**Physik auf dem Computer** **SS 2017**

# Worksheet 8: Differentiation and Integration

June 22, 2017

## General Remarks

- The deadline for handing in the worksheets is **Monday, June 26th, 2017, 12:00 noon**.
- For this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to your tutor:
  - Johannes Zeman `zeman@icp.uni-stuttgart.de` (Tue 15:45–17:15)
  - Michael Kuron `mkuron@icp.uni-stuttgart.de` (Wed 15:45–17:15)
  - ~~Kai Szuttor `kai@icp.uni-stuttgart.de` (Thu 14:00–15:30)~~
  - **Johannes Zeman `zeman@icp.uni-stuttgart.de` (Thu 14:00–15:30)**
- Please try to only hand in a single file that contains your program code for all tasks. If you are asked to answer questions, you should do so in a comment in your code file. If you are asked for graphs or figures, it is sufficient if your code generates them. You may as well hand in a separate PDF document with all your answers, graphs and equations.
- The worksheets are to be solved in groups of two or three people.

## Task 8.1: Solving the One-dimensional Poisson Equation (6 points)

In this task, you will numerically approximate the solution of the one-dimensional Poisson equation

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\phi(x) = \rho(x)\,. \tag{1}$$

- **8.1.1** (1 point) Using the three-point-midpoint rule (see lecture notes from June 14, 2017), discretize the one-dimensional Poisson equation and write it down.
- **8.1.2** (1 point) With the discretization above, you transformed the differential equation (1) into a system of linear equations

  $$A\vec{\phi} = \vec{\rho}\,. \tag{2}$$

  From your discretization, you can read off the coefficients of the matrix $A$. Let $\phi = 0$ at the boundaries. What does that imply for $\rho$ at the boundaries? Write down the first and last three rows of the equation system (2).
- **8.1.3** (2 points) Implement a Python function `solve_poisson1d_exact(rho,h)` that approximates the solution of the Poisson equation by solving the system of linear equations (2). Here, `rho` is a one-dimensional NumPy-Array of length $N$ that contains $N-2$ values of the charge distribution $\rho$ (The first and last values are required for the boundary conditions!) and `h` is the step size of the spatial discretization.

**Hints**

- To solve the system of linear equations, use the Python function `scipy.linalg.solve()`.
- Consequently, the solution of the system of linear equations is exact, while the solution of the differential equation itself is not.

- **8.1.4** (2 points) Using the Python function `solve_poisson1d_exact(rho,h)` from above, approximate the solution of the Poisson equation for the charge distribution $\rho(x) = \sin(\frac{2\pi}{L}x)$ on the interval $(0,\, L)$, where $L = 10$ at $N \in \{10, 50\}$ points on the interval $[0,\, L]$. (Note the open/closed boundaries of the intervals!)

  Calculate the corresponding analytical solution of the Poisson equation (1) and plot the numerical solutions for both values of $N$ together with the analytical solution. Where do you see the biggest deviations between numerical and analytical solution and why?

## Task 8.2: One-dimensional Integration (4 points)

The error function is a sigmoidal function that is used in statistics and some other areas. It is defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int\limits_0^x e^{-\tau^2}\, d\tau \tag{3}$$

Your job in this task is to implement the error function in Python.

- **8.2.1** (1 point) Implement a Python function `integrate1d_trapezoid(f,a,b,N)` which approximates $\int_a^b \mathrm{f}(x)\, dx$ using the composite trapezoid rule with `N` support points. Do *not* use functions such as `numpy.trapz()` to do so.
- **8.2.2** (1 point) Implement a Python function `erf_trapezoid(x,N)` which computes the error function for `N` points using your previously implemented function `integrate1d_trapezoid(f,a,b,N)`.
- **8.2.3** (1 point) Evaluate the error function $\mathrm{erf}(x)$ at 100 equidistant points $x$ on the interval $[0, 2.0]$ using `erf_trapezoid(x, N)` with $N = 64$, and plot the resulting approximation of the error function.
- **8.2.4** (1 point) Compute a reference value $e_{\mathrm{ref}} = \mathrm{erf}(1)$ using `scipy.special.erf(1)`. Create a loglog plot of the absolute deviation of `erf_trapezoid(x,N)` from $e_{\mathrm{ref}}$ when computing $\mathrm{erf}(1)$ for $N \in \{4, 8, 16, 32, 64, 128, 256, 512\}$. How does the deviation scale with $N$? Explain the origin of the scaling behavior!