Dirk Praetorius
Winfried Auzinger

Sommersemester 2018
11.04.2018

Übungen zur Vorlesung
Computermathematik

Serie 4

Aufgabe 4.1. Let $I := \int_0^5 \exp(x)\,dx$. Use the quadrature from exercise 2.7, to compute a sequence of approximations $I_N := \sum_{j=1}^{N} \omega_j f(x_j)$. Here, N is the number of equidistant nodes $0 = x_1 < x_2 < \ldots < x_N = 5$. Further, visualize the error $E_N := |I - I_N|$ as well as the error-estimator $\delta_N := |I_{2N} - I_N|$ in a double-logarithmic plot. What is the convergence behavior of the sequences E_N and δ_N? Use the Aitken-Δ^2-algorithm, does the order of convergence increase?

Aufgabe 4.2. Implement the *Quicksort* algorithm, which sorts a vector $x \in \mathbb{R}^n$. To do so *Quicksort* chooses an arbitrary Pivot element from x, e.g., x_1. Then, x is splitted into two parts, $x^{(<)}$ and $x^{(\geq)}$, and the Pivot element x_1: $x^{(<)}$ contains all the elements $\leq x_1$, while $x^{(\geq)}$ contains all the elements $\geq x_1$. $x^{(<)}$ and $x^{(\geq)}$ are sorted recursively. Afterwards, the result is put together. The implementation of this algorithm in MATLAB (in contrast to C/C++), requires additional storage. Why?

Aufgabe 4.3. Write a MATLAB script, which measures and visualizes the runtime of quicksort from exercise 4.2. First consider a sequence of random vectors $x \in \mathbb{R}^N$ and $N = 100 \cdot 2^n$ with $n = 0, 1, 2, \ldots$. Further construct a sequence of vectors which realize the worst-case complexity of your implementation. Compare the two different commands `tic-toc` and `cputime`. Devise suitable plots to visualize the computational cost of your implementation. What is your expectation for the computational complexity? Do the experiments underpin your theoretical findings?

Aufgabe 4.4. Write a function `plotPotential`, which takes a function $f : [a, b]^2 \to \mathbb{R}$, a interval $[a, b]$ 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 4.5. Suppose you are given a C function with signature

```
double f(double x, double y);
```

Write a MEX-MATLAB function `fct` which takes matrices $X, Y \in \mathbb{R}^{M\times N}$ and returns the matrix

```
Z = fct(X,Y)
```

with $Z \in \mathbb{R}^{M\times N}$ and $Z_{jk} = f(X_{jk}, Y_{jk})$. The MEX function should check whether the dimensions of X and Y coincide, and should throw an error if they do not. To check your implementation, implement the function $f(x, y) = x\cos(y)\exp(-x^2) + \exp(-y^2)\sin(x^2)$ in C and reproduce the plots of the lecture slides 134-140.

Aufgabe 4.6. 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 4.7. 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 126 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 4.6 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 4.8. 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,:);
   c1 = [x(triangles(i,1)),y(triangles(i,1))] ;
   d21 = [x(triangles(i,2)),y(triangles(i,2))]  - c1;
   d31 = [x(triangles(i,3)),y(triangles(i,3))]  - c1;

   area = 0.5*(d21(:,1).*d31(:,2)-d21(:,2).*d31(:,1));
   B = [1/12,1/24,1/24;1/24,1/12,1/24;1/24,1/24,1/12] * area;
   A(nodes,nodes) = A(nodes,nodes) + B;
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`.