Physik auf dem Computer **SS 2013**

Worksheet 7: Root Finding

June 4, 2013

General Remarks

- Deadline is **Tuesday, 11th 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 7.1 (6 points): Root finding

- 7.1.1 (1 points) Implement a Python function `newton(f, fp, x)` that finds the root of a function f using Newton's method, where fp is the derivative of the function and x is the initial value of the iteration. Let the function record the difference to the previous iteration (*i.e.* the *accuracy*) in a list, *i.e.* $|x_{n-1} - x_n|$, where the x_i are the successive approximations. Terminate the iteration when an accuracy of 10^{-12} is reached.
- 7.1.2 (1 points) Implement a Python function `bisect(f, a, b)` that finds the root of a function f in the interval $[a, b]$ using bisection. Let the function record the difference to the previous iteration (*i.e.* the *accuracy*) in a list, *i.e.* $|b_i - a_i|$, where the a_i and b_i are the interval boundaries in successive iterations. Terminate the iteration when an accuracy of 10^{-12} is reached.

- 7.1.3 (1 points) Implement a Python function `secant(f, x0, x1)` that finds the root of a function f using the *secant method*, where x_0 and x_1 are the initial x values. Let the function record the difference to the previous iteration (*i.e.* the *accuracy*) in a list, *i.e.* $|x_{n-1} - x_n|$, where the x_i are the successive approximations. $|x_n - x_{n-1}|$. Terminate the iteration when an accuracy of 10^{-12} is reached.
- 7.1.4 (1 point) Use the functions from the previous tasks to compute the root of the function $f(x) = x^2 - 1$ (initial guess for Newton's method $x_0 = 3$, initial guesses for secant method $x_0 = 0$, $x_1 = 3$, initial interval for bisection $[0, 3]$) and the root of the cosine function `numpy.cos` (initial guess for Newton's method $x_0 = 2$, initial guesses for secant method $x_0 = 0$, $x_1 = 2$, initial interval for bisection $[0, 2]$. Create a plot with logarithmic scale on the y-axis that shows the accuracy of the three methods versus the iteration number for both cases.
- 7.1.5 (1 point) As in task 7.1.4, try to use the function to compute the root of the cosine function, but this time start Newton's method with an initial value of $x_0 = 0$. Then use the secant method to compute the root of the cosine function, but start the method with initial values of $x0 = -1$ and $x_1 = 1$. What is happening? Why do the methods fail?
- 7.1.6 (1 point) Summarize the advantages and disadvantages of the three methods.

Task 7.2 (4 points): Compute fractional roots

- 7.2.1 (2 points) Use any of the functions from the previous task to compute $13^{\frac{2}{3}}$, *without* using Python's methods to draw roots.
- 7.2.2 (2 points) Implement a Python function `root(x,k)` that computes $x^{\frac{1}{k}}$ using the function `newton`, *without* using Python's methods to draw roots.