Einführung in die Numerik
3. Programmierübung

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Mathematik **WS 2015/16**
Prof. Dr. Herbert Egger, Dr. Mirjam Walloth, Thomas Kugler **11.1.2015**

Programmierübung

Aufgabe P1
gauss.m: Schreiben Sie eine Routine
`[R,y]=gauss(A,b),`
welche in Abhängigkeit der Eingabe folgende Aktionen durchführt:

(a) Falls `size(b,2)>=1` ist, ist R und y das Resultat der Gauß-Elimination mit Pivotisierung für das System $Ax = b$. Insbesondere sollen auch mehrere rechte Seiten (Spalten von b) zugelassen sein.

(b) Falls `size(b,2)==0` ist, gilt $y = (-1)^l$ wobei l die Anzahl der Zeilenvertauschungen ist.

Modifizieren Sie hierzu den Gauß-Algorithmus 7.8 aus dem Skriptum geeignet. Bestimmen Sie die Komplexität in Abhängigkeit der Anzahl der Spalten von b.
testp1.m: Testen Sie Ihre Implementierung mit den Eingaben

(a) $A = [2, 1; 1, 2]$ und $b = [3, 4; 3, 1]$.

(b) $A = [0, 1; 1, 1]$ und $b = [1 : 10; 2 : 11]$.

(c) $A = -\text{ones}(10) + 11 * \text{eye}(10)$ und $b = [1 : 10]'$.

Wiederholen Sie die Tests mit $b = [\,]$ und erklären Sie Ihre Ergebnisse.

Aufgabe P2
backward.m: Implementieren Sie einen Algorithmus
`function x=backward(R,y),`
welcher die Lösung x zu $Rx = y$ mittels Rückwärtseinsetzen berechnet. Die rechte Seite y darf auch mehrere Spalten haben. Modifizieren Sie hierzu den Algorithmus 7.11 aus dem Skriptum geeignet. Bestimmen Sie die Komplexität in Abhängigkeit der Anzahl der Spalten in y.
testp2.m: Testen Sie Ihre Implementierung mit den Resultaten der Tests aus Aufgabe P1.

Aufgabe P3
solve.m, inverse.m, determinant.m: Schreiben Sie Routinen
```
function x=solve(A,b)
function B=inverse(A)
function d=determinant(A)
```
welche unter Verwendung der Funktionen aus obigen Aufgaben die entsprechenden Resultate liefert. Im ersten Fall darf b wieder mehrere Spalten besitzen.
testp3.m: Lösen Sie mit Hilfe der Funktion `solve` die Gleichungssysteme aus der ersten Aufgabe. Berechnen Sie darüber hinaus jeweils die Inverse sowie die Determinante der Systemmatrix A. Vergleichen Sie die Ergebnisse mit den Resultaten der entsprechenden MATLAB Routinen $A\backslash b$, inv(A) und det(A).

Aufgabe P4

cholesky.m: Implementieren Sie die Cholesky-Zerlegung

```
function [L,D]=cholesky(A)
```

zur Berechnung der Zerlegung $A = LDL^\top$ eine symmetrischen positiv-definiten Matrix A. Modifizieren Sie hierzu den Algorithmus 7.25 aus dem Skriptum gemäß der Bemerkung 7.26 im Skript (aktuelle Version!). Überprüfen Sie die Komplexität Ihres Algorithmus!

testp4.m: Berechnen Sie die Cholesky-Zerlegung zur ersten und dritten Matrix aus Aufgabe P1 und überprüfen Sie, ob $LDL^\top = A$ gilt.

Aufgabe P5

forward.m: Schreiben Sie eine Routine

```
function y=forward(L,b)
```

zum Vorwärtseinsetzen, d.h., zum Lösen des Systems $Ly = b$ mit linker oberer Dreiecksmatrix L. Dabei darf b auch mehrere Spalten besitzen. Bestimmen Sie die Komplexität in Abhängigkeit der Größe von b.

solveCholesky.m: Kombinieren Sie die Routinen aus den vorhergehenden Aufgaben geeignet, um mit Hilfe von

```
x=solveCholesky(A,b)
```

die Lösung zu $Ax = b$ mit symmetrisch positiv definitem A zu berechnen. Die rechte Seite b darf wieder mehrere Spalten besitzen.

testp5: Testen Sie das Vorwärtseinsetzen anhand der Matrizen L welche als Resultate der vorigen Aufgabe erhalten wurden und der zugehörigen rechten Seiten b aus Aufgabe P1. Bestimmen Sie dann mit Hilfe der Choleksyzerlegung die Lösungen zum ersten und dritten Gleichungssystem aus Aufgabe P1. Vergleichen Sie mit den Resultaten aus Aufgabe P3.

Organisatorische Hinweise:

a) Legen Sie ein Verzeichnis `pp3` an und speichern Sie alle Funktionen und Skripte für diese Übung darin ab. Der Inhalt des Verzeichnisses sollte hier also sein:

```
testp1.m gauss.m
testp2.m backward.m
testp3.m solve.m inverse.m determinant.m
testp4.m cholesky.m
testp5.m forward.m  solveCholesky.m
```

b) Zippen Sie das Verzeichnis; in Linux: `zip -r pp3.zip pp3` von außerhalb des Verzeichnisses. In Windows kann WinZip oder ein ähnliches Tool verwendet werden.

c) Überprüfen Sie, dass alle Skripte und Funktionen lauffähig sind!

d) Kommentieren Sie die Funktionen geeignet. Insbesondere sollten Eingabe und Ausgabe Parameter beschrieben werden. `help <funktionsname>` sollte Information über die Routinen liefern.

e) Antworten auf die Fragen sollten in den `testp*.m` files als Kommentare hinterlegt werden.