Technical University of Munich
School of Computation, Information and Technology
Dr. Michael Obersteiner
Dr. Daniel Lehmberg
Chinmay Datar
Hayden Liu Weng

WT 2022/2023

Lab Course
Scientific Computing

Worksheet 4

distributed: Thu., 22.12.2022
due: Sun., 15.01.2023, 23:59 (submission on the Moodle page)
oral examination: Tue., 17.01.2023 (exact time slots announced on the Moodle page)

In this worksheet, we switch to differential equations for functions with more than one independent variable, for example functions depending on several space dimensions. Such differential equations are called partial differential equations. We start with a first simple example, the two-dimensional stationary heat equation

$$T_{xx} + T_{yy} = -2\pi^2 \sin(\pi x) \sin(\pi y) \tag{1}$$

on the unit square $\Omega =]0, 1[^2$ with the temperature $T(x, y)$, the two-dimensional coordinates x and y, and homogeneous Dirichlet boundary conditions

$$T(x, y) = 0 \ \forall (x, y) \in \partial\Omega. \tag{2}$$

The boundary value problem (1),(2) has the analytical solution

$$T(x, y) = \sin(\pi x) \sin(\pi y). \tag{3}$$

a) The discretization of (1) and (2) leads to a large system of linear equations for the values $T_{i,j}, i \in \{1, \ldots, N_x\}, j \in \{1, \ldots, N_y\}$ denoting the approximate values of the temperature T at the discrete grid points $\left(i \cdot \frac{1}{N_x+1}, j \cdot \frac{1}{N_y+1}\right)$.

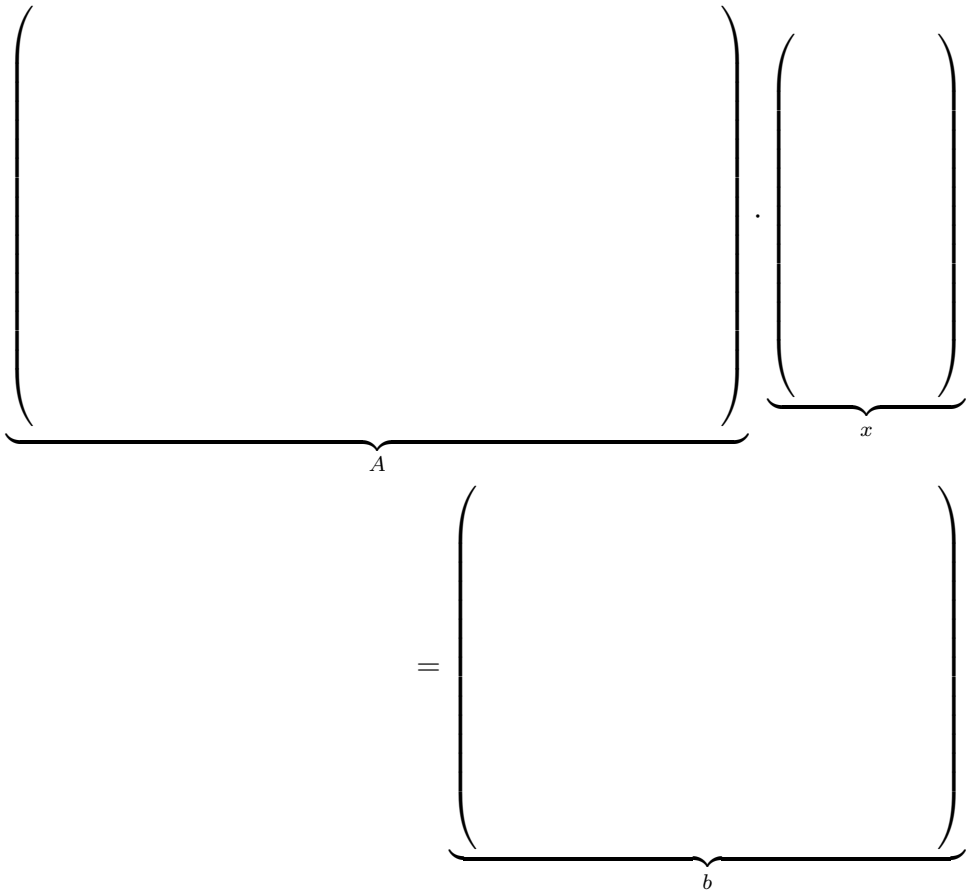
Use the finite difference approximation of the second derivatives

$$T_{xx}|_{i,j} \approx \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{h_x^2},$$

$$T_{yy}|_{i,j} \approx \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{h_y^2}$$

with $h_x = \frac{1}{N_x+1}$ and $h_y = \frac{1}{N_y+1}$.

Sketch a few lines of the resulting system in the following scheme



b) Implement a function creating the matrix from **a)** as a function of N_x and N_y.

c) Since the matrix created in **b)** is mostly composed of zeros, and only a few entries are nonzero, it is advantageous to directly create a *sparse* matrix. Thus, implement

a different function creating the matrix from **a)** as a function of N_x and N_y in sparse format.

Remark: Do NOT(!!!) call `sparse(A)`, as this assumes that you have the full/dense matrix beforehand—and this is exactly what we are trying to avoid. Instead, use your knowledge on the matrix structure to build it up.

d) Implement a Gauss-Seidel solver for the system in **a)** as a function of the right hand side b, N_x, and N_y. The output of the solver is a matrix with dimensions N_x and N_y containing the computed approximate values of T at the grid points $(i \cdot h_x, j \cdot h_y)$ in entry (i, j). As a termination criterion for the iteration use an accuracy limit of 10^{-4} for the residual norm.

Remark 1: Do NOT(!!!) use the explicit system matrices from **b)** or **c)** in your Gauss-Seidel solver. Use your knowledge about the particular, constant form of the lines of the linear system to completely avoid any storage of matrix entries!

Remark 2: Think about the how to handle the boundary points for an efficient traversal of the mesh! Also, remember to only return the inner domain values from your Gauss-Seidel implementation!

Remark 3: The residual norm for a general system $Ax = b$ is defined as

$$R = \sqrt{\frac{1}{N} \sum_k \left(b_k - \sum_m a_{k,m} x_m \right)^2},$$

where N denotes the number of unknowns. In the Gauss-Seidel solver, $N = N_x \cdot N_y$, $a_{i,j}$ are not stored (but known), and the entries of x are the temperature values $T_{i,j}$ at the grid points $(i \cdot h_x, j \cdot h_y)$.

e) Solve the system $Ax = b$ from **a)**

1) storing the system matrix as a "normal" (i.e., full) $N \times N$ matrix and using the MATLAB direct solver,

2) storing the system matrix as a sparse matrix and using the MATLAB direct solver,

3) without storing the system matrix (use Gauss-Seidel with zero as initial guess for T!).

f) Visualize the solutions as a

1) a coloured surface where the temperature represents the height of the surface,

2) a contour plot.

for $N_x = N_y = 3, 7, 15, 31, 63$. Set the range of the temperature values so that the surface doesn't look flat. Think about a reasonable range.

Remark: Also include the boundary values i.e. your plot should contain $(N_x + 2) \cdot (N_y + 2)$ data points.

g) Compare the runtimes and the storage requirements (measured by the number of doubles needed to store the matrices and vectors) for **1)** – **3)** in **a)** and for $N_x = N_y = 7, 15, 31, 63$:

direct solution with full matrix				
N_x, N_y	7	15	31	63
runtime				
storage				

direct solution with sparse matrix				
N_x, N_y	7	15	31	63
runtime				
storage				

iterative solution with Gauss-Seidel				
N_x, N_y	7	15	31	63
runtime				
storage				

Remark 1: Consider in the runtime both the matrix construction (if applicable) and the system solution, as this enables a more fair comparison across all three methods.

Remark 2: If your Gauss-Seidel runtime is unexpectedly high then this is a sign of a poorly implemented solver!

h) Compute the solutions for $N_x = N_y = 7, 15, 31, 63, 127$ with the Gauss-Seidel solver and fill in the resulting errors

$$e = \sqrt{\frac{1}{N_x \cdot N_y} \sum_{j=1}^{N_y} \sum_{i=1}^{N_x} \left(T_{i,j} - T(x_i, y_j)\right)^2}$$

in the following tabular:

$N_x = N_y$	7	15	31	63	127
error					
error red.	—				

Questions:

- Q**1** How many non-zero entries do you achieve in the system matrix from **a)**?
- Q**2** Compare the number of non-zero entries to the number of entries of a full matrix with the same size. What conclusion concerning the methods to be used for the storage of the system matrix can you draw for increasing N_x and N_y?
- Q**3** Which solver would you suggest to use for very big N_x and N_y?
- Q**4** Use the results of **h)** to guess the convergence order of the discretization used for the Laplace operator.
- Q**5** Think about how the observations would change for more general cases (other boundary conditions, arbitrary meshes or geometries, higher dimensions...).