

# Worksheet 3: LU Decomposition and Taylor Polynomials

April 24, 2014

## General Remarks

- The deadline for the worksheets is **Wednesday, 30 April 2014, 10:00** for the tutorial on Friday and **Friday, 2 May 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 3.1: LU-Decomposition (5 points)

As you will learn later in the lecture, when you are able to solve linear equation systems (*e.g.* via Gauss elimination), this can be used to approximate numerical solutions of differential equations, for example the Poisson equation  $\Delta\Phi(r) = -\rho(r)$  where  $\Phi(r)$  is the electrostatic potential and  $\rho(r)$  is the charge distribution.

In this task, you will approximate the solution of the Poisson equation in the two-dimensional case by solving  $A\bar{\Phi} = \bar{\rho}$  where  $\bar{\Phi}$  and  $\bar{\rho}$  are the discretized electrostatic potential and charge distribution, respectively, and  $A$  is a correctly constructed matrix.

The Python program `ws3.py` and the IPython Notebook `ws3.ipynb` from the homepage demonstrate how to use Gauss elimination to compute the solution of the Poisson equation for 2d-systems of randomly distributed charges and how to time the runtime of a Python function. To be able to run it, you will have to put the file `poisson2d.py` from the homepage into the same directory, as `ws3.py`.

- **3.1.1** (1 point) When executing the code, one notices that solving the equation for all  $50 \times 50$ -system is already relatively costly.

Extend the Python program or notebook such that it measures the runtime  $t$  for computing the Gauss elimination of a *single*  $N \times N$ -system for  $N \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$  using the function `timeit.timeit`. Create a plot that shows the timing  $t$  against  $N$  in double-logarithmic scale. Estimate the time that it would take to solve the Poisson equation for a  $1000 \times 1000$ -system.

- **3.1.2** (2 points) Extend the Python code such that it uses LU-factorization to solve the equation via `scipy.linalg.lu_factor` and `scipy.linalg.lu_solve`. Measure the runtime of both functions separately for the different values of  $N$ . Create a plot that allows to compare the runtimes of the Gauss elimination, the LU factorization, and LU solving against  $N$  in double-logarithmic scale.

- **3.1.3** (2 points) Using LU-factorization, speed up the computation of the potential for 25 charge distributions. Show your success by computing the potential for 25  $100 \times 100$  charge distributions.

### Task 3.2: Taylor Polynomials (5 points)

In this task, you are entitled to plot the Taylor polynomials of the following functions on the specified domains:

Name	Definition	Domain
Sine Function	$f(x) = \sin x$	$[0, 2\pi]$
Runge Function	$g(x) = \frac{1}{1+x^2}$	$[-5, 5]$
Lennard-Jones Function	$h(x) = x^{-12} - x^{-6}$	$[1, 5]$

- **3.2.1** (3 points) Calculate the coefficients of the truncated Taylor series of the sine function  $f(x)$  at  $x_0 = 0$ , the Runge function  $g(x)$  at  $x_0 = 0$  and the Lennard-Jones function at  $x_0 = 1$  up to 10th degree. Use the Python class `numpy.poly1d` to define the  $k$ -th order Taylor polynomials of  $f(x)$ ,  $g(x)$  and  $h(x)$  for  $k \in \{3, 5, 10\}$  at arbitrary  $x$ .
- **3.2.2** (2 points) For each of the functions  $f(x)$ ,  $g(x)$  and  $h(x)$ , create a plot that shows the function and their respective  $k$ -th degree Taylor polynomials ( $k \in \{3, 5, 10\}$ ) on the specified domain.

#### Hints

- You may use Mathematica or Wolfram alpha to compute the Taylor series.
- The Python class `numpy.poly1d` is used as follows:

```
# f is the polynomial f(x) = 3*x**2 + 2*x + 1
# Note the order of the coefficients!
f = numpy.poly1d([3,2,1])
# Compute the value of the polynomial at x=42
print f(42)
```

- Take care to handle the argument of the Taylor polynomial of  $h(x)$  correctly!