Institut für Angewandte und Numerische Mathematik
Prof. Dr. Christian Wieners, Dipl.-Math. techn. Daniel Maurer

Numerische Mathematik 1 — Programmierpraktikum
Übungsblatt 1 vom 22. Oktober 2012
Wintersemester 2012/13

Programmieraufgabe 1 (Einführung)
Nachfolgend finden Sie die wichtigsten Befehle, die Sie für Matlab brauchen. Sie können sich alle Funktionsbeschreibungen auch in Matlab erläutern lassen, indem Sie `help Funktionsname` eingeben.

- **Höhere mathematische Funktionen:** `sin(t), 5^3, log(1)`
- **Vordefinierte Konstanten:** `pi, i, eps`
- **Matrizen:** Matrizen und Vektoren können wir folgt definiert werden:

```
x = [1 2 3]; x = [1,2,3];
y = [1;2;3];
A = zeros(3,4);
B = ones(2,3);
C = diag([1 2 3]);
```

Dies ergibt
$\mathtt{x} = \begin{bmatrix}1 & 2 & 3\end{bmatrix}, \mathtt{y} = \begin{bmatrix}1\\2\\3\end{bmatrix}, \mathtt{A} = \begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}, \mathtt{B} = \begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\end{bmatrix}, \mathtt{C} = \begin{bmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 3\end{bmatrix}$ Entsprechend `x,y` können auch Matrizen definiert werden, wobei `,` Zeileneinträge trennt und `;` Spalteneinträge. Mit `'` können Matrizen transponiert werden und mit Hilfe von `size(A)` können wir die Dimension von `A` erfahren. Mit dem `diag`-Befehl ist es auch möglich, Nebendiagonalen zu definieren (zum Beispiel `diag([1 2 3],-1)`). Durch `eye(n,m)` kann eine $n \times m$-"Einheitsmatrix" definiert werden.

- **Komponentenweise Operationen:** Ein Punkt vor `+,-,*,/,^` sorgt dafür, dass die entsprechenden Operationen komponentenweise durchgeführt werden. So ergibt `x.*x` beispielsweise $\begin{bmatrix}1 & 4 & 9\end{bmatrix}$. Dies ist nützlich, um Funktionen auf Vektoren anwenden zu können und nicht nur auf einzelne Werte.
- **Vektorgenerierung** Mit

```
x = startwert: inkrement: endwert;
y = linspace(startwert,endwert,AnzahlWerte);
```

können einfach und schnell Vektoren mit äquidistanten Werten erstellt werden.

- **2d-Plot:** Um zu Plotten, wird ein Vektor `x` mit den Stützstellen und ein Vektor `fx` mit den entsprechenden Funktionsauswertungen benötigt. Der Funktionsgraph wird dann mit `plot(x,fx)` ausgegeben. Mit Hilfe des Befehls `hold on` können mehrere Plots in einem Fenster nacheinander gezeichnet werden. Unter anderem kann zusätzlich die Linenart und -farbe bestimmt werden.
- **Funktionen schreiben:** Um eigene Funktionen zu schreiben, erstellen Sie ein sogenanntes M-File. Funktionen beginnen immer mit `function [a,b,...] = functionname(x,y,...)`. Dabei sind `a,b,...` die Ausgabedaten und `x,y,...` die Eingabedaten. Im nachfolgenden Funktionsverlauf müssen dann die Ausgabedaten gesetzt werden. Mit Hilfe des Prozentzeichens `%` kann der Code kommentiert werden. Kontrollstrukturen wie `if x~=0` oder `for i=1:n` funktionieren wie gewohnt, müssen aber durch `end` abgeschlossen werden. Ferner muss der Funktionsname und der Name der Datei übereinstimmen.

Die eigentliche Aufgabe lautet: Spielen Sie ein wenig mit den Funktionen herum, bis Sie alle verstanden haben! Nutzen Sie dabei auch den `help`-Befehl.

Programmieraufgabe 2 (Erstellen von Matrizen)

(a) Verwenden Sie ausschließlich die Funktionen `zeros` und `ones` sowie die Operationen `+`, `-`, `.*`, `,` und `;`, um folgende Matrizen und Vektoren zu erzeugen:

$$u = \begin{pmatrix}1\\1\\-1\\-1\end{pmatrix}, A = \begin{pmatrix}5 & 5 & 5 & 5 & 5\\5 & 5 & 5 & 5 & 5\\0 & 0 & 0 & 0 & 0\end{pmatrix}, v = \begin{pmatrix}0 & 3 & 3 & 3 & 0\end{pmatrix}, B = \begin{pmatrix}1 & 1 & 2 & 2\\1 & 1 & 2 & 2\\3 & 3 & 4 & 4\\3 & 3 & 4 & 4\end{pmatrix}.$$

(b) Verwenden Sie ausschließlich die Funktionen `ones`, `diag` und `eye`, den Doppelpunktoperator `:` sowie die Operationen `+`, `-` und `.*`, um folgende Matrizen und Vektoren zu erzeugen:

$$A = \begin{pmatrix}5 & 0 & 0 & 0\\0 & 5 & 0 & 0\\0 & 0 & 5 & 0\end{pmatrix}, B = \begin{pmatrix}0 & 1 & 0 & 0\\-1 & 0 & 1 & 0\\0 & -1 & 0 & 0\\0 & 0 & -1 & 0\end{pmatrix}, C = \begin{pmatrix}1 & 3 & 5 & 7\\0 & 1 & 3 & 5\\0 & 0 & 1 & 3\\0 & 0 & 0 & 1\end{pmatrix}.$$

(c) Verwenden Sie ausschließlich die Funktion `diag` sowie den Doppelpunktoperator `:` und den Transpositionsoperator `'`, um folgende Matrizen und Vektoren zu erzeugen:

$$w = \begin{pmatrix}4 & 5 & 6 & 7\end{pmatrix}, x = \begin{pmatrix}0\\2\\4\\6\\8\end{pmatrix}, y = \begin{pmatrix}4\\3\\2\\1\end{pmatrix}, C = \begin{pmatrix}1 & 0 & 0 & 0\\0 & 2 & 0 & 0\\0 & 0 & 3 & 0\\0 & 0 & 0 & 4\end{pmatrix}.$$

Programmieraufgabe 3 (Plotten von Funktionen)
Verwenden Sie die Funktionen `linspace` und `plot`, um den Graph der Funktion

$$f(x) = \cos(20x)$$

im Intervall $[0, 2\pi]$ zu plotten.

Programmieraufgabe 4 (LU-Zerlegung)
Unter `/misc/maurer/Programm1/` an den Rechnern im Rechnerraum oder auf der Vorlesungshomepage finden Sie zwei Arten einer LU-Zerlegung einer Matrix A. Beide berechnen die LU-Zerlegung wie in der Vorlesung, so dass die Matrix A mit dieser Zerlegung überschrieben wird. Es besteht folgender Unterschied:

- `LU_einzeln(A)`: Hier wird auf jedes Element einzeln zugegriffen
- `LU_vektor(A)`: Die zwei inneren `for`-Schleifen wurden durch Zugriffe auf Untermatrizen von A ersetzt.

In `MATLAB` existiert weiterhin eine Funktion `[L,U,P] = lu(A)`, die eine LU-Zerlegung mit Permutation berechnet.

(a) Konstruieren Sie mit Hilfe von `A = rand(N)` für $N = 100, 300, 500, 1000$ Zufallsmatrizen und wenden Sie die oben genannten Funktionen darauf an. Messen Sie mit Hilfe von `tic;Funktion;toc;` die benötigte Rechenzeit und plotten Sie diese in ein Diagramm.

(b) Erstellen Sie eine Hilbertmatrix $A \in \mathbb{R}^{N\times N}$ mit Einträgen $A(i,j) = \frac{1}{i+j-1}$ (dazu dürfen Sie den Befehl `A=hilb(N)` benutzen). Untersuchen Sie für $N = 3, 5, 10, 15, 20$ folgende Eigenschaften:

- Determinante von A über `det(A)`.
- Die Kondition von A über `cond(A)`.
- Erstellen Sie die rechte Seite `b` mit $b(i) = \sum_{n=1}^{N} A_{in}$. Lösen Sie mit Hilfe von folgenden Befehlen das Gleichungssystem
  - `inv(A)*b`: Hier wird zuerst die Inverse von `A` berechnet.
  - `A\b`: Die Berechnung geschieht direkt über Gauss.
  - `linsolve(A,b)`: Hier wird die LU-Zerlegung mit Pivotisierung angewandt.
- Mit `invhilb(N)` erhalten Sie die "exakte" Inverse für $N \leq 15$. Untersuchen Sie die Differenz von `invhilb(N)-inv(hilb(N))`.

Diskutieren Sie die Ergebnisse!

Abgabe der Programmieraufgaben:

Die bearbeiteten Programmieraufgaben können Sie mittwochs von 15:00 - 17:00 im Rechnerpool Geb. 01.93 (Kronenstr. 32, Raum 101) vorführen und erläutern. Dort haben Sie auch die Möglichkeit, unter Hilfestellung zu programmieren. Die Abgabe dieses Programmierblattes ist bis spätestens **Mittwoch, den 07. November 2012** möglich.

Service/Material:

Unter http://www.math.kit.edu/ianm3/lehre/numa12012w/ finden Sie die Homepage zur Vorlesung. Dort finden Sie neben den aktuellen Übungsblättern auch aktuelle Informationen zum Vorlesungsbetrieb.

Sprechstunden:

Prof. Dr. Christian Wieners: Dienstag, 09.30-10.30 Uhr und nach Vereinbarung
Dipl.-Math. techn. Daniel Maurer: Mittwoch, 14.30-15.30 Uhr und nach Vereinbarung