Prof. Dr. Karsten Urban
Iris Häcker

SoSe 2013
Blatt 3
ab 27.05.2013

Wima Praktikum 1

Praktikumsblatt 3

1. **Function Handles**

   a) Lösen Sie die Aufgabe 5 (a) von Blatt 1 mit `function handles`. Ihr `function handle` soll dabei in der Lage sein, Vektoren/ Matrizen als Input zu erhalten und zu verarbeiten.

   b) Gegeben sei $f : \mathbb{R}^2 \to \mathbb{R}$ mit

   $$f(x, y) = \sin(x)\cos(y)$$

   Berechnen Sie die Funktionswerte für $x \in [-3, 3]$ und $y \in [-3, 3]$ unter Verwendung von `function handles`. Ihr `function handle` soll dabei in der Lage sein, Vektoren/ Matrizen als Input zu erhalten und zu verarbeiten.
   Hinweis: Machen Sie sich mit der Funktion `meshgrid` vertraut.
   Zusatz: Stellen Sie die Funktion als ein 3D-Plot dar. Benutzen Sie dafür das Meshgrid und plotten Sie die Daten mittels `surf`.

2. **Matlab-Programmierung**

   a) Gegeben sei folgende stückweise konstante Funktion

   $$f(t) = \begin{cases} 0, & \text{wenn } t \in [0, 1] \\ 1, & \text{wenn } t \in (1, 2] \\ 2, & \text{wenn } t \in (2, 3] \end{cases}$$

   Wir wollen ein Skript schreiben, welches diese Funktion auswertet.

   i. Lösen Sie dies zunächst wie in Java oder C (d.h. das Skript, soll wie ein Java- oder C-Programm arbeiten).

   ii. Lösen Sie dies nun, indem Sie einen Matlab-typischen Code schreiben.
   Hinweis: Vermeiden von Schleifen; Logische Vektoren; `find`

   b) Gegeben sei folgender Code:

Listing 1: Vermeiden von for-Schleifen

```
tic
for i = 1:1000
  for j = 1:1000
    r(i,j) = sqrt(i^2+j^2);
  end
end
toc
```

Schreiben Sie diesen Code ohne `for`-Schleife und vergleichen Sie mittels `tic-toc` die Laufzeit.
Hinweis: Benutzen Sie `meshgrid` und nutzen Sie aus, dass die Funktion `sqrt` mit Matrizen arbeiten kann.

c) Gegeben sei folgender Code:

Listing 2: Vermeiden von for-Schleifen

```
function A = myFuncMinMax(myMatrix, myMinValue, myMaxValue)
  [m, n] = size(myMatrix);
  for i = 1:m
    for j = 1:n
      c = myMatrix(i,j);
      if ( c > myMaxValue )
        myMatrix(i,j) = myMaxValue;
      elseif ( c < myMinValue )
        myMatrix(i,j) = myMinValue;
      end
    end
  end
  A = myMatrix;
end
```

Die Funktion werde wie folgt aufgerufen:

Listing 3: Vermeiden von for-Schleifen

```
clear all

myMatrix = rand(1234,9876)*100;
myMaxValue = 89.765;
myMinValue = 34.12;

tic
A = myFuncMinMax(myMatrix,myMinValue,myMaxValue);
toc
```

Was macht die Funktion `myFuncMinMax`? Schreiben Sie eine neue Funktion

`myFuncMinMax2(myMatrix,myMinValue,myMaxValue)`,

welche dasselbe Ergebnis liefert wie die obige Funktion, jedoch ohne `for`-Schleifen. Vergleichen Sie die Laufzeit beider Funktionen mittels `tic` und `toc`.
Hinweis: Benutzen Sie die Funktionen `min` und `max`. Der Code ist ein 1-Zeiler.

3. **Lagrange-Interpolation**
Wir wollen eine Funktion interpolieren, d.h. zu gegebenen x_i- und y_i-Werten wollen wir eine Funktion P konstruieren, welche diese Punkte exakt trifft, also $P(x_i) = y_i$. Bei der Lagrange-Interpolation wählt man die Funktion P als ein Polynom.

- Gegeben:

$$x_0 < x_1 < \cdots < x_n \in \mathbb{R}$$
$$y_0 := f(x_0), \ldots, y_n := f(x_n) \in \mathbb{R}$$

- Gesucht: Polynom P_n mit

$$P_n(x_i) = f(x_i) \quad i = 1, \ldots, n$$

Es gilt: Das Lagrange-Interpolationsproblem ist eindeutig lösbar und das Polynom P_n besitzt die explizite Darstellung

$$P_n(x) = \sum_{i=0}^{n} f(x_i) l_{i,n}(x)$$

wobei

$$l_{i,n}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Schreiben Sie eine Funktion

`myLagrange(x,y,x_eval)`,

welche das Polynom P_n, ausgewertet an den Stellen `x_eval`, zurückgibt. Um die $l_{i,n}$ zu bestimmen, schreiben Sie eine weitere Funktion

`getL(x,x_eval)`,

welche diese $l_{i,n}$ an den Stellen `x_eval` auswertet und die Matrix

$$L = \left(l_{i,j}\right)_{\substack{1 \leq i \leq \texttt{length(x)} \\ 1 \leq j \leq \texttt{length(x\_eval)}}} = \left(l_{i,n}\left(\texttt{x\_eval(j)}\right)\right)_{\substack{1 \leq i \leq \texttt{length(x)} \\ 1 \leq j \leq \texttt{length(x\_eval)}}}$$

mit den ausgewerteten Punkten zurückgibt.
Testen Sie Ihr Programm für die Funktion

$$f(x) = \tan(x)$$

und Stützstellen

$$x_i \in \{-1.5, -0.75, 0, 0.75, 1.5\},$$

sowie für die Funktion

$$f(x) = x^3$$

und Stützstellen

$$x_i \in \{-2, -1, 0, 1, 2\}.$$

Plotten Sie anschliessend die Funktionen und das Interpolationspolynom P_n.
Hinweis: Schreiben Sie zusätzlich eine Funktion `myLagrangeMain`, welche die Stützpaare (x_i, y_i) und die auszuwertenden Stellen `x_eval` anlegt. Sie können dann die Funktion `myLagrange` aufrufen und anschliessend mit dem zurückgegebenen Vektor das Schaubild erzeugen.

4. **Programmieren**

a) **Glückliche Zahlen**
Die glücklichen Zahlen sind eine Folge von Zahlen, welche wie folgt entsteht:
Gegeben sei eine Folge von natürlichen Zahlen
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 ...

- Die 1 ist definitionsgemäß glücklich.
- Zunächst wird jede zweite Zahl gestrichen (also alle geraden Zahlen)
  1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 ...
- Die Zahl 3 ist die nächste nichtgestrichene Zahl. Also wird jede dritte noch stehende Zahl gestrichen
  1 3 7 9 13 15 19 21 25 27 ...
- Die nächste Zahl wäre dann 7, also würde jede siebte Zahl gestrichen werden. Danach die 9, also jede neunte Zahl streichen usw.

Quelle: http://de.wikipedia.org/wiki/Gl%C3%BCckliche_Zahl

Schreiben Sie eine Funktion

`luckyNumber(n)`,

welche alle glücklichen Zahlen bis zu einer vorgegebenen Zahl $n \in \mathbb{N}$ zurückgibt.

b) **Primzahlpaare**
Primzahlpaare sind Primzahlen, deren Abstand zwei beträgt. Also

$$(3,5),(5,7),(11,13),(17,19),\ldots$$

Schreiben Sie eine Funktion

`pairsOfPrimes(n)`,

welche alle Primzahlpaare bis zu einer vorgegebenen Zahl $n \in \mathbb{N}$ zurückgibt.
Hinweis: Zur Ermittlung der Primzahlen bis n benutzen Sie die Funktion `primes`.

Zusatz: Schreiben Sie die Funktion `primes` selbst. Benutzen Sie dabei die Methode „Sieb des Eratosthenes“
http://de.wikipedia.org/wiki/Sieb_des_Eratosthenes.

5. **Lineare Regression**

a) Laden Sie die Datei `linReg.txt` von der Homepage runter und lesen Sie diese in MATLAB ein.

b) Stellen Sie die Daten als (x,y)-Punkte in einem Koordinatensystem dar.
Welcher Zusammenhang besteht zwischen den x- und y-Werten?

c) Wie Sie wahrscheinich festgestellt haben, besteht ein linearer Zusammenhang zwischen den Datenpunkten, d.h.

$$y_i \approx a + bx_i.$$

Man kann daher eine Gerade finden, so dass die Punkte in der Umgebung dieser Geraden liegen. Die Aufgabe besteht darin diese Gerade (also a und b) zu finden. Im Allgemeinen und auch in unserem Beispiel existiert keine Gerade, die alle Punkte exakt trifft. Deshalb verfolgt man den Ansatz

$$y_i = a + bx_i + \varepsilon_i \quad \forall\, i = 1,\ldots,n$$

wobei $n \in \mathbb{N}$ die Anzahl der Datenpunkte bezeichnet. Dabei bezeichnet ε_i den Fehler (auch: Residuum). Ziel ist es a und b so zu bestimmen, dass der Fehler $\varepsilon := (\varepsilon_1,\ldots,\varepsilon_n)$ bzgl.

einer Norm minimiert wird. Bei der „Methode der kleinsten Fehlerquadrate“ minimiert man die euklidische Norm, in unserem Fall also

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - (a + bx_i))^2 \to \min$$

Es folgt, dass das Minimum angenommen wird für

$$b = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$
$$a = \bar{y} - b\bar{x}$$

wobei $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ und $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ das arithmetische Mittel der x- bzw. y-Werte bezeichnet.

Schreiben Sie ein Skript, welches das Problem für die Daten aus Teilaufgabe (a) löst, d.h. bestimmen Sie a und b, und stellen Sie die Geradengleichung auf und zeichnen Sie diese in das obige Schaubild mit den Datenpunkten.

Hinweis: Machen Sie sich klar, wie Sie Ihr Programm strukturieren.