Winfried Auzinger
Gregor Gantner
Alexander Haberl
Dirk Praetorius

Sommersemester 2016
25. März 2016

Übungen zur Vorlesung Computermathematik

Serie 4

Aufgabe 4.1. Extend the code of slide 110 by adding the Δ^2-method from Aufgabe 1 Übung 3 to improve the convergence behavior of the central and forward difference quotients. Which are the observed convergence rates? Visualize them appropriately.

Aufgabe 4.2. Write a MATLAB function `mergesort` and an auxiliary function `merge` (included in the file `mergesort.m`) with the following features:
• Given two row vectors with entries sorted into ascending order $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$, the function `merge` merges them (through a suitable loop) to obtain a row vector $c \in \mathbb{R}^{m+n}$ sorted into ascending order, e.g., for $a = (1, 3, 3, 4, 7)$ and $b = (1, 2, 3, 8)$ the function returns $c = (1, 1, 2, 3, 3, 3, 4, 7, 8)$. The function exploits the fact that the vectors a and b are already sorted, and therefore must not include any sorting algorithm (or the MATLAB function `sort`).
• Given a row vector $c \in \mathbb{R}^N$, the recursive function `mergesort` returns it sorted into ascending order. The algorithm should be implemented as follows: If $N \leq 2$, then c is manually sorted. If $N > 2$, c is halved into two parts, a and b, which are recursively sorted by calling `mergesort` to sort a and b, and `merge` to merge the sorted vectors a and b to build the sorted vector c.

Aufgabe 4.3. Write a MATLAB script, which visualizes the runtime of `mergesort` for random vectors $x \in \mathbb{R}^N$ and $N = 100 \cdot 2^n$ with $n = 0, 1, 2, \ldots$. Devise suitable plots to visualize the computational cost of your implementation. What is your expectation for a vector of length 2^N?

Aufgabe 4.4. The following code computes a sparse matrix $A \in \mathbb{R}^{N \times N}$ (You can download the code from the COMPMATH webpage).

```
function A = matrix(N)

x = rand(1,N);
y = rand(1,N);
triangles = delaunay(x,y);
n = size(triangles,1);

A = sparse(N,N);
for i = 1:n
   nodes = triangles(i,:);
   B = [1 1 1 ; x(nodes) ; y(nodes)];
   grad = B \ [0 0 ; 1 0 ; 0 1];
   A(nodes,nodes) = A(nodes,nodes) + det(B)*grad*grad'/2;
end
```

Plot the computational time $t(N)$ over N and visualize the growth $t(N) = \mathcal{O}(N^\alpha)$ for $N = 100 \cdot 2^k$ and $k = 0, 1, 2, \ldots$. Which growth do you see? What is the reason for it? What is the bottleneck of this implementation? What can be done to improve the runtime behavior? Write an improved code which leads to a better computational time. Visualize its runtime in the same plot to show that the improved code is really superior. Which growth do you expect and see for your improved code? **Hint:** You might want to have a look at `help sparse`.

Aufgabe 4.5. Let $m, n, N \in \mathbb{N}$. Let $I, J, a \in \mathbb{R}^N$ represent the coordinate format of a sparse matrix $A \in \mathbb{R}^{m \times n}$, i.e., for all $k = 1, \ldots, N$ holds $A_{ij} = a_k$ with $i = I_k$, $j = J_k$. Write a MATLAB function

```
[II,JJ,AA] = naive2ccs(I,J,a,m,n)
```

which returns the corresponding vectors of the CCS format.

Aufgabe 4.6. Given the vectors of the CCS format of a sparse matrix $A \in \mathbb{R}^{m \times n}$ from the last exercise, write a MATLAB function

```
Ax = mvm(II,JJ,AA,m,n,x)
```

which computes the matrix-vector multiplication $b = Ax \in \mathbb{R}^m$ for given $x \in \mathbb{R}^n$. The complexity of the code must be $\mathcal{O}(N)$. **Hint:** You can verify your code as follows: Suppose that A is a sparse matrix (e.g., the triadiagonal matrix from page 124 of the lecture notes). Then, the coordinate format of A is obtained by `[I,J,a] = find(A)` in MATLAB. Use your code from Aufgabe 4.5 to compute the vectors of the CCS format and compare the outcome of your function `mvm` with the matrix-vector multiplication `A*x` in MATLAB.

Aufgabe 4.7. Write a function `plotPotential`, which takes a function $f : \mathbb{R}^2 \to \mathbb{R}$, a domain $[a, b]^2$ and a step size $\tau > 0$, and plots the projection of $f(x, y)$ onto the 2D plane (i.e., `view(2)`). Add a `colorbar` to the plot. For the visualization, use a tensor grid with step size τ. You may assume, that the actual implementation of f takes matrices $x, y \in \mathbb{R}^{M \times N}$ and returns a matrix $z \in \mathbb{R}^{M \times N}$ of the corresponding function values, i.e., $z_{jk} = f(x_{jk}, y_{jk})$. Optionally, the function `plotPotential` takes a parameter $n \in \mathbb{N}$. For given n, add n (black or white) contour lines to the figure. To verify your code, write a MATLAB script which visualizes the potential $f(x, y) = x \cdot \exp(-x^2 - y^2)$ from the lecture notes.

Aufgabe 4.8. Write a MATLAB function `saveMatrix` which takes a matrix $A \in \mathbb{R}^{M \times N}$ and writes it into an ASCII file `matrix.dat` via `fprintf` (see also `help fopen`). Use `%1.16e` for `fprintf` to write the matrix coefficients! (Why does this make sense?) Optionally, the function takes a string `name` and writes the matrix to the ASCII file `name.dat`. To verify your code, write a MATLAB script which creates a random matrix $A \in \mathbb{R}^{M \times N}$ and writes it to an ASCII file `A.dat`. Load the matrix via `B = load('A.dat')` and check whether A and B coincide.