Computational Physics (SS 2020)

Prof. Dr. Riccardo Mazzarello

Sheet 4 (20 points)
Submission: 7 May 2020, 10:00 a.m.

Remarks

- The exercise sheets can be solved in teams of two students.
- Provide a single PDF-file containing the written solutions of all exercises, i.e. analytical derivations, numeric results, their interpretation, etc. If it contains a scan, please make sure it is well readable. Plots can either be embedded into this file or, alternatively, given separately (e.g. as PNG-files) and referred to in the PDF.
- Use a current version of Python 3 for your implementations.
- Please hand in your solutions (PDF-file with answers, source code files, and, if not embedded in the PDF, PNG-files of plots) as a **single** .zip-file named

  Surname1-MatNum1-Surname2-MatNum2-SheetX.zip,

  where MatNum is your matriculation number, by uploading it to RWTH Gigamove (`https://gigamove.rz.rwth-aachen.de`). Please send the download link using your RWTH address to

  brancaccio@physik.rwth-aachen.de,

  and write in the object of the email

  CP SheetX - Surname1-MatNum1-Surname2-MatNum2.

1. Relaxation methods for the Laplace equation (10 points)

An architect wants to cover a square area

$$-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}, \quad -\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$$

with tissue that is fixed along the boundary of the square by arches on all four edges:

$$\Phi\left(x = \pm\frac{\pi}{2}, y\right) = \cos(y), \quad \Phi\left(x, y = \pm\frac{\pi}{2}\right) = \cos(x).$$

The cover is supposed to be a minimal surface. Minimal surfaces are described by the Laplace equation if the field Φ is interpreted as the height of the surface above a two-dimensional plane V. Dirichlet boundary conditions define the height of the surface along the boundary ∂V.

- Implement the Jacobi relaxation, the Gauss-Seidel relaxation, and the successive overrelaxation (SOR) in combination with Chebyshev acceleration. Each method shall be implemented as a Python function that accepts two arguments: a two-dimensional grid (numpy array) of arbitrary size and the error threshold. The function shall return the relaxed grid.

- Use a discretization of the square area with $N \equiv N_x = N_y = 81$ such that the first and the last discretization point along one axis (with index 0 and index $N-1$) lie on the boundary (where the value of Φ is fixed) whereas the remaining $N-2$ points along one axis lie inside the square area. For each of the three algorithms, start the iteration with $\Phi^{(0)}(\vec{r}_m) = 0$ on the interior points and use a convergence threshold of 10^{-5} for the successive error

  $$\delta\Phi = \max_m \left| \Phi^{(k)}(\vec{r}_m) - \Phi^{(k-1)}(\vec{r}_m) \right| .$$

  For each of the algorithms, how many iterations are necessary to reach the desired accuracy of 10^{-5}?
- For each of the algorithms, visualize the intermediate grid after 100 iterations in a 3D plot showing the shape of the covering tissue. Choose identical perspectives for an easy comparison. Embed the three plots in your PDF or include them in a *single* PNG-file. (For example, use `Axes3D` from `mpl_toolkits.mplot3d` in combination with `np.meshgrid`.)

2. Conjugate-gradient method (10 points)

(a) Implement the steepest-descent (SD) and the conjugate-gradient (CG) method:

- Implement both SD and CG following the descriptions given in the lecture notes (in particular, use the Hestenes-Stiefel scheme for CG). Use variable names similar to those in the lecture. Each method shall be implemented as a Python function that accepts the following arguments: the matrix A, the vector $\boldsymbol{b}$, an initial guess $\boldsymbol{x}$, and the threshold value ε. *The convergence condition for both methods is supposed to be fulfilled as soon as the magnitude of the current direction vector $\boldsymbol{n}_k$ is less than the threshold ε.*
- Test your implementations on the linear set of equations $A\boldsymbol{x} = \boldsymbol{b}$ with the 10×10 matrix A provided in RWTHmoodle ("CG_Matrix_10x10.dat"), $\boldsymbol{b} = (1, ..., 1)^T$, and $\boldsymbol{x} = \boldsymbol{b}$ as an initial guess. Use the threshold value $\varepsilon = 10^{-10}$. Report the values of $x[0]$ and $|\boldsymbol{x}|$ obtained from SD and CG.
- Compare the efficiency of the two methods: How many iterations do you need for $|\boldsymbol{n}_k| < \varepsilon = 10^{-10}$ in each case?

(b) Solve the problem of exercise 1 with a modified version of CG that never stores the matrix A of the discretised negative Laplace operator $-\Delta$ but exploits the sparsity of the matrix (cf. lecture 6, remarks about CG):

- Implement a function `multA` that computes the vector $A\boldsymbol{x}$ for a given input vector $\boldsymbol{x}$, i.e., performs the matrix-vector product without setting up the matrix A. There should be no matrix A in memory at any time. The function shall accept one argument (vector $\boldsymbol{x}$).
- Implement a CG version using this function and solve the problem for the same discretization as in exercise 1, i.e., for 79^2 variable grid values, which are again all set to zero for the initial guess. How many CG steps are necessary to reach a desired accuracy of $|\boldsymbol{n}_k| < \varepsilon = 10^{-5}$?
- Visualize the intermediate grid after 10, 50, and 100 CG steps in a 3D plot showing the shape of the covering tissue. Embed the three plots in your PDF or include them in a *single* PNG-file.