Scientific Computing I

Winter semester 2015/2016
Prof. Dr. Marc Alexander Schweitzer
Sa Wu

Exercise sheet 0.

for revision and warmup, not graded, review in tutorials

2015-10-28, 2015-10-30

- The website of this lecture can be found at
`http://schweitzer.ins.uni-bonn.de/teaching/lectures/ws15_v3e1/`
- For admission to the exams at the end of the lecture *50%* of achievable points from normal excercises and *50%* of achievable points from programming exercises are required.
- Please, feel free to submit your homework assignments in groups of up to 3 students. Be aware, that this this is intended such that you only have to produce one neat, legible submission per 3 students. However, for learning and a succesful exam we recommend all participants to work on all exercises.
- Tutorials for this lecture take place in the seminar room Seminarraum 5.002 on floor 5 of HRZ, Wegelerstraße 6
  - Wednesday, 0800-1000 and
  - Friday, 1200-1400 .

  They start on Wednesday, 2015-10-28 and Friday, 2015-10-30.
- Because of the conference "Panorama of Mathematics" (`http://www.hcm.uni-bonn.de/events/eventpages/2015/panorama`) there will be no lecture on Thursday, 2015-10-22. Participation in the conference is recommended.
- Bacause sufficient demand as been met, the lecture will be held in English.
- This first worksheet is inteded for revision and preparation. In particular, it should be used to refresh and (re)introduce Python/NumPy/matplotlib.

Programming exercise 1. (Python/NumPy/matplotlib Revision)

a) Get to know your colleagues and find people to submit homework with together.

b) Get a working installtion of Python (`http://www.python.org`, `http://ipython.org`), NumPy (`http://numpy.org`) and matplotlib (`http://matplotlib.org`). These packages are also installed on the computers in the CIP pools `http://www.iam.uni-bonn.de/pcpool/start/` in Wegelerstraße 6 and Endenicher Allee 60. The same applies to the practical course room 6.020 in in Wegelerstraße 6. One easy to install (larger) package with everything necessary is Anaconda (`https://www.continuum.io/why-anaconda`).

c) Alternatively, we offer a VirtualBox-Image, known from the previous lecture Einführung in die Numerik, with necessary packages preinstalled. Please, find it at the link `http://ins.uni-bonn.de/teaching/vorlesungen/Numerik_SoSe15/software/software.html`.

d) Get to know Python/NumPy/matplotlib by plotting the Lagrange polynomials for the abscissae $x_0 = -1, x_1 = 0, x_2 = 1$ all into one picture. The polynomials are given by

$$L_i = \prod_{j=0,j\neq i}^{2} \frac{x - x_j}{x_i - x_j}$$

for $i = 0, 1, 2$.

e) Define a map $L_i \mapsto$ "'Name of student in your submission group"' and create a legend for the plot accordingly.

f) Create a figure with 3 subplots containing one of the Lagrange polynomials each. Also, add captions to each of the subplots.

Programming exercise 2. (A simple inital value problem)

Consider the initial value problem

$$\epsilon y'(x) + y(x) = 0 \qquad\qquad y(0) = 1$$

from the lecture.

a) What is the exact solution y_ϵ of said initial value problem for $\epsilon \neq 0$?

b) For $\epsilon = 2^{-k}, k = 1, \ldots, 10$ plot the exact solution y_ϵ on $[0, 1]$. Use a reasonable plotting command that enables one to see the differences caused by different ϵ. A list of MATPLOTLIB plotting command can be found at `http://matplotlib.org/api/pyplot_summary.html`.

c) Implement at method that, for given ϵ, mesh width $h = \frac{1}{N}$ and nodes $x_i = ih, i = 0, \ldots, N$, computes approximations $y_{\epsilon,i} \approx y_\epsilon(x_i)$ with the explicit Euler method.

d) For $\epsilon = 2^{-1}$ and $\epsilon = 2^{-10}$ produce reasonable convergence plots for $N = 2^1, \ldots, 2^{10}$ using the error

$$e_{\epsilon,N} = \max_{i=0,\ldots,N} |y_\epsilon(x_i) - y_{\epsilon,i}| \, .$$

Exercise 1. (Preconditioned CG method)

Let $A, C \in \mathbb{C}^{n\times n}$ be hermitian, positive definite and $b \in \mathbb{C}^n$.
Using the inner product $\langle x, y\rangle_C = \langle x, Cy\rangle$, consider the steps of the CG method for the system of equations

$$(AC)\,y = b\;, \qquad\qquad x = Cy\;.$$

Use algebra to reformulate the steps of the CG method for this problem and inner product such that it yields an algorithm for the approximation of x using just the regular Euclidean inner product. Two helpful quantities for this are

$$e_k := Cd_k\;, \qquad\qquad s_k := Cr_k\;,$$

where r_k are the residuals and d_k the search direction from the CG method.
The resulting algorithm is also know as the Preconditioned Conjugate Gradient (PCG) method. There are different approaches for the choice of the preconditioner C. The method can alternatively also be constructed based on $(CA)\,x = Cb$.

Programming exercise 3. (PCG for sparse matrices)

In many practical applications linear systems have to solved. However, the involved matrices are often sparse, i.e. the number of nonzero entries is significantly lower than the dimension of the matrix. In most cases a sparse matrix $A \in \mathbb{R}^{n\times n}$ will only have $O(n)$ instead of $O(n^2)$ nonzero entries. This can be often alleviated for a significant reduction of algorithms used on these sparse matrices. In this exercise, we will use so called sparse matrices stored in CSR format, for an efficient PCG method.

a) Read the documentation of the module `scipy.sparse`.
`http://docs.scipy.org/doc/scipy/reference/sparse.html`

b) In particular, read documentation on construction of CSR matrices. Construct

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 16 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 64 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 256 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1024 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4096 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 16384 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 65536 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 262144 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1048576 \end{pmatrix}$$

as a CSR matrix and check the number of non zero entries.

c) Implement a Python method for efficient computation of the inverted diagonal $D(A)^{-1}$ of a CSR matrix from `scipy`.

d) Using c), implement the PCG method for solution of $Ax = b$ using the preconditioner $C = D(A)^{-1}$. The method shall stop when a maximum number of iterations `maxIt` or an error bound $\|Ax - b\| <$ `eps` has been reached.

e) Using d), compute a solution to $Ax = b$ with A from b, $b = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T$, inital guess $x_0 = 0$, `maxIt`=100 and `eps`=10^{-13}.