Übungen zum MATLAB Kurs
Teil 1
29.09.04

Indizierung
Erzeugen Sie eine 5 x 5 Matrix A mit der Funktion rand
Überlegen und testen Sie die Ergebnisse der folgende Ausdrücke:

- `A([3 5],:)`
- `A(2,:)`
- `A([3,5])`
- `A(:)  wie setzt sich das Ergebnis zusammen (?)`

Logische Arrays
Berechnen und überprüfen Sie die folgenden logischen Ausdrücke bzgl der Matrix
`B = [1 0  9  0; 4  0  5  0 ; 8  5 7 8]`

- `any(B)`
- `all(B)`
- `all(B')`
- `all(any (B))`
- `[z s] = find (B)`
- `any( B( 1:2,3:end))`

Bearbeiten von Matrizzen
Erzeugen Sie sich eine Matrix A (z.B. mit der Funktion eye, magic oder rand). Wenden Sie die folgenden MATLAB Funktionen darauf an.

- `rot90(...)`
- `permute(...)`
- `fliplr(A)`
- `flipud(A)`

Plotten
Plotten Sie die Funktionen e^x und ln x für verschiedene Werte eines Vektors x.
Benutzen Sie zum Darstellen der jeweiligen Funktionen die MATLAB Funktionen subplot(...) in Verbindung mit plot(...).

Matrixinitilisierung
Erzeugen Sie eine n x m Matrix, dessen Spalten jeweils gleich sind. (Tipp: “Tonys Trick” siehe Präsentation)

Matrix-Initialisierung, Zeitmessung
Vergleichen Sie die folgenden Arten der Initialisierung von Arrays
Zum Vergleichen der Bearbeitungszeiten können Sie die Funktionen tic and toc

```
val = 11;
sz = [1e3 , 1e3];

tic
A = repmat(val, sz);
toc

tic
B = val * ones ( sz);
toc

tic
C( prod( sz)) = val;
C = reshape(C, sz);
C(:) = C(end);
toc

tic
D = val( ones( sz));
toc
```

Mittelwertberechnung
Berechnen Sie den Mittelwert der Zeilen für alle Einträge der Matrix x, die kleiner als 0.5 sind.

```
x = [ 0.35  0.13 0.76; 0.57 0.46  0.24; 0.98 0.22 0.67];
```

(Lösung: rowmeans.m)

Maskieren von Elementen, Maximalwerte
Finden Sie die Maximalwerte jeder Zeile einer 2D Matrix und erzeugen Sie anschließend eine andere Matrix der gleichen Größe, die jedoch nur die Maximalwerte an den ursprünglichen Positionen enthält.
(Lösung: findmax_setzero.m)

2D Normalverteilung,
Schreiben Sie eine Funktion, die eine 2dimensionale Gaussverteilung am Bildschirm ausgibt (surf (...)). Die Verteilung berechnet sich nach folgender Formel:

g(x,y) = 1/(2 pi sigma) * exp(-1/(2 sigma^2)(|x-mx|^2 + |y-my|^2);

Als Parameter sollen die Vektoren (x und y) mit den Gitterwerten, ein Mittelwert (mx) in x, ein Mittelwert in y Richtung (my) und die Standardabweichung sigma übergeben werden.
Mögliche Werte zum Testen der Funktion können wie folgt gewählt werden:
X=0:0.01:1
Y=0:0.01:1
mx = 0.2
my = 0.8
sigma = 0.1

(Lösung: gauss_func.m)

Histogramm, Veränderung der Diagrammbezeichnungen
Schreiben Sie ein Script mit dem Sie ein Zeichenhistogramm für einen gegebenen Text berechnen und darstellen. Der Text ist in der Datei *input_str.m* abgespeichert. Gezählt werden sollen alle Buchstaben, Leerzeichen, "-" Zeichen und Punkte. Diese Zeichen definieren folglich den Bin-Vektor. Zur Darstellung ist es ratsam statt der Funktion plot die Funktion bar zu verwenden. Mit der Funktion set (...,'xtick',...) kann die x-Achse an die Bins angepasst werden Mit der Funktion set (...,'xticklabel',...) können die Zeichen der einzelnen Bins an der x-Achse dargestellt werden.

(Lösung: letterhist.m)

Normierung, Funktion von Funktionen
Schreiben Sie eine Funktion *"vec_norm"*, die eine Datenreihe (1 x m Matrix) übergeben bekommt und diese als mittelwertfreie (Mittelwert = 0), varianznormierte Datenreihe zurückgibt (Division durch die Standardabweichung).
Schreiben Sie eine weitere Funktion *"minmax_norm"* die ebenfalls eine Datenreihe (1 x m Matrix) übergeben bekommt und diese so normiert, dass der Maximalwert bei 1 und der Minimalwert bei 0 liegt.

Eine dritte Funktion "plotandnorm" soll nun wiederum eine Datenreihe normieren und diese anschließend mit plot(...) darstellen. Die Art der Normierung soll vom Anwender

bestimmt werden, indem er den Namen der Normierungsfunktion als String übergibt .

Eine sinnvolle Lösung könnte sein, dass innerhalb der Funktion plotandnorm der Aufruf der Normierungsfunktion mit Hilfe eines Funktionspointers und der MATLAB Funktion feval(...) erfolgt.

(Lösung: plotandnorm.m)

Animation
Laden Sie die Bilder img1, img2, ... aus dem /image Verzeichnis in ein 3D Array und erzeugen Sie daraus mit der Funktion movie(...) (und getframe(...)) eine Animation.
(Lösung: animation.m)

Bildbearbeitung, Indizierung von Arrays
Schreiben Sie ein Script, das ein Bild darstellt und daraus einen Ausschnitt ausschneidet. Der Benutzer soll die Möglichkeit haben durch Klicken (ginput(...)) von 2 Punkten auf dem Bild ein Rechteck zu bestimmen (linke obere Ecke und rechte untere Ecke), das den Ausschnitt definiert. Stellen Sie den ausgeschnittenen Bereich als neues Bild dar und markieren Sie den ausgeschnittenen Bereich im Originalbild mit roten Kreisen. Beachten Sie, dass die Funktion ginput(...) x und y Werte zurückliefert !

(Lösung: imregion_select.m)

Messwertverarbeitung
Laden Sie die 10 Datenreihen aus der Datei *extracteddata.m* Stellen Sie alle Reihen in einem Plot dar (Funktionen figure zusammen mit hold). Verwenden Sie die Funktion filter (...) um bei den einzelnen Datenreihen das Rauschen zu unterdrücken. Dies lässt sich z.B. mit einem "running average" Filter erreichen, der wie folgt realisiert werden kann.

```
windowsize = 5;
kernel = ones( 1,windowsize)/windowsize;
filt_data (ii,:)= filter(  kernel,1, data( ii,:));
```

Stellen Sie die resultierenden Daten wiederum in einem Diagramm dar und vergleichen Sie, wie gut das Rauschen entfernt wurde (evtl. erneut mit anderer Fenstergröße filtern)

(optional: Um die Datenreihen zu normieren, können Sie eine Varianznormierung durchführen (siehe weiter oben). Dazu können Sie die Funktion vec_norm (siehe oben) verwenden.)

Für die gefilterten Daten lassen sich schließlich mit einer Fouriertransformation die einzelnen Frequenzanteile ermitteln. Kompakt lässt sich dies (z.B. für die Datenreihe Nummer 5) so formulieren:

```
fft_data = fftshift( abs( fft( filt_data( 5,:))));
```

Bitte schauen Sie sich die Hilfstexte zu den einzelnen Funktionen an. Um nun den

Frequenzverlauf graphisch darstellen zu können (interessant ist nur ein Abschnitt aus dem gesamten Verlauf) können Sie statt der Funktion plot(...) auch die Funktion stem(...) verwenden.
(Lösung: data_processing.m)

Weitere Aufgaben finden Sie in den Tutorials und Demos von MATLAB.
