Aschaffenburg University of Applied Sciences
Faculty for Engineering
Prof. Dr. Michael Möckel

Email: Michael.Moeckel@th-ab.de
Telephone: +49 (0)6021 / 4206 - 507

Numerical Methods and Simulation
Module MT 03
Winter term 2022/23

https://moodle.th-ab.de/course/view.php?id=624

Organizational Matters

Problem Set 3

Root finding (dt. Nullstellensuche)

Coursework

Exercise 1: Simple but slow root finding: The bisection method

This method is a simple algorithm for finding a root (dt Nullstelle) $x*$ of the function f(x) which works for simple roots (whenever the function changes sign at the root). It requires two initial approximations for the root, one smaller ($x_l < x*$) and one bigger ($x_r > x*$) than the root. This defines an interval $[x_l, x_r]$ around the root. Then an iterative algorithm starts: The midpoint $x_i = (x_L + x_R)/2$ is calculated where i is the number of the iteration. Then $f(x_i)$ is determined and evaluated: If the sign of $f(x_i)$ agrees with the sign of $f(x_l)$ then x_i is an improved value for x_l, else x_i is an improved value for x_r. This defines a new interval of half length. A simple estimation of the error is given by the length of the interval $x_r - x_l$.

(1) Create a function `[x,e] = bisection1(f, l,r, n)` where f is an anonymous function, l and r are initial values for the left and right boundary of the interval, and n is the number of iterations. x represents an approximation of the root, e a rough estimate of its error. Save it in a file `bisection1.m`.

(2) Implement the algorithm using a `for 1:n ... end` loop.

(3) Document the iterative calculation by creating a matrix with a single line `res = [x, l, r, f(l), f(e), e]` for the initial values and adding a new line with the updated information to the matrix `res = [res; x, l, r, f(l), f(e), e]`. After all iterations have been performed print the matrix with the command `printmat(res, 'Results', ROWS, COLS)` where `ROWS = num2str([0:n])` and `COLS = 'x l r f(l) f(e) e'` are strings with reasonable row and column labels. Also plot the error as a function of i.

(4) Define an anonymous function $f(x) = x^2 - 4$ and test the function by calling it with simple arguments. Then call the function `bisection1` for this function and the initial values l =0, r=10 and n =5. What do you get? Switch to format long and repeat. Increase the number of iterations to $n = 50$.

Exercise 2: Compare Matlab routines for root finding

Use the same function $f(x) = x^2 - 4$ to study how Matlab routines find roots of nonlinear equations. Compare with the output of the bisection algorithm.

(1) Write the function as a vector of its polynomial coefficients $p = [a2, a1, a0]$ and apply the Matlab root finding routine for polynomials `roots(p)`. This routine uses a completely different numerical approach.

(2) Matlab has another routine for calculating roots called `fzero(f, x)` where x is a guess for the root. Use x = 4 and x=0 and compare. This routine uses improved versions of the bisection algorithm.

(3) Matlab provides another independent option for calculating roots based on the more general routine `fsolve(f, x)`. Try it for an initial guess x, e.g. $x = 1.8$. Note that `fsolve` is a powerful tool which has beed designed to solve large systems of nonlinear equations. However, its standard precision is limited. If you need to calculate a result with higher precision, special options are available.

(4) The routines `fsolve` and `fzero` use different algorithms to calculate roots. To see the difference, define the simple function $g(x) = x^2$ as an anonymous function. First try to find its root with `fzero(g,1)` or a similar initial guess. Then try the corresponding `fsolve(g,1)`.

Exercise 3: Newton's method

Newton's method for finding the root of a nonlinear function is based on a first order Taylor expansion. This requires the knowledge of the first derivative. Then the assumption is that a first guess for the root x_0 is close enough to the actual root that higher order terms can be neglected.

$$f(x) \approx f(x_0) + f'(x)(x - x_0) \tag{1}$$

We are looking for the unknown root, i.e. for $f(x) = 0$ and solve for x. This gives

$$x \approx x_0 - \frac{f(x_0}{f'(x_0)} \tag{2}$$

We can use this expression for an iterative algorithm. Starting from the initial guess we can calculate (hopefully) better values x_i for the root (i describes the step of the iteration).

(1) Create a function `[x] = newton1(f, f1, x0, n)` where f is an anonymous function, f1 is its analytic derivative, x_0 is an initial values and n is the number of iterations. The output x represents an approximation of the root.

(2) Implement the algorithm using a `for 1:n ... end` loop.

(3) Document the iterative calculation by creating a matrix with a single line `res = [x, f(x), f1(x)]` for the initial values and adding a new line with the updated information to the matrix `res = [res; x, f(x), f1(x)]`. After all iterations have been performed print the matrix with the command `printmat(res, 'Results', ROWS, COLS)` where `ROWS = num2str([0:n])` and `COLS = 'x f(x) f1(x)'` are strings with reasonable row and column labels. Also implement a plotting command which plots the set of x_i values over the iteration index.

(4) Define an anonymous function $f(x) = x^3 - 5$ and its derivative $f1(x)$ by hand and test the function `newton1.m`. Compare with a direct Matlab result.

(5) Newton's method can be very fast and efficient whenever the derivative at all intermediate points x_i is finite. An example where things can go wrong is given by the function $f(x) = x^{1/3}$. Draw the function and confirm that the root should be at $x^* = 0$. Calculate the derivative analytically. Then call the function `newton1.m` with $x_0 = 0.1$ (pretty close to the root) and $n = 10$. What happens?

Homework

Exercise 4: The regula falsi (False position) method

A little modification of the bisection algorithm leads to an alternative method: Instead of taking the midpoint of the interval the root of the linear function defined by the two points $(l, f(l))$ and $(r, f(r))$ can be used to update the boundary of the interval. This line is a *secant* of the function. Hence the regula falsi method is counted among the secant methods for root finding.

(1) Set up the equation $y = l(x)$ for the secant line on paper. Solve for the root, i.e. for x with $l(x) = 0$.

(2) Rename and save the function `bisection1` as `secant1.m`. Replace the midpoint rule by the equivalent expression for the root of $l(x)$.

(3) Call the function with the same parameters as in exercise 2.

(4) What do you observe when you follow the evolution of the boundaries of the interval? Note that the error gives a wrong result. Delete the error calculation from the function. Calculate instead the difference between two successive values of x_i and x_{i+1}. Hint: Introduce a new variable `xold`.