Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
24.04.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Aufgabe 6.1. Given a differentiable function $f : [a, b] \to \mathbb{R}$ and $x \in [a, b]$, the derivative $f'(x)$ can be approximated by the different quotient

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{for } h > 0.$$

Write a function `double diff(double (*f)(double), double x, double h0, double tau)`, which computes the sequence $\Phi(h_n)$, where $h_n := 2^{-n} h_0$ ($n \in \mathbb{N}$), until

$$|\Phi(h_n) - \Phi(h_{n+1})| \le \begin{cases} \tau & \text{if } |\Phi(h_n)| \le \tau, \text{or} \\ \tau\,|\Phi(h_n)| & \text{else.} \end{cases}$$

The function then returns the value $\Phi(h_n)$ as an approximation of $f'(x)$. Use `assert` to check whether $\tau, h_0 > 0$ holds. The function uses a suitable implementation of the object function `double f(double x)`. Finally, write a main program which reads x, h_0 and τ from the keyboard and prints the value of $\Phi(h_n)$ on the screen. How can you test your code? What are good examples? Save your source code as `diff.c` into the directory `serie06`.

Aufgabe 6.2. An alternative root-finding algorithm (see also the bisection method from the lecture) is the so-called *secant method*. Let $f : [a, b] \to \mathbb{R}$. Given two initial guesses x_0 and x_1, the appromation x_{n+1} is obtained as the root of the line which connects the points $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$, i.e.,

$$x_{n+1} := x_n - f(x_n)\,\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}.$$

Write a function `double secant(double (*f)(double), double x0, double x1, double tau)`, which performs the above iteration until either

$$|f(x_n) - f(x_{n-1})| \le \tau$$

or

$$|f(x_n)| \le \tau \quad \text{and} \quad |x_n - x_{n-1}| \le \begin{cases} \tau & \text{for } |x_n| \le \tau, \\ \tau|x_n| & \text{else.} \end{cases}$$

In the first case, print a warning to inform that the result is presumably wrong. The function returns x_n as an approximation of the root z_0 of f. Use `assert` to check whether $\tau > 0$ holds. The function employs a suitable implementation `double f(double x)` of the object function f. Moreover, write a main program, that reads x_0 and x_1 from the keyboard and displays x_n. How can you test your code? What are good examples? Save your source code as `secant.c` into the directory `serie06`.

Aufgabe 6.3. A well-known root-finding algorithm is the *Newton method*. Let $f : [a, b] \to \mathbb{R}$. Given an initial guess x_0, define the sequence $(x_n)_{n\in\mathbb{N}}$ via

$$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1}) \quad \text{for } n \ge 1.$$

Implement the algorithm in a function

`double newton(double (*f)(double), double (*fprime)(double), double x0, double tau)`,

which, given x_0 and a tolerance $\tau > 0$, performs the Newton iteration until

$$|f'(x_n)| \leq \tau$$

or

$$|f(x_n)| \leq \tau \quad \text{and} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{for } |x_n| \leq \tau, \\ \tau|x_n| & \text{else.} \end{cases}$$

In the first case, print a warning to inform that the result is presumably wrong. The function returns the value x_n of the approximate root. Use `assert` to check whether $\tau > 0$ holds. In the function `newton`, realize the function $f : [a, b] \to \mathbb{R}$ and the corresponding derivative $f' : [a, b] \to \mathbb{R}$ as function pointers. The function gets function pointers of `double f(double x)` and its derivative `double fprime(double x)`. Write a main program, which reads x_0 and τ from the keyboard and prints x_n to the screen. How can you test your code? What are good examples? Save your source code as `newton.c` into the directory `serie06`.

Aufgabe 6.4. The Newton method from Exercise 6.3, besides a function `f` to evaluate the object function, also requires a function `fprime` to evaluate the derivative f' of f. Alternatively, you might replace $f'(x_k)$ with the different quotient $\Phi_h(x_k)$ from Exercise 6.1. Realize this idea in a function `double newton2(double (*f)(double), double x0, double h0, double tau)`, which implements the Newton method from Exercise 6.3 replacing the derivative $f'(x_k)$ with the result of `diff(f, xk, h0, tau)`. Use `assert` to check whether $\tau, h_0 > 0$ holds. How can you test your code? What are good examples? Save your source code as `newton2.c` into the directory `serie06`.

Aufgabe 6.5. Write a function which computes and returns the minimum, maximum, and the mean value $\frac{1}{n}\sum_{j=1}^{n}$ of a given vector $x \in \mathbb{R}^n$. Write two versions of the function with the following signatures:

```
double* minmaxmean1(double* x,int n);
void minmaxmean2(double* x,int n,double* min,double* max,double* mean)
```

Additionally, write a main program that reads in a vector $x \in \mathbb{R}^n$ and prints out the output of `minmaxmean1` and `minmaxmean2`. Save your source code as `minmaxmean.c` into the directory `serie06`.

Aufgabe 6.6. Write a function `void unique(double* x, int* n)`, which sorts a vector $x \in \mathbb{R}^n$ in ascending order, eliminates all entries that appear more than once, and returns the shortened vector. For instance, the vector $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ should be replaced by the vector $x = (1, 3, 4, 5) \in \mathbb{R}^4$. Work with dynamically allocated memory (the input vector must be overwritten with the shortened one, in particular the length must be adjusted accordingly). Write a main program that reads the vector x as well as its length n from the keyboard and prints out the shortened vector. Test your implementation with suitable examples. Save your source code as `unique.c` into the directory `serie06`.

Aufgabe 6.7. Where are the errors in the program? Explain why!

```
#include <stdio.h>

void square(double* x)
{
  double* y;
  x=(*y)*(*x);
}

int main(){
  double x=2.1;
  square(&x);
  printf("x^2=%f\n",x);
  return 0;
}
```

Change *only* the function `square`, such that the output of the code is correct.

Aufgabe 6.8. Explain the differences between variables and pointers. What are advantages resp. disadvantages of these?
Write a function `swap` that swaps the contents of two variables `x`, `y`. What is the problem with the following code?

```
void swap(double x, double y)
{
   double tmp;
   tmp = x;
   x = y;
   y = tmp;
}
```

Save your source code as `swap.c` into the directory `serie06`.