Worksheet 9: Differentiation and Integration

June 15th, 2016

General Remarks

- The deadline for handing in the worksheets is **Tuesday, June 21st, 2016, 10:00**.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to `mkuron@icp.uni-stuttgart.de`.

Task 9.1: Solving the One-dimensional Poisson Equation (3 points)

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

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

- **9.1.1** (1 point) Using finite differences, discretize the one-dimensional Poisson equation as shown for the Bessel equation in the lecture notes in section 6.1.3. Write down the equation that corresponds to equation (6.17) in the lecture notes, which allows you to read off the matrix A from the system of linear equations $A\vec{\phi} = \vec{\rho}$.
- **9.1.2** (1 point) Using the discretization from the previous task, implement a Python function `solve_poisson1d_exact(rho,h)` that approximates the solution of the Poisson equation, where `rho` is a one-dimensional NumPy-Array that contains N values of the charge distribution ρ and `h` is the step size between the values of `rho`. Let $\phi = 0$ on the boundaries.

  ### 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.
- **9.1.3** (1 point) Using the Python function `solve_poisson1d_exact(rho,h)` from the 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 $N = 10, 50$ points on the interval. Plot the numerical solutions for both values of N and compare them to the analytical solution. Where do you see the biggest deviations and why?

Task 9.2: One-dimensional Integration (7 points)

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

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} \, \mathrm{d}\tau \tag{2}$$

The job in this task is to implement the error function.

- **9.2.1** (1 point) Implement a Python function `integrate1d_midpoint(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x) \, \mathrm{d}x$ using the midpoint rule with `N` support points.
- **9.2.2** (1 point) Implement a Python function `integrate1d_romberg(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x) \, \mathrm{d}x$ using Romberg's method with `N` support points on the finest level, i. e. use $h = \frac{b-a}{N-1}$ as smallest step size.

  Hint Note that the implementation in the lecture script uses $N = 2^{kmax}$ support points, so choose $kmax$ accordingly.
- **9.2.3** (1 point) Implement a Python function `integrate1d_mc(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x) \, \mathrm{d}x$ using `N` steps of Monte-Carlo Integration.
- **9.2.4** (1 point) Implement the Python functions `erf_midpoint(x,N)`, `erf_romberg(x,N)` and `erf_mc(x,N)` that compute the error function for `N` points with the corresponding integration rules.
- **9.2.5** (1 point) Evaluate the error function $\operatorname{erf}(x)$ on the interval $[0, 2.0]$ at 100 equidistant points x using `erf_romberg` with $N = 64$, and plot the resulting approximation of the error function.
- **9.2.6** (1 point) Compute a reference value $e_{\mathrm{ref}} = \operatorname{erf}(1)$ using Romberg's method with $N = 512$. Create a loglog plot of the accuracy of all implemented methods when computing $\operatorname{erf}(1)$ for $N \in \{4, 8, 16, 32, 64, 128, 256, 512\}$.
- **9.2.7** (1 point) Use the `integrate1d_*` functions to approximate π via

$$\frac{\pi}{4} = \int_0^1 \sqrt{1 - x^2} \, \mathrm{d}x \tag{3}$$

  Create a loglog plot of the accuracy of the different integration schemes versus the number of points $N \in \{4, 8, 16, \ldots, 512\}$. Use `numpy.pi` as reference value. Why does the accuracy behave differently this time?