

## Worksheet 2: Gauss Elimination

April 20th, 2016

### General Remarks

- The deadline for handing in the worksheets is **Tuesday, April 26th, 2016, 10:00**.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to `mkuron@icp.uni-stuttgart.de`.
- 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.
- The worksheets are to be solved in groups of two or three people. We will not accept hand-in-exercises that only have a single name on it.
- The tutorials take place in the CIP-Pool of the Institute for Computational Physics (ICP) in Allmandring 3 on Wednesday at 15:45.

### Task 2.1: Gauss Elimination (3 points)

Copy the Python program `gauss.py` or the IPython Notebook `gauss.ipynb` from the home page to your home directory.

The program contains the function `backsubstitute(A, b)`, that solves the equation  $Ax = b$  for the case where  $A$  is an upper triangular  $N \times N$  matrix.

Furthermore, it contains the skeleton of a function `gauss_eliminate(A,b)`, that shall do a Gauss elimination for an  $N \times N$  square matrix, and the function `solve(A,b)`, that shall use the previous two functions to solve the linear equation system  $Ax = b$ .

Implement the function `gauss_eliminate(A,b)`. Do not use any functions from `scipy.linalg` etc. to do so.

**Hint** To test whether the function works, `gauss.py` contains two matrices  $A_1$  and  $A_2$ , two vectors  $b_1$  and  $b_2$  and the solutions  $x_1$  and  $x_2$  that solve the equation  $A_i x_i = b_i$ . The function `solve(A,b)` should be able to solve the first of these equations, while the second can only be solved after task 2.3 has been completed.

### Task 2.2: Interpolating Polynomial (3 points)

The interpolating polynomial of a set of  $N$  points  $(x_i, y_i)$  is the polynomial  $p(x)$  of  $(N - 1)$ th degree that passes through all of these points. When the points are chosen from another function  $f(x)$  (so that the points are given by  $(x_i, f(x_i))$ ), the polynomial is an approximation of the function. The interpolating polynomial is given by

$$p(x) = \sum_{i=0}^N a_i x^i. \quad (1)$$

where the coefficients  $a_i$  are defined by the set of  $i$  equations  $p(x_i) = f(x_i)$ .

- **2.2.1** (1 point) Write a Python program that uses the function `solve()` from the previous task to determine the coefficients  $a_i$  of the interpolating polynomial of the function `numpy.sin()` for 4 equidistant points on the interval  $[0, 2\pi]$ . Do not use any functions from `scipy.interpolate` etc. to do so.
- **2.2.2** (1 point) Write a function `polyeval(a, x)` that evaluates the polynomial  $p(x)$  for the given set of coefficients  $a$ . Do not use any functions from `scipy.polynomial` etc. to do so.
- **2.2.3** (1 point) Write a Python program that plots `numpy.sin()` and its interpolating polynomial on the interval  $[0, 2\pi]$  (as in figure 1).

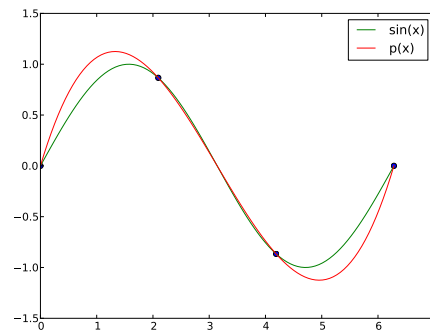


Figure 1:  $\sin(x)$  and the interpolating polynomial  $p(x)$

### Task 2.3: Gauss Elimination and Pivoting (4 points)

- **2.3.1** (2 points) Extend the function `gauss_eliminate()` from task 2.1 such that it implements the Gauss elimination with partial pivoting (Spaltenpivotisierung). With this function, it should be possible to solve  $A_2x_2 = b_2$  from Task 2.1.
- **2.3.2** (2 points) Extend the function `gauss_eliminate()` and `solve()` from task 2.1 such that it implements the Gauss elimination with complete pivoting (Totale Pivotisierung).