Dirk Praetorius
Winfried Auzinger

Sommersemester 2017
29.03.2017

Übungen zur Vorlesung
Computermathematik

Serie 4

Aufgabe 4.1. Implementieren Sie den *Quicksort*-Algorithmus, um einen Vektor $x \in \mathbb{R}^n$ zu sortieren: *Quicksort* wählt willkürlich ein Pivotelement aus der zu sortierenden Liste x, z.B. x_1. Dann zerlegt man die Liste in zwei Teillisten $x^{(<)}$ und $x^{(\geq)}$ und das Pivotelement x_1: $x^{(<)}$ enthält dabei alle Elemente $< x_1$, $x^{(\geq)}$ enthält nur Elemente $\geq x_1$. $x^{(<)}$ und $x^{(\geq)}$ werden rekursiv sortiert. Anschließend wird das Ergebnis zusammengesetzt. Eine Implementierung dieses Algorithmus in MATLAB hat (im Gegensatz zu C/C++) allerdings den Nachteil, dass zusätzlicher Speicher benötigt wird. Warum?

Aufgabe 4.2. Schreiben Sie ein Skript, das die Laufzeiten von Quicksort aus Aufgabe 4.1 bestimmt und geeignet visualisiert. Testen Sie zuerst eine Folge von zufälligen Vektoren $x \in \mathbb{R}^N$ und $N = 100 \cdot 2^n$ mit $n = 0, 1, 2, \ldots$. Konstruieren Sie zusätzlich eine Folge von Vektoren, welche den worst-case realisiert. Vergleichen Sie in dabei auch `tic-toc` mit `cputime`. Zeichnen Sie geeignete Vergleichsgeraden in den Plot, um visuell den Aufwand Ihrer Implementierung zu bestimmen. Welchen Aufwand erwarten Sie theoretisch?

Aufgabe 4.3. Betrachten Sie das Integral $I := \int_0^5 \exp(x)\,dx$. Verwenden Sie die zusammengesetzte Mittelpunktsregel aus Aufgabe 3.7, um die Folge der approximativen Integrale I_N zu berechnen. Verwenden Sie einen doppelt-logarithmischen Plot, um den Fehler $E_N = |I - I_N|$ und den Fehlerschätzer $\delta_N = |I_{2N} - I_N|$ zu visualisieren. Verifizieren Sie das Konvergenzverhalten $\mathcal{O}(N^{-2})$. Welches Konvergenzverhalten beobachten Sie, wenn Sie das Δ^2-Verfahren von Aitken verwenden? Welches Konvergenzverhalten beobachten Sie, wenn Sie die Auswertung $f\big((x_{j-1} + x_j)/2\big)$ beim Mittelpunkt durch die Auswertung $f(x_{j-1})$ ersetzen?

Aufgabe 4.4. Schreiben Sie eine Funktion `plotPotential`, welche die Funktion $f : [a, b]^2 \to \mathbb{R}$, das Intervall $[a, b]$ und eine Schrittweite $\tau > 0$ nimmt und die Projektion $f(x, y)$ auf die Ebene (d.h. `view(2)`) plottet. Fügen Sie dem Plot eine `colorbar` hinzu. Für die Visualisierung, verwenden Sie ein Tensorgitter der Schrittweite τ. Sie dürfen annehmen, dass f so implementiert ist, dass es Matrizen $x, y \in \mathbb{R}^{M \times N}$ als Eingabe nimmt und eine Matrix $z \in \mathbb{R}^{M \times N}$ der zugehörigen Funktionswerte zurückgibt, also $z_{jk} = f(x_{jk}, y_{jk})$. Optional, habe die Funktion den Eingabeparameter $n \in \mathbb{N}$. Zu gegebenem n, fügen Sie der Figure n (schwarze oder weiße) Kontourlinien hinzu. Zur Verifizierung Ihres Codes, Schreiben Sie ein MATLAB Skript, welches das Potential $f(x, y) = x \cdot \exp(-x^2 - y^2)$ aus den Vorlesungsunterlagen visualisiert.

Aufgabe 4.5. Seien $m, n, N \in \mathbb{N}$. Seien $I, J, a \in \mathbb{R}^N$ die Repräsentanten des Koordinatenformates einer sparse Matrix $A \in \mathbb{R}^{m \times n}$, d.h., für alle $k = 1, \ldots, N$ gilt $A_{ij} = a_k$ mit $i = I_k$, $j = J_k$. Schreiben Sie eine MATLAB Funktion

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

welche die zugehörigen Vektoren des CCS Formats zurückgibt.

Aufgabe 4.6. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und Vektoren $I, J, Aij \in \mathbb{R}^N$, die eine schwachbesetzte Matrix in naiver Speicherung (Koordinatenformat) repräsentieren. Schreiben Sie eine Funktion

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

die als Ergebnis die entsprechenden Vektoren des CCS-Formats zurückliefert. Schreiben Sie ferner eine Matrix-Vektor-Multiplikation

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

die den Produkt-Vektor $Ax \in \mathbb{R}^m$ für eine im CCS-Format gespeicherte Matrix $A \in \mathbb{R}^{m \times n}$ und einen Spaltenvektor $x \in \mathbb{R}^n$ zurückliefert. Sie können Ihre Funktion testen, indem Sie für eine `sparse`-Matrix A mittels `[I,J,Aij] = find(A)` das Koordinatenformat auslesen und das Ergebnis von `A*x` mit Ihrer eigenen Matrix-Vektor-Multiplikation vergleichen.

Aufgabe 4.7. Der folgende Code berechnet die sparse Matrix $A \in \mathbb{R}^{N \times N}$ (Der Code kann auf der COMPMATH Webseite heruntergeladen werden).

```
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
```

Plotten Sie die Rechenzeit $t(N) = \mathcal{O}(N^\alpha)$ über N für $N = 100 \cdot 2^k$ und $k = 0, 1, 2, \ldots$. Welchen Aufwand beobachten Sie? Woran liegt das? Was kann getan werden, um das Laufzeitverhalten zu verbessern? Schreiben Sie einen verbesserten Code, der zu einer besseren Rechenzeit führt. Visualisieren Sie die Laufzeit in dem gleichen Plot, um zu zeigen, dass der neue Code tatsächlich schneller ist. Welchen Aufwand erwarten und beobachten Sie für Ihren verbesserten Code? **Hinweis:** Verwenden Sie `help sparse`.

Aufgabe 4.8. Nehmen Sie an, Sie haben eine C Funktion mit Signatur

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

gegeben. Schreiben Sie eine MEX-MATLAB Funktion `fct`, welche die Matrizen $X, Y \in \mathbb{R}^{M \times N}$ nimmt und die Matrix

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

mit $Z \in \mathbb{R}^{M \times N}$, $Z_{jk} = f(X_{jk}, Y_{jk})$ zurückgibt. Die MEX Funktion soll überprüfen, ob die Dimensionen von X und Y übereinstimmen und eine Fehlermeldung ausgeben, falls nicht. Um Ihre Implementierung zu verifizieren, implementieren Sie die Funktion $f(x, y) = x \cos(y) \exp(-x^2) + \exp(-y^2) \sin(x^2)$ in C und reproduzieren Sie die Plots der Folien 134-140 aus letzten Vorlesung.