Worksheet 8: Root Finding and Solving ODEs

June 11, 2013

General Remarks

- Deadline is **Tuesday, 18th June 2013, 10:00**.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Olaf (`olenz@icp.uni-stuttgart.de`; Wednesday, 14:00–15:30)
  - Elena (`minina@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Friday, 15:45–17:15)
- Attach all required files to the mailing. If asked to write a program, attach the *source code* of the program. If asked for a text, send it as PDF or in the text format. We will *not* accept MS Word files!
- The worksheets are to be solved in groups of two or three people.
- The tutorials take place in the CIP-Pool of the ICP in Allmandring 3.

Task 8.1 (4 points): Halley's Method

As shown in the lecture for higher dimensionalities, Newton's method can be derived from the Taylor expansion. Cutting off the expansion after the second term yields Newton's method. To get a higher-order method, the expansion can be cut off after some more terms. When using one further term, one obtains *Halley's method*.

- 8.1.1 (2 points) Derive the formula of Halley's method.

  ### Hints

  - A parabola has two roots. Use the one with the smaller correction term.
  - When deriving the method, it might happen that the denominator becomes 0, which might cause numerical trouble. Use the quadratic complement (*quadratische Ergänzung*) to solve this problem.

- 8.1.2 (1 point) On the home page, you will find the sample solution of worksheet 7 `find_roots.py`. Expand the program by the new method and compare the method's accuracy to the other methods.

- 8.1.3 (1 point) In which cases does Halley's method *not* converge significantly faster than Newton's method?

Task 8.2 (6 points): Solving the Poisson Equation

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

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

- 8.2.1 (1 points) Using finite differences, discretize the one-dimensional Poisson equation as shown for the Bessel equation in the script in section 6.1.3. Write down the equation that corresponds to equation (6.17) in the script, that allows to read off the matrix A from the system of linear equations $A\vec{\phi} = \vec{\rho}$.
- 8.2.2 (2 points) 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 solution of the differential equation itself is not.
- 8.2.3 (1 points) Using the Python function `solve_poisson1d_exact(rho,h)` from the previous task, 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 the analytical solution.
- 8.2.4 (2 points) Approximate the same solution as in task 8.2.3 for $N = 10$ points using the successive overrelaxation method as implemented in the Python function `sor_step(A,b,omega,x)` from chapter 8.1.3 of the script. Plot the solution after 5, 10 and 20 steps of the method at $\omega = 1$.