Dr. Michael Lehn
Iris Häcker
Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WS 2013/2014

Aufgaben 

1. *(Starten von Matlab im Pool Helmholtzstr. 18, E.44)*

   - Öffne unter *Activities/Applications* ein Terminal.
   - Gib `matlab` ein.

2. *(Einfache Eingabe von Matrizen und Vektoren)*

   Gib folgende Zeilen in Matlab ein:

   ```
   A = [ 1, 2, 3; 4, 5, 6; 7, 8, 1]
   B = [ 3 1 6; 9 1 1]
   x = [ 1, 2, 3]
   y = [ 1; 2; 3]
   z = [ 1 2 3]'
   z = [ 1 2 3].'
   ```

   Beachte:

   - Am Ende jeder Zeile steht kein Strichpunkt. Deshalb wird nach jeder Zeile von Matlab die neu angelegte Matrix bzw. der Vektor angezeigt.
   - Wichtig ist, dass man zwischen Kommas und Strichpunkten unterscheidet. Mit einem Strichpunkt beginnt man eine neue Zeile in der Matrix.
   - Statt Kommas kann man auch Leerzeichen benutzen um Zahlen zu trennen.
   - Mit dem Apostroph `'` oder Punkt-Apostroph `.'` wird eine Matrix oder Vektor transponiert. Sind die Werte komplex, so werden bei `'` die Einträge zusätzlich konjugiert. Sind die Einträge reell, so sind beide Operationen äquivalent.

3. *(Einfaches Rechnen mit Matrizen und Vektoren)*

   Wieder einfach abtippen, ausprobieren und rumspielen:

   ```
   A+A   2*A   A+B   A*x   x*A   x'*A
   x*y   x'*y  x*y'  B*B'  B'*B  x*(y'+z')'
   ```

   In welchen Fällen wird eine Fehlermeldung ausgegeben? Und wieso?

4. *(Umgang mit Matrix Ausschnitten)*

   Was passiert, wenn man folgendes eingibt:

   (a) `A(1:3,2)=x`

   (b) `A(2,1:3)=0`

   (c) `A(1:2,:) = B`

   (d) `B + y*x(2:3)`

   (e) `B' + y*x(2:3)`

5. *(Erster Umgang mit der Matlab Hilfe)*

   (a) Gibt man `size(B)` ein, so gibt Matlab sowohl die Anzahl der Zeilen als auch die Anzahl der Spalten aus. Finde mit der Matlab-Hilfe heraus, wie jeweils nur die Anzahl der Zeilen bzw. nur die Anzahl der Spalten ermittelt werden kann!

   (b) Sei $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{pmatrix}$. Berechne die Inverse A^{-1} mit Matlab.

   (c) Löse das lineare Gleichungssystem $Ax = b$ mit $b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ mittels ...

      i. ... der Inversen von A.

      ii. ... dem Backslash-Operator.

6. *(Die While-Loop in Matlab)*

   Mit einer sogenannten *While-Loop* kann man bestimmte Anweisungen wiederholt ausführen und zwar solange wie eine bestimmte Bedingung erfüllt ist. Das hört sich abstrakt an, sollte aber an einem einfachen Beispiel klar werden:

   Eine Zahl x soll solange halbiert werden, bis sie kleiner gleich 1 ist. Außerdem soll der Faktor bestimmt werden, durch den das ursprüngliche x letztlich geteilt wurde. Für $x = 5,678$ erreicht man das so:

```
faktor = 1;
x = 5.678;
while x>1
       x = x / 2;
       faktor = faktor * 2;
end
```

   Tippe diese Zeilen ein und bestimme, welche Werte die Variablen `x` und `faktor` danach haben.

   Ändere nun obige Zeilen so ab, dass folgendes Problem gelöst wird: Eine Zahl $x = 1.234$ soll so oft verdoppelt werden, bis $\frac{1}{x}$ kleiner als $0,001$ ist. Mit welchem Faktor wurde x letztlich multipliziert?

7. *(Funktionen in Matlab)*

   (a) Um euch einen einfachen Einstieg in das Programmieren von *Matlab Funktionen* zu ermöglichen, wird euch die erste Funktion vorgegeben:

```
function [u, v] = obscure(A, x)
   B = A + x*x';
   u = B*x;
   v = u'*A*x;
end
```

   Erstellt mit dem Matlab-Editor eine neue Datei namens `obscure.m` und tippt dort diese Zeilen ein. Wenn alles geklappt hat, kann man dann im Matlab Komandofenster diese neue Funktion benutzen, z.B.:

```
A = [1, 2; 3, 4];
x = [6; 7];
[u, v] = obscure(A, b)
```

   (b) Nach der ersten Erfahrung mit Funktionen seid ihr nun in der Lage, eine eigene Funktion `meineFunktion` zu programmieren. Diese soll als Eingabe eine Zahl `x` erhalten und dann die `while` Schleife aus Aufgabe 6 ausführen. Die Funktion soll dann `faktor` und das veränderte `x` zurückgeben. Diese Funktion kann dann so benutzt werden:

```
[x, faktor] = meineFunktion(5.678)
```

8. *(Noch eine Schleife: For-Loop)*

   Bei einer For-Loop werden bestimmte Anweisungen so oft ausgeführt, bis eine sogenannte *Zählvariable* einen vorgegebenen Bereich durchlaufen hat. Dieses Konzept ist bekannt vom Summenzeichen $\sum_{k=m}^{n} \ldots$.

   (a) Welchen Wert hat `sum` nachdem man folgendes eingibt:

```
sum = 0;
for i=1:10
    sum = sum + i;
end
```

   (b) Welchen Wert hat `sum` nachdem man folgendes eingibt:

```
sum = 0;
for i=1:2:10
    sum = sum + i;
end
```

   (c) Berechne mit Hilfe einer For-Loop die Summe $\sum_{k=1}^{10} k^2$

   (d) Schreibe eine Funktion `mysum.m`, die $\sum_{k=1}^{n} k^2$ berechnet und wie folgt benutzt werden kann:

```
erg = mysum(21)
```

9. *(Bedingte Anweisungen: If-Else)*

Mit dem *If-Else* Konstrukt kann eine Anweisung abhängig von einer Bedingung ausgeführt werden oder nicht:

```
function y = obscure2(x)
   y = 1;

   if (x>0)
      y = y*x;          % Anweisung (1)
   end

   if (y>1)
      y = -y;           % Anweisung (2)
   else
      y = 1 - y;        % Anweisung (3)
   end
end
```

Ist es möglich, Werte für `x` so zu wählen, dass beim Aufruf von `obscure2(x)` genau die Anweisungen

(a) (1) und (2),

(b) (1) und (3),

(c) nur (2),

(d) nur (3)

ausgeführt werden?