Dr. Ole Klein
M.Sc. Robin Görmer, M.Sc. Philipp Thiele

Lima, Oct. 25, 2019

Spring School on the Introduction on Numerical Modelling of Differential Equations – Programming Exercise 3

Exercise 3.1 [Gaussian Elimination with hdnum]

Write a new headerfile `gauss.hh`, that implements the template function

```
template<typename NUMBER>
void gauss( hdnum::DenseMatrix<NUMBER>& A, // Input A
       hdnum::Vector<NUMBER>& x        // Output x
       hdnum::Vector<NUMBER>& b        // Input b
           )
{
  ...
}
```

for solving the linear equation system $Ax = b$ using gaussian elimination. This headerfile will be needed to compile and execute the program `gaussmain.cc`. Compile the program for both data types `double` and `float` and check the maximal dimension n for which the equation system gives a correct solution.
Hint: To get access to the `hdnum` library type in the following command

```
git clone https://parcomp-git.iwr.uni-heidelberg.de/Teaching/hdnum
```

Exercise 3.2 [Polynomial Interpolation]

All programmed functions in this exercise should accept a *template* parameter to allow different representations of the real numbers (`float`, `double`, etc.).

(a) Write a function that evaluates the interpolating polynomial for a given function $f\colon \mathbb{R} \to \mathbb{R}$ at a given point x. The function should be given as points $(x_i)_{i=1}^n \in \mathbb{R}$ and matching values $(y_i)_{i=1}^n$, i.e.

```
template<typename T>
T interpolation(T x, std::vector<T> x_i, std::vector<T> y_i){...}
```

(b) Write a program that interpolates the following functions on the interval $I = [-1, 1]$ with equidistant points $x_i = -1 + ih, \quad i = 0, \dots, n$ with $h = 2/n$. The degree of the interpolating polynomial n should be chosen as $n = 5, 10, 20$.

$$f_1(x) = \frac{1}{1 + x^2}$$
$$f_2(x) = \sqrt{|x|}$$

Evaluate the polynomials on a fine grid (1000 grid points) and plot the results using gnuplot. Compare the plots to the actual functions. What do you see?

Exercise 3.3 [Numerical Differentiation]

(a) Write a program `numdiff.cc`, that computes the second derivative of $\sinh(x)$ at $x = 0.6$ with the second differential quotient for a given h

$$a(h) := \frac{\sinh(x + h) - 2\sinh(x) + \sinh(x - h)}{h^2} \approx \frac{d^2}{dx^2} \sinh(x).$$

Calculate the values $a(h_i)$ for $h_i = 2^{-i}$, $i = 1, \dots, 20$ and compare with the exact value of the second derivative.
Hint: $\frac{d^2}{dx^2} \sinh(x) = \sinh(x)$. The function $f(x) = \sinh(x)$ is available in C++: Using the library `<cmath>` you can call the function `std::sinh(double x)`.

(b) The error decreases until $i = 12$. Calculate with your output the number j such that the error is of order $\mathcal{O}(h^j)$. Explain why the approximations for smaller h become worse.

(c) Examine how the extrapolation to the limit can be used to improve the numerical values. Write a function, that has the input arguments $\widetilde{h}_j$, $j \in \{0, \dots, k\}$ and computes $a(0)$ using the extrapolation of the limit. Calculate for $h_i = 2^{-i}$, $i = 1, \dots, 10$ the approximation of $a(0)$ with two values $(h_i, h_i/2)$, resp. three values $(h_i, h_i/2, h_i/4)$. Examine the error.

(d) **Bonus:** Use the fact that $a(h)$ can be represented as series in h^2 for the extrapolation and use the pairs $(h_i^2, a(h_i))$ instead of $(h_i, a(h_i))$. Explain how this change affects the rate of convergence of the error.
N.B.: This modification is also called Richardson extrapolation.