Worksheet 8: Root Finding

May 27, 2014

General Remarks

- The deadline for the worksheets is **Wednesday, 4 June 2014, 10:00** for the tutorial on Friday and **Friday, 6 June 2014, 10:00** for the tutorials on Tuesday and Wednesday.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Johannes (`zeman@icp.uni-stuttgart.de`; Tuesday, 9:45–11:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Shervin (`shervin@icp.uni-stuttgart.de`; Friday, 11:30–13:00)

Task 8.1: Root Finding (5 points)

- **8.1.1** (3 points) Implement three Python functions that find the root of a function:
  - `newton(f, fp, x0)` should use Newton's method to find the root of function f, where fp is the derivative of the function and $x0$ is the initial value of the iteration.
  - `secant(f, x0, x1)` uses the *secant method* to find the root of function f, where x_0 and x_1 are the initial x values.
  - `bisect(f, a, b)` uses bisection to find the root of function f in the interval $[a, b]$.

  Use the functions to compute the root of the function $f(x) = x^2 - 1$ (initial guess for Newton's method $x_0 = 3$, initial interval for the secant method and bisection $[0, 3]$) and the root of the cosine function `numpy.cos` (initial guess for Newton's method $x_0 = 2$, initial interval for the secant method and bisection $[0, 2]$).

  The methods should terminate when they have reached an *accuracy* of 10^{-12}. The accuracy of a given iteration is the difference to the previous iteration in Newton's method and the differences between the two guesses in the other methods.
- **8.1.2** (1 point) Extend the Python functions from the previous task such that they record the reached accuracy in every iteration. Create a plot with logarithmic scale on the y-axis (`matplotlib.pyplot.semilogy`) that shows the accuracy of the three methods versus the iteration number for both cases.
- **8.1.3** (1 point) As in the previous subtask, 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?

Task 8.2: Halley's Method (3 points)

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*.

- **8.2.1** (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.

- **8.2.2** (1 point) Implement the new method in a Python function and extend the program from the previous task to compare the method's convergence with the other methods. In which cases does Halley's method *not* converge significantly faster than Newton's method?

Task 8.3: Computing Fractional Roots (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.