the distance between the nodal points given by $h = (b - a)/N$.

We will also assume **periodic boundary conditions** meaning that $f(x) = f(x+2z)$ for $z \in \mathbb{N}$. E.g., $f(-1) = f(1) = f(3) = \ldots$.

Warmup for the stencil values

Each (explicit) FD scheme can be applied with a stencil given by relative grid indices $K = [k_1, ..., k_S]$ which are relative to the point where the stencil is evaluated and their associated weights $W = [w_1, ..., w_S]$ to approximate the p-th order derivative of the data represented by discrete data points $f(x_i)$.

The application of a stencil (discrete convolution) with stencil array indices K and weights W for the point x_i is then given by

$$g(x_i) = \sum_{1 \le j \le |K|} W[j] \cdot f(x_{i+K[j]}) \tag{3}$$

or using a different notation by

$$g_i = \sum_{1 \le j \le |K|} W_j \cdot f_{i+K_j}. \tag{4}$$

1.1) Computation of the stencil values

As a first step, we will determine the stencil values for different FD schemes.

For a brief validation of this, hand-compute with the Taylor series approach (see lecture / exercises for the FD approximation) to determine the stencil which is at least of 2nd order accuracy for a centered 1st order finite difference.

For numerics code, one can compute these stencils automatically. A function `get_fd_stencil_weights` is provided which returns the weights W. Note, that this function is different from the lecture notes just to get used to using other implementations. It's up to you to reverse engineer it, e.g., by **looking up the reference** provided in the comments.

Read the following hints:

(a) Since we work on an equidistantly spaced grid, the stencil weight vector W (returned by the function) is the same for each point where the finite difference should be computed on, independent to the current point x_i at which the stencil is applied to due to the equidistantly spaced nodes. Therefore, you need to compute the stencil weights only once for one point, e.g., $x = 0$.

(b) Take into account the cell spacing h in the generation of the stencil! You need to hand over actual grid coordinates and not the array indices of the stencil!

(c) The provided function also allows you to specify where to evaluate the finite differences. If you work with relative coordinates as described above, you simply evaluate the finite difference at the current point $x = 0$.

(d) **Test the returned values to match the stencil weights** W determined before for the **centered finite difference**! (How to test it? Just use some print statements and a `sys.exit(1)` to exit the program after printing out the values.)

(e) If you have questions, tell your instructor that you also read this last point so that it becomes clear that you really read this worksheet ;-)

1.2) Application of the stencil

Extend the method `apply_stencil` function which applies the previously determined stencil to approximate the finite differences on all grid points with the help of Eq. (4). Implement this on your own and don't rely on special numpy calls.

Hints:

- You need to use two for loops. One over all grid points and another one over the stencil values.
- Again, test your implementation in a simplified way. Call this function with some particular values where you know the output result.

1.3) Plotting

Write a plotting routine which plots

- the function $f(x)$ itself,
- its analytical derivative $f^{(p)}(x)$ and

- its approximated derivative $\tilde{f}^{(p)}(x)$.

Create a good-looking plot which you'd also use in a "business" meeting.

1.4) Error computation

In order to assess the quantitative errors made by the FD scheme, you'll compute the error of the discrete approximation of the derivative. We will use the infinity / maximum norm given by

$$L_\infty(\vec{a}, \vec{b}) = \max_i |\vec{a}_i - \vec{b}_i|$$

to assess the error.

Complete the function `lmax` to compute the L_∞ error norm. No `numpy` functions which directly compute norms or errors are allowed! Hint: It is possible to implement this in just one line.

1.5) Check convergence rate

Before we advance to the next exercise, we investigate the console output regarding the convergence rate. Check the numerical convergence rate to match closely the expected convergence rate for the individual finite difference test configurations! If anything seems to be wrong, it's now time to start a numerical debugging of your development before advancing to the next step.

1.6) Plot convergence results

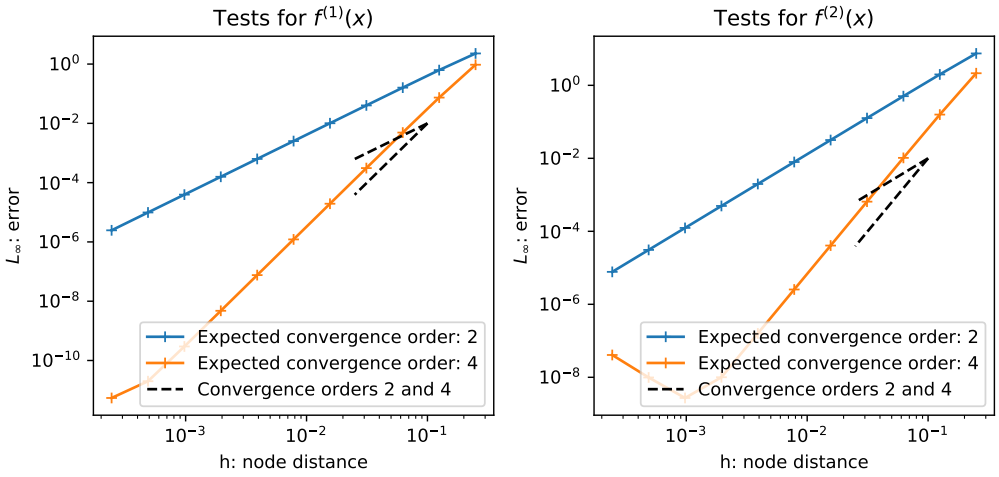
All convergence results should be now available in the array `errors`. Extend the code at the very end by plotting the errors in the following way:

- The errors for the 1st order derivative should be plotted in a subplot at the left side and the errors for the 2nd order derivatives should be plotted on the right handed side of the figure.
- Make sure that the viewer of the plot knows which resolutions were tested by using adequate markers.

1.7) Reference convergence behavior

Extend the previous plot on numerical convergence results by additional lines which plot a reference convergence behavior for a 2nd and 4th order accurate method. These lines should be parallel to the numerical results!

An example plot is provided in the following figure for sake of "inspiration".



2) Transfer functions (optional)

This is an optional exercise for those who already finished the previous one.

The goal of this exercise is to illustrate numerically the notion of transfer functions of a FD scheme with the periodic function $f(x) = \exp(i\omega x)$.

Recap: (See lecture notes in Section 2.2.4.) The transfer function $T(x)$ is the function representing the corresponding operator acting on a particular frequency $\exp(i\omega x)$ in grid space. We start by investigating the transfer function for different discretizations of the first order derivative. Since we can write each function $f(x)$ as a linear combination of Fourier series, we choose a particular one, $\exp(i\omega x)$ to do this investigation.

For an exact derivative we get

$$f'_\omega(x) = \exp'(i\omega x) = \underbrace{i\omega}_{T(x)} \exp(i\omega x)$$

and hence the transfer function $T_\omega(x) = i\omega$.

Using a forward (aka. right-sides, aka. downstream) finite difference scheme

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{5}$$

we can write the finite difference operator as

$$S_\omega^{(fwd)}(x) \approx \frac{\exp(i\omega(x+h)) - \exp(i\omega x)}{h} = \underbrace{\frac{\exp(i\omega h) - 1}{h}}_{T_\omega^{(fwd)}(x)} \exp(i\omega x)$$

with the transfer function given by

$$T_\omega^{(fwd)}(x) = \frac{\exp(i\omega h) - 1}{h}.$$

Similarly, for the centered differences we get the transfer function

$$T_\omega^{(ctr)}(x) = \frac{i \cdot \sin(\omega h)}{h}$$

and for the backward differences we get

$$T_\omega^{(bwd)}(x) = \frac{1 - \exp(-i\omega h)}{h}$$

which we will need later on.

Excursion to amplitude and phase

Make sure to understand the concepts of amplitude and phase by working through this derivation.

Assuming a real-valued solution, the corresponding k-th Fourier mode can be written as

$$\boldsymbol{f}_k(x) = a \sin(2\pi k x + p) \tag{6}$$

with amplitude a and phase p. Its corresponding Fourier coefficient can be written as

$$\begin{aligned}
\tilde{\boldsymbol{f}}_m &= \sum_{n=0}^{N-1} e^{-i2\pi m \frac{n}{N}} \boldsymbol{f}_m\left(\frac{n}{N}\right) \\
&= \sum_{n=0}^{N-1} e^{-i2\pi m \frac{n}{N}} \left(a \sin\left(2\pi m \frac{n}{N} + p\right)\right) \\
&= a \sum_{n=0}^{N-1} \left(\frac{1}{2i}\left(e^{-i2\pi m \frac{n}{N}} e^{i\left(2\pi m \frac{n}{N} + p\right)} - e^{-i2\pi m \frac{n}{N}} e^{-i\left(2\pi m \frac{n}{N} + p\right)}\right)\right) \\
&= aN \frac{1}{2i} e^{ip} - a \frac{1}{2i} \sum_{n=0}^{N-1} \left(e^{-i2\pi m \frac{2n}{N}} e^{ip}\right) \\
&= N \frac{a}{2i} e^{ip}
\end{aligned}$$

where we used

$$\sin(x) = \frac{1}{2i}\left(e^{ix} - e^{-ix}\right).$$

Finally, we can infer the amplitude by "ignoring" the frequency e^{ip} which amplitude is always 1. Solving $\tilde{\boldsymbol{f}}_m = N\frac{a}{2i}e^{ip}$ for a yields

$$a = \left| N^{-1} \tilde{\boldsymbol{f}}_m 2 \right|.$$

Given (or not given) our real valued amplitude we can write

$$\begin{aligned}
\tilde{\boldsymbol{f}}_m &= N \frac{a}{2i} e^{ip} \\
\underbrace{\frac{2i\tilde{\boldsymbol{f}}_m}{Na}}_{lhs=w} &= e^{ip} = \cos p + i \sin p
\end{aligned}$$

which needs to be solved for p and where we will use w in the following phase determination. Note, that the phase is independent to the scalars of the amplitude a, modal number N and the

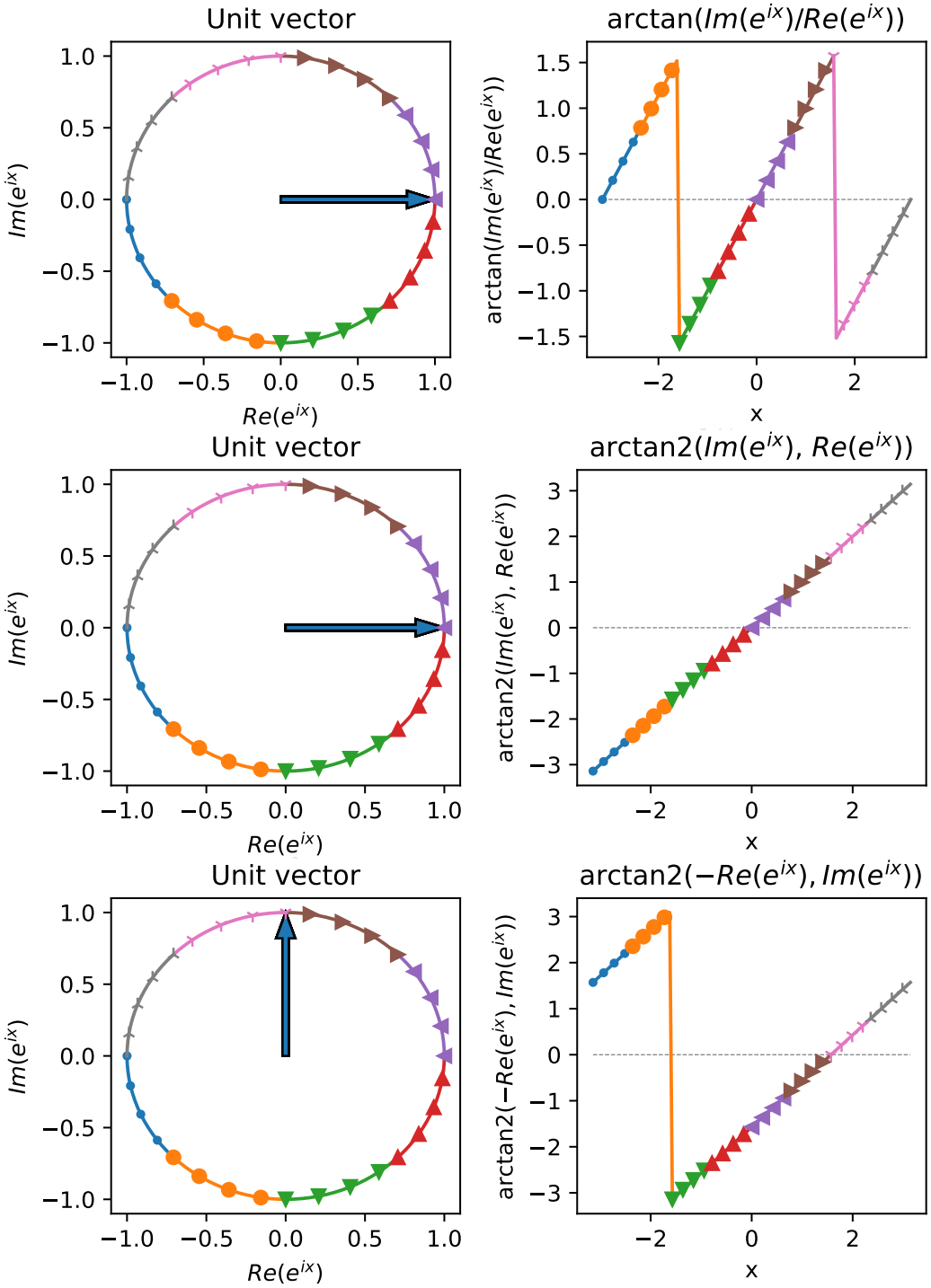


Figure 1: Left: Visualization of phase on the complex plane. Right: Visualization of angle relative to the arrow on the complex plane with the computation based on the point on the unit circle on the left side.

scalar 2. Hence, it is not necessary to compute the amplitude before. Only the multiplication with the imaginary number i plays an important role.

The phase is related to the angle of the particular mode on the unit circle given by Eulers formula

$$e^{ip} = \cos p + i \sin p.$$

It can be computed, e.g., by

$$p = arctan\left(Im(w)/Re(w)\right).$$

However, we loose a certain range of continuity for p due to, e.g.,

$$arctan\left(Im(w)/Re(w)\right) = arctan\left(Im(-w)/Re(-w)\right),$$

see top image in Fig. 1. Alternatively, we can use the special $arctan2$ function yielding

$$p = arctan2\left(Im(w), Re(w)\right)$$

with an example plotted in the top image of Fig. 1 which doesn't suffer from this effect.

2.1) Amplitude amplification factor

Next, we will investigate the amplitude by creating different plots.

To compute the change in amplitude for a given ω (and h), we can use the simple equation

$$e_{Amplitude}(\omega) = \frac{\left|T_\omega^{(fwd)}\right|}{|T_\omega|} = \left|T_\omega^{(fwd)}\right| \tag{7}$$

which provides information on the amplification of the amplitude. Plot this amplification for varying ω on the x axis where you use $h = 1.0$.

Optional: Create such plots for varying h. What do you observe?

2.2) Phase difference

There are various ways to determine the phase difference where we suggest only two. Feel free to develop your own method!

2.2.a) arctan2-style

Using the equations from above, we can compute the phase difference from the exact one to the numerical solution.

Hints:

- The phase difference to the exact one should be 0 which is good for debugging!
- You might run into the challenge of having jumps in the solution. This is caused by the property that a π shifted sine mode is equivalent to the non-shifted one scaled by -1. Therefore, make sure to have all phases within the range $[-pi/2; pi/2]$.

2.2.b) dot-cross

The most straight-forward way is to compute the angle between the reference vector on the complex plane (representing the exact phase) and the one of the numerical method. Once we got the angle, we can use a cross product to finally determine the sign of the angle.

Homework for next time

Finish the exercise 1.