MATLAB

Blockkurs:

27.1.-31.1. und 10.2.-14.2.
9:00-12:00 oder 13:00-16:00

Organisation

- 10 Module je 3h Vorlesung mit Aufgaben
- Projekt:
  - Zuteilung 1. Woche,
  - Abgabe bis Freitag, 21.02., 12:00 per Email
  - Umfang: Matlab Code + kurze Dokumentation (wie benutzt man Ihren Code)
- Kurze Prüfungen in der Woche 24.02.-28.02., nach Vereinbarung

Matlab

- Matrix Laboratory
- Verwendungszweck: Hauptsächlich rechenintensive Datenverarbeitung mit Gleitkommazahlen
- Viele Toolboxen für Spezialanwendungen: PDE, Neural Network, Image processing, Financial toolbox etc. etc.
- Symbolisches Rechnen und exakte Arithmetik möglich (Symbolic Math Toolbox), aber Mathematica besser geeignet

Ziele

- Selbstständiges Arbeiten mit Matlab und Toolboxen
- Funktionsweise verstehen
- Schreiben von einfachen Skripten und Funktionen
- Curriculum der VL in etwa wie “Certified Matlab Associate” Kurs

Material

- Diese Slides
  http://www.math.uzh.ch/?ve_vo_det&key2=1990
- Skript von Felix Fontein (VL Homepage)
- Matlab Hilfe und Matlab Online Hilfe

Zugriff auf Matlab

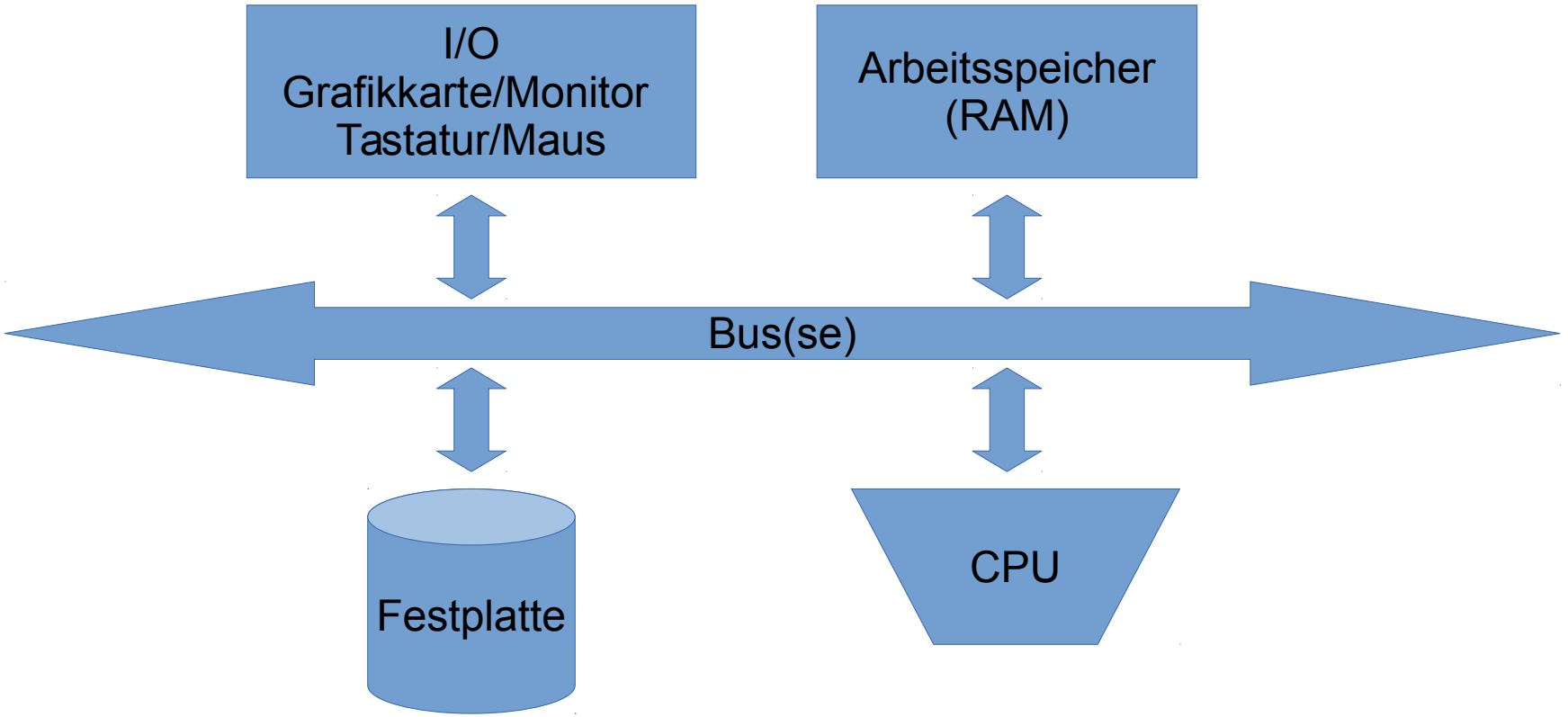
- Hier: Mit IMATH account
- zu Hause:
  - Thinlinc Client
  - Octave: freier Matlab Klon
    www.octave.org
  - Benutzeroberfläche für Octave: qtOctave

Ablauf

- Vorstellung von Befehlen/Theorie
- Befehle gleich selbst Ausprobieren in Matlab
- Aufgaben: selbstständige Bearbeitung 5-20 min

Computer

Sehr stark (!) vereinfachtes Modell:



Programmiersprachen

- CPU verarbeitet Instruktionen im Maschinencode (binär, schwer les- und schreibbar)
- Hochsprachen
  - Erzeugung von Maschinencode (kompilieren) (C++, Pascal, Delphi)
  - Interpretierte Skriptsprachen (JavaScript, Matlab, Sh, Perl, PHP)
  - Mischformen: Erzeugung virtuellen Maschinencodes (Java, C#, .NET Sprachen)

Matlab

- Matlab starten:
  Applications > Science > Matlab
  Passwort eingeben
- Beachte:
  - Dateinavigation links
  - Kommandozeile mitte
  - Variablen im Arbeitsspeicher oben rechts
  - Command history unten rechts
  - Menü oben

Matlab

Eingabe von Befehlen:

- `32 * 3+5`
- `32*(3+5)`
- `sin(3 * pi)`
- `help sin`
- `doc sin`
- `sin` (+F1)
- `si` (+TAB, autocomplete)

Aufgaben

- Berechnen Sie die Anzahl Stunden und Sekunden im Jahr mit Matlab
- Lassen Sie sich die Hilfeseite für die Wurzelfunktion (`sqrt`) anzeigen.
- Finden Sie die Funktion für die n-te Wurzel und berechnen Sie die dritte Wurzel von 10
- Berechne das Multiplikative Inverse von 1+2i

Ausdrücke

- Klammern ()
- Arithmetische Operationen
  * ,+,-, /, ^
- Vergleichsoperationen
  ==, ~=,<=,>=
- Logische Operationen
  &, |,~
- Kein true/false in Matlab.
  False=0, True=1 (bzw. nicht 0)

Operatorpräzedenz

- Achtung: Operatoren haben unterschiedliche Präzedenz
  `(3+5)*2` ist 16, `3+5*2` ist 13

- In absteigender Reihenfolge: (), ^,~, -(Negation), *,\, /, +, -, == etc., &, |

- `doc 'operator precedence'`

Ausdrücke (2)

- Numerische Ausdrücke

```
1
0.23, .23
23e-2 = 23*10^-2
5+4i, 5+4j
```

- Funktionen
  funktionsname(parameter1,parameter2,...)
- Häufig gebrauchte Funktionen:

```
sin, cos, tan, sqrt, exp, log, log2,
log10, floor, ceil, round, abs
```

Variablen

- Zuweisung des Wertes eines Ausdruckes zu einer Variablen
  ```
  x=25
  ```
- Variable erscheint oben rechts im workspace explorer, oder:
  ```
  who
  whos
  ```
- Die Variable kann nun ihrerseits in Ausdrücken verwendet werden
  ```
  sqrt(x)
  ```
- Löschen der Variable
  ```
  clear x
  clear all
  ```

Vordefinierte Variablen

`ans`	Resultat letzte Berechnung
`pi`	
`eps`	Maschinengenauigkeit
`i,j`	sqrt(-1)
`nan`	not a number, z. B. Resultat von 0/0
`inf`	infinity, z. B. Resultat von 1/0
`realmin,realmax`	kleinste/grösste Gleitkommazahl

Diese Variablen bitte nicht anderweitig zuweisen!!
`i=1` z.B. überschreibt den vordefinierten Wert

Tips & Tricks

- Ausgabe unterdrücken mit ;
  `a=23;`
- Mehrere Befehle in einer Zeile mit , oder ;
  `a=23;24*5,c=24`
- `4, 5; % Was ist ans?`
- `format long` verändert die Anzahl angezeigter Dezimalstellen

Aufgaben

- Überschreiben Sie die Variablen i,j, und machen Sie den Effekt wieder rückgängig.
- Gibt es Zahlen a, b, c, so dass a op1 (b op2 c) ungleich (a op1 b) op2 c ist? Was ist dann a op1 b op2 c in Matlab?
  - (op1,op2) = (-,-)
  - (op1,op2) = (+,-)
  - (op1,op2) = (|,&)
  - (op1,op2) = (&,>=)

Aufgaben

- Fügen Sie maximal viele (einfache) Klammern hinzu, ohne den Wert des Ausdrucks zu ändern

```
2 + 4 * - 1 / 3 / 4
2 + 4 == 5 | 3 + 5 ^ - 1 & 2
- 3 < 4 < 5
sin ( n + 1 ) / ( n + 2 )
```

- Zeichnen Sie jew. einen Baum, der beschreibt in welcher Reihenfolge Matlab die Operationen ausführt.

Aufgabe

Welchen Wert hat `ans` jeweils nach jedem Befehl?

```
>> 23;
>> ans ^2;
>> x = sqrt(ans);
>> x^2 - ans;
>> x^2 - ans;
>> ans - x ^2;
```

Skripte

- Man kann mehrere Befehle in einem Skript zusammenfassen. Clicken Sie auf “New Script”
- Geben Sie den Text ein:

```
z=23;
hallo=12 *z
```

- Speichern Sie das Skript unter eintest.m
- Führen Sie das Skript aus durch Click auf den Run Button

Skripte

- Führen Sie das Skript nochmals aus, indem Sie in der Kommandozeile `eintest` eingeben
- Beachten Sie, dass im workspace Variablen `z` und `hallo` erzeugt wurden
- Beachten Sie, dass die Autovervollständigung (mit Tab) für Ihr Skript funktioniert

Kommentare

- Kommentare schreibt man wie folgt

```
% single line comment
%{
multi line
comment
%}
```

- `%% beschreibt einen Codeblock`

Kommentare

- Kommentare ganz oben im Skriptfile erscheinen in der Hilfe

```
% Dies in die ersten Zeilen
% einfuegen
```

- `help eintest`
- `eintest` + F1

für Windowser

Preferences>Matlab>Keyboard>Shortcuts>
Windows Default Set

Funktionen

- Funktionen in Matlab haben kein, ein oder mehrere Argumente und keine, ein oder mehrere Rückgabewerte
- Clicken Sie New > Function, ändern Sie den Funktionsnamen in power8 und speichern Sie die Funktion als power8.m
- Im Editor

```
function y = power8( x )
% Das hier wird wieder in der Hilfe angezeigt
y=x^8;
end
```

- In der Kommandozeile

```
power8(2)
```

Funktionen

- Mehrere Argumente und / oder outputs

```
function [y1, y2] = power8(x1,x2,x3)
y1=x1;
y2=x2+x3;
```

- Aufrufen:

```
[a,b]=power8(1,2,3)
```

- Outputs ignorieren

```
a=power8(1,2,3)
[~,b]=power8(1,2,3)
```

Funktionen

- Achtung: Man kann eine Funktion mit weniger (aber nicht mehr) Argumenten (in oder out) aufrufen. Innerhalb der Funktion kann man mit `nargin, nargout` die Anzahl der tatsächlich übergebenen Argumente abfragen.
- Ausserhalb der Funktion gibt nargin/nargout die Anzahl der Argumente in der Deklaration zurück

```
nargin(@size)
nargout(@sin)
```

Funktionen

- Mehrere Funktionen im File

```
function y=power8(x)
y=mypower8(x);
end

function y=mypower8(x)
y=x^8;
end
```

- Achtung: Matlab findet dann nur `power8`, entsprechend dem Dateinamen!

Variablenscope

```
function y=power8(x)
% kein ; → erscheint in der Ausgabe
y=x^8
% die nächste Zeile erzeugt keine
% Variable im Workspace
meine_variable =5;
end
```

Variablenscope (2)

```
function y=power8(x)
y=x^8;
% die nächste Zeile erzeugt eine
% globale Variable
global globaaal
globaaal=x
end
```

- Zum Anzeigen im Workspace muss man noch in der Kommandozeile eingeben

```
global globaaal
```

anonyme Funktionen

- Man kann Funktionen erstellen, ohne ein .m File zu schreiben

```
f = @(x,y) x^y
f(2,3)
```

- `f` ist dann eine anonyme Funktion vom Datentyp `function_handle`.

```
class(f)
```

- Alte Syntax: `inline` (nicht mehr verwenden)

anonyme Funktionen

- Man kann auch für bestehende Funktionen `function_handle` erhalten

```
g=@sin
g(2)
```

Tips&Tricks

- `function y=power8(x)`
  `y= … % Mehrzeiler`
  `x^8;`

- `disp(x)` zeigt den Wert der Variablen/des Ausdrucks x an (ohne “x=”, “ans=”)

- `return` verlässt das Skript oder die Funktion

- `error('meine Nachricht')` gibt dem Nutzer eine Fehlermeldung aus und beendet die Funktion oder das Skript

Tips&Tricks

- Aufruf von `whos` in einer Funktion zeigt den workspace der Funktion an

Closures

- Achtung: taucht eine workspace Variable in einer anonymen Funktionsdefinition auf, so wird deren Wert zum Zeitpunkt der Funktionsdefinition kopiert

```
a=2;
f = @(x) x+a;
f(2)   % 4
a=3;
f(2)   % still 4

g=@() length(whos)+a
```

Aufgaben

- Schreiben Sie eine Funktion, die ganze Zahlen a und b annimmt und a/b (Integerdivision) und den Rest der Division ausgibt.
- Schreiben Sie eine Funktion `curry(f, g)` die zwei anonyme Funktionen mit einem Parameter annimmt und die Zusammensetzung der Funktionen x → f(g(x)) ausgibt.

Debugger

- Breakpoints setzten durch Click links neben die Zeile
- Schritt durch Click auf den Pfeilbutton
- Wert einer Variablen im Tooltip ersichtlich
- In Funktion zeigt der workspace explorer den workspace der Funktion an.

if

```
if x == 5
   % wird ausgefuehrt, falls x=5
   % sonst nicht
   y=1;
end
```

if

```
if x == 5
    % wird ausgefuehrt, falls x=5
    y=1;
elseif x==6
    z=2;
else
    y=3;
end
```

for

```
for k=1:10
  disp(k^2)
end
```

for (2)

- Beachte: Die Schleifenvariable nicht `i` oder `j` nennen (Wieso nicht?)
- Sprung zum Schleifenbeginn

```
for k=1:10
    continue;
    a=1/0;  % not executed
end
```

- Schleife abbrechen

```
for k=1:10
    break;
    a=1/0;  % not executed
end
```

while

- Schleife ausführen, solange Bedingung gilt

```
a=0
while a<5
    a=a+1
end
```

- Schlechtes Beispiel (Ctrl+C bricht Skript ab)

```
a=6
while a>5
    a=a+1
end
```

Aufgabe

- Schreiben Sie ein Skript, dass `if, for, while, continue` und `break` und einen Aufruf einer selbstgeschriebenen Funktion enthält und überwachen Sie die Ausführung mit dem Debugger

Variablenscope (3)

- Variablen, die innerhalb einer Schleife oder eines `if` Konstruktes definiert wurden sind dennoch ausserhalb der Schleife gültig.

```
clear x
for k=1:3, x=k; end
x  % kein Fehler
```

Aufgaben

- Schreiben Sie eine Funktion fibo(n), die die Glieder der Fibonaccifolge berechnet 1,1,2,3,5,8, ...
  - einmal ohne Schleife
  - einmal mit Schleife
- Erweitern Sie die Funktion so, dass sie optional beliebige Anfangswerte F1, F2 annimmt und zusätzlich das n-1-te Glied ausgibt
- Schreiben Sie einen Hilfetext

Aufgaben

- Verifizieren Sie numerisch, dass

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

- Schreiben Sie eine Funktion, der eine Funktion f und 2 Werte (Intervallgrenzen) a,b annimmt und eine Nullstelle von f mit dem Bisektionsverfahren im Intervall [a,b] findet. Wenden Sie Ihre Funktion auf `sin` an um `pi` zu berechnen.

Aufgaben

- Schreiben Sie eine Funktion, die eine Funktion f, deren Ableitung und einen Startwert annimmt und eine Nullstelle von f mit dem Newtonverfahren sucht.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- Berechne wiederum `pi`.

Matrizen

- Alles ist eine Matrix (oder Synom ein array), d. h., Zahlen sind 1x1 Matrizen, Vektoren nx1 oder 1xn Matrizen.
- Erzeugung mit `[]`:

```
A=[1 2 3; 4 5 6]
A=[1, 2, 3; 4, 5, 6]
```

- Leerzeichen oder `,` trennt Spalten und `;` Zeilen
- Zugriff auf Werte

```
A(2,3)
```

% 2. Zeile, 3. Spalte (=6)

- Werte verändern

```
A(2,3)=7
```

Matrizen

- `diag` erzeugt eine Diagonalmatrix oder gibt die Diagonale zurück

```
diag(A)
B=diag([1,2])
```

- `diag(A,k)` gibt (oder setzt) die k-te Nebendiagonale

- Zusammensetzung von Matrizen

```
C=[B A]
D=[C; A B]
```

- Werteditierung über GUI auch durch Doppelclick im workspace explorer möglich; siehe auch toolbar > new variable

Matrizen

- 5x5 Einheitsmatrix
  `eye(5)`
- 3x8 Matrix aus 1en
  `ones(3,8)`
- 4x6 Matrix aus 0en
  `zeros(4,6)`
- Zufallsmatrix
  `rand(5)`
  `randn(3,4)`
- Magic square
  `magic(5)`
- Leere Matrix
  `[]`

Matrizenoperationen

`+,-`	Elementweise Summe/Differenz `A+A+A`
`', .'`	Konjugiert transponierte Matrix, z.B. `A',` transponierte Matrix `A.'`
`*`	Matrizenmultiplikation (Dimensionen müssen stimmen, oder ein Faktor ist ein Skalar) `B*A, 3*B`
`^`	Potenz eine quadratischen Matrix
`.*, ./,.\`	Elementweise Multiplikation/Division
`.^`	Elementweise Potenz
`|,&,~`	Elementweise logische Operationen
==,~=,<,><=,>=	Elementweise Vergleichsoperationen
`inv`	Inverse Matrix
`/, \`	A \ B löst das Gleichungssystem A*x=B, A/B löst A=x*B (s. später)

Matrizenoperationen

- Grösse einer Matrix

```
size(A)
size(A,1)
size(A,2)
```

- Form ändern, Werte werden “Spalten zuerst” aufgefüllt

```
A2=reshape(A,3,2)
```

- Mehrere Kopien einer Matrix Zusammenkleben

```
A3=repmat(A,2,2)
```

- Die meisten Funktionen agieren auf Matrizen Elementweise

```
sin(A)
```

Zugriff auf Elemente

- Untermatrix:

```
A3(1:3, 2:4)
A3(1:3, [1 4 3])
A3(1:3, 1:2:6)
A3(1:5)
A3(1, 1:end)
A3(1, 1:end-1)
A3(:,1:3)
A3(:)
```

- Genauer: der Ausdruck

```
a:b:c
```

erzeugt einen Vektor (Zeilenmatrix) mit Werten von a bis c, Schrittgrösse b

```
1:3:10
.1:.2:5
plot(sin(0:.01:2*pi))
```

Zugriff auf Elemente

- Nur ein Index

```
A3(5)
```

- Was passiert hier?

```
A=zeros(4,5);
A(:)=1:numel(A)
```

Zugriff auf Elemente

- Beachte: Indizierung in Matlab 1 basiert, nicht 0 basiert
- Zugriff mit logischen Matrizen

```
A>=3
A(A>=3)
```

- `find` gibt die Indizes zu wahren Einträgen zurück

```
find(A>=3)
A(find(A>=3))
```

Tricks

- Löschen von Spalten/Zeilen

```
A=rand(5)
A(:,[2,4])=[]
```

- Bei Zuweisungen wird die array-Grösse ggf. automatisch angepasst.

```
A=[1 2 3 4]'
size(A)
A(25)=3
size(A)
```

Hilfreiche Funktionen

- `max, min` findet das Maximum/Minimum eines Vektors. Operiert wie fast alle Matlab-Befehle auf Matrizen spaltenweise

  ```
  max(A(:))
  max(A)
  ```

- ebenso `sum, mean, median, std`

- Oft kann man mit einem 2. Parameter die Dimension bestimmen, entlang derer die Funktion operiert

  ```
  mean(A,2)
  ```

Hilfreiche Funktionen 2

- `isequal` prüft, ob 2 Matrizen gleich sind (als ganzes, nicht komponentenweise)
- `kron` berechnet das äussere (Tensor-)Produkt zweier Matrizen

kron(A,B) = $\begin{pmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ \dots & \dots & \dots & \dots \\ A_{m1}B & \dots & \dots & A_{mn}B \end{pmatrix}$

- `linspace(0,2*pi, 100)`

for (3)

- Generell wird in

```
for ttt=A
   % do something with ttt
end
```

der Code in der Schleife einmal aufgerufen für jedes Element von A (falls eindimensional) oder für jede Spalte von A (falls zweidimensional)

Aufgaben

- Konstruiere die folgenden Matrizen effizient in Matlab (Hinweis: `help diag`):

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 4 & 0 \\ 2 & 2 & 2 & 2 & 0 & 0 & 5 \\ 2 & 2 & 2 & 2 & 6 & 6 & 7 \\ 2 & 2 & 2 & 2 & 6 & 6 & 8 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 5 & 1 & 0 & 0 & 0 & 5 & 5 & 6 & 7 & 8 \\ 6 & 6 & 0 & 2 & 0 & 0 & 6 & 1 & 0 & 3 & 0 \\ 7 & 7 & 0 & 0 & 3 & 0 & 7 & 0 & 2 & 0 & 4 \\ 8 & 8 & 0 & 0 & 0 & 4 & 8 & 5 & 6 & 7 & 8 \end{pmatrix}$$

Aufgaben

- Erzeugen Sie eine 50x50 Zufallsmatrix und finden Sie die Werte und Indizes der 3 grössten Elemente
  Hinweis: `sort, ind2sub, sub2ind`

Aufgaben

- “Invertiere” einen Zufallsvektor, d. h., drehe die Reihenfolge der Einträge herum.
- Schreibe eine Funktion, die einen Vektor/Matrix zyklisch verschiebt (ohne `circshift` zu verwenden)
- Erzeuge eine MxN Zufallsmatrix A.
  - Stell Dir die Matrix als Schachbrett vor und erzeuge eine Matrix mit den gleichen Einträgen wie A auf den schwarzen Feldern und Nullen sonst.
  - Falls M,N gerade, erzeuge eine Matrix nur mit den schwarzen Einträgen.

Aufgaben

- Wie berechnet man das arithmetische Mittel aller Einträge einer Matrix?
- Was machen die Funktionen rank, null, rref, orth?
- Verifizieren Sie nochmals, dass $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ aber diesmal ohne Schleife.

Aufgaben

- Was passiert hier?
  ```
  hist(rand(1,100000),100)
  hist(randn(1,100000),100)
  ```
- Was machen die Ausdrücke
  - `A'.'.^A;`
  - `diag(diag(A)) + diag(diag(A, 1), 1) + diag(diag(A, 2), 2)`
    wobei A eine nxn-Matrix ist mit n>=4?

Aufgaben

- Überführe die Matrix

  ```
  A = reshape (1:20 , 4, 5) ';
  ```

  durch elementare Zeilenumformungen in reduzierte Zeilenstufenform

- Erzeuge eine Wertetabelle mit N Werten der Funktion f(x) im Intervall [a,b], d.h. eine Nx2 Matrix mit Einträgen x, f(x) in jeder Zeile.

  - f(x) = x^5 -4*x +1
  - f(x) = exp(i*x/10)

Aufgaben

- Gegeben eine Matrix A, berechnen Sie die Dimension des Raumes der mit A kommutierenden Matrizen, d. h., Matrizen X, so dass A*X=X*A
- Testen Sie Ihr Skript auf Zufallsmatrizen

Aufgabe

- Finden und beheben Sie den Fehler im Skript `fehlersuche_1.m` auf der Vorlesungshomepage, dass eine geordnete Liste mit Zufallszahlen ausgeben soll.

plot

- Plotten eines Vektors

```
x=linspace(0,2*pi,200);
y1=sin(x);
y2=cos(x);
plot([y1',y2'])
```

- Richtige x-Achsenwerte

```
plot(x,[y1',y2'])
```

- Was macht

```
plot(y1,y2)
```

  und was

```
plot([y1;y2])
```

  ?

plot (2)

- Legende:
```
legend('sin', 'cos')
```
- Titel
```
title('Mein plot');
```
- Farben und Linien
```
plot(x,y1,'r*-', x,y2, 'k:')
```
- Achsenbeschriftung
```
xlabel('das ist x')
ylabel('und das y')
```

plot (3)

- Mehrere Plots in ein Koordinatensystem

```
plot(x,y1)
hold on
plot(x,y2)
hold off
```

- Mehrere Plots in verschiedene Koordinatensysteme

```
figure(1)       % Wähle Figure 1 aus
plot(x,y1)
figure(2)       % Wähle Figure 2 aus
plot(x,y2)
```

plot (4)

- Mehrere kleine Plots in eine Figure

```
subplot(1,2,1)
plot(x,y1)
subplot(1,2,2)
plot(x,y2)
```

- Allgemein wählt `subplot(a,b,n)` das n-te Koordinatensystem in einer axb Matrix von Koordinatensystemen

mehr plots

- `clf` löscht die aktive Figure
- Bar graphen
  ```
  bar(rand(1,10))
  ```
- Pie Chart
  ```
  pie(rand(1,5))
  pie(rand(1,4),[0,0,1,0])
  pie3(rand(1,5))
  ```
- scatter plot
  ```
  scatter(y1,y2)
  ```

Aufgabe

- Zeichnen Sie die ersten 4 Besselfunktionen (nu=0,1,2,3) 1. und 2. Art in ein 1x2 array von Plots, mit entsprechenden Beschriftungen

  (Hinweis: `besselj`, `bessely`)

- Gebe `help plot` ein, um den Hilfetext des plot-Befehls zu erhalten. Erzeuge damit die folgenden Plots:
  - die Funktion f = x^2-.5 auf [-1,1] mit roter gestrichelter Linie
  - die Funktionen f=sin(s*x) mit s=1,2,3,4 in verschiedenen Farben, mit Legende
  - f = sin(2 * pi * x), wobei keine Linie dargestellt werden soll, sondern die Funktionswerte als kleine schwarze Kreise an den Stellen x = n/10

3D plots

- 3D Linie

```
plot3(x,y1,y2)
```

- 3D Oberfläche

```
[xx,yy]=meshgrid(x,x);
surf(xx,yy,sin(xx).*cos(yy))
```

- Frage: Was passiert, wenn man * statt .* schreibt?

noch mehr plots

- `mesh(peaks)`
- `contour(peaks)`
- `pcolor(peaks)`

Aufgabe

- Stellen Sie die folgenden Funktionen auf einem sinnvollen Definitionsbereich graphisch dar:
  - f(x,y) = cos(xy) + sin(x)
  - f(x,y) = 1/(x^2 + y^2)

Animation

- Mehrere plots zu einem Video zusammensetzen

```
clear M
for k=1:40
    plot(x,y1*sin(2*pi* k/40))
    axis([0,2*pi,-1,1])
    M(k)=getframe;
end
movie(M,2)
```

Animation (2)

- Write to avi file

```
writerObj =
VideoWriter('myvid.avi');
open(writerObj);
for k=1:40
    plot(x,y1*sin(2*pi* k/40))
    axis([0,2*pi,-1,1])
    writeVideo(writerObj, getframe)
end
close(writerObj)
```

Aufgabe

- Zeichnen sie die ersten Eigenmoden einer quadratischen Trommel mit Seitenlängen 1.
- Erstellen Sie ein Video von einer Überlagerung von zwei Eigenmoden.

  Hinweis: Die Eigenschwingungen der Trommel sind gegeben durch die Formel

$$\sin(m\pi x)\sin(n\pi y)\sin\left(c\sqrt{m^2+n^2}\,t+\varphi\right)$$

Datentypen

`logical`	0 oder 1
`single, double (standard)`	Gleitkommazahlen
`int8, int16, int32, int64`	Ganzzahlen (integers)
`uint8, unit16, uint32, uint64`	nichtnegative Ganzzahlen (unsigned integers)
`char`	ein Zeichen (16 bit), ein string ist eine 1xn char-matrix
`cell`	“Variabler Datentyp”, siehe später
`table`	“Matrix” in der die Zeilen und Spalten zusätzlich Namen tragen können
`struct`	Datenstruktur mit Feldern
`function_handle`	
`object, map, graphics handle, time series etc.`	nicht behandelt

Datentypen

- Typ überprüfen

```
a= 1==0
class(a)
b=rand(5)
class(b)
```

- Konvertierung mit gleichnamigen Funktionen

```
a=1.5
b=int8(a)
class(b)
char(1:1000)
```

- Achtung: Kombination verschiedener Typen nur zum Teil erlaubt, und unerwartete Ergebnisse

```
c=b*2.7
class(c)
b*single(2.7)
```

Gleitkommazahlen

- auch: Fliesskommazahlen. Matlab Datentypen `single` und `double`, letzterer mit doppelter Genauigkeit
- Werte double:

$$s\, m\, 2^{e}$$

mit
Vorzeichen s=+1 oder -1
Mantisse $m = 1 + x\, 2^{-52},\ x \in \{0, \ldots, 2^{52} - 1\}$
Exponent e=-1022,..., 1023

- Maschinengenauigkeit `eps`

Gleitkommazahlen

- Zusätzlich spezielle Werte: `inf`, `-inf`,`nan`, +0=-0

```
0==-0
1/0
-1/0
0/0
nan == nan
```

- Nicht in dieser Form darstellbare Zahlen werden gerundet

```
eps
2^-52
1+eps == 1
1+eps/2 == 1
1+3*eps/4 == 1
```

Aufgabe

- Stelle die folgenden Zahlen als Gleitkommazahl dar, oder begründe, wieso das nicht (exakt) möglich ist:

  15

  0.5

  0.1

  1/3

  1/256

Char

- Zeichenketten (strings) sind 1xn matrizen vom Typ `char`

```
s='hallo'
class(s)
size(s)
s(2)
```

- strings aneinandersetzen

```
s2 = [s ' welt']
```

String-Funktionen

- Konversion Zahl<->string

```
num2str(1.234)
str2double('1.234')
```

- Alternativ:

```
sprintf('Die Zahl ist %f oder %d.',
1.234, 5)
sprintf('Die Zahl ist %.1f.', 1.234)
sprintf('Die Zahl ist %f. ', [1,2,3,4])
```

- Einen Teil-string ersetzen

```
strrep('Ein Test', 'Test', 'Toast')
```

Datum

- Daten/Uhrzeiten werden in Matlab dargestellt als double, die Anzahl vergangener Tage (und Bruchteile davon) seit dem 0. Januar 0.

```
now
```

- Konvertierung in Jahr/ Monat/ Tag/ Stunde/ Minute/ Sekunde bzw. string

```
datevec(now)
datestr(now)
datestr(0)
```

- Umgekehrt:

```
datenum('1.1.2001 09:00', 'dd.mm.yyyy
HH:SS')
```

struct

- Eine structure hat Felder auf die über (structname).(feldname) zugegriffen werden kann. Dient zur Gruppierung zusammengehöriger Werte.

```
calib=struct  % optional
calib.focal_length=1.2
calib.shutter_time=0.01
calib.date=now
fieldnames(calib)
```

Wie üblich wird gegebenenfalls die array-Grösse angepasst

```
calib(3).focal_length=1.5
size(calib)
```

Cell arrays

- die Felder von cell arrays (Matrizen vom Typ cell) können beliebige Datentypen beherbergen. Erzeugung:

```
s={ [1 2 3], 'asdasd'; 2, 1==0 }
class(s), size(s)
```

- Zugriff auf Teil-arrays wie gewohnt, z.B.,

```
a=s(1,1)
class(a) % Achtung: ist noch eine cell!
```

- Zugriff auf den Wert

```
a=s{1,1}
class(a)
```

Cell arrays

- Frage: Was ist dann s{1,1:2} ?

```
class(s{1,1:2})  % error
```

- Es ist eine Komma-separierte Liste, äquivalent dazu, die einzelnen Felder mit Kommas getrennt einzugeben.

```
s2={1,2,[3,4]}
[s2{:}]
```

Beliebig viele Eingabe/Ausgabeargumente

- Beliebig viele Eingabe-/Ausgabeargumente kann man mit den keywords `varargout`, `varargin` zulassen. `varargin` ist dann ein 1xN cell array. Ebenfalls muss man in `varargout` ein 1xM cell array füllen.

```
function varargout =
myfunction(myotherparam, varargin)
```

- `nargin(f), nargout(f)` sind negativ, falls `varargin` bzw. `varargout` benutzt wurde

Aufgabe

- Schreiben Sie eine Funktion `mysum`, die beliebig viele Eingabeargumente akzeptiert und deren k Potenzsummen in die k (beliebig viele) Ausgabeargumente schreibt.
- Erweitern Sie Ihre Funktion `curry`, so dass die Argumente (Funktionen) eine beliebige, aber feste Anzahl Eingabeparameter enthalten können.

Mehrdimensionale Arrays

- Bisher: 2 dimensionale Matrizen. Mehr Dimensionen sind möglich

```
A=rand(4,5,7,4)
size(A)
```

- Zugriff auf Elemente entsprechend

```
A(1,2,3,4)
```

Funktionen für mehrdimensionale arrays

- Die meisten Funktionen operieren auch auf mehrdimensionalen arrays, insbesondere repmat, reshape.

```
A=rand(3); B=repmat(A,[1,1,2])
```

- Dimensionen verschieben

```
C=shiftdim(B,1);
B(3,2,1)
C(2,1,3)
```

- singleton-Dimensionen entfernen

```
D=reshape(B,[3,3,1,1,2,1,1])
D2=squeeze(D)
isequal(B,D2)
```

Aufgaben

- Überlege, welchen Typ und Wert die folgenden Ausdrücke haben, mit A=[1 2;3 4*i]. Verifiziere die Antwort.

```
isequal(A, A');
sum(sum(A - A')') == 10;
sum(diag(A == A')) & abs(sum(sum(A)));
int8(255) + 1;
int32(int8(255) + 1) * 5;
logical(255) + 1;
int64(double(int64(2)^63) - 1) - int64(2)^63
```

Aufgaben

- Erzeugen Sie einen string der Form
  Der Wert von x^2 an der Stelle 2 ist 4.
  Der Wert von x^2 an der Stelle 2.1 ist 4.41.
  ...
  (bis x=3) (Hinweis: neue Zeile mit `\n`.)

- Berechnen Sie die den numerischen Datumswert Ihres Geburtstages. Wie viele Tage sind Sie alt?

- Was macht `mat2cell`?

Daten lesen/speichern

- Oft muss man Daten in Dateien speichern oder von dort auslesen. Einfachster Weg:

  ```
  save
  ```

  speichert den kompletten workspace im file matlab.mat.

- Mit load lädt man den workspace wieder

  ```
  clear
  load
  ```

save/load

- Anderer Dateiname, bzw. nur bestimmte Variablen

```
A=rand(3); B=rand(4);
save myfile.mat A B
clear
load myfile.mat A
who
```

- Speicherung als textdatei (anstatt binär)

```
save myfile.txt -ascii
```

Bemerkung

- In Matlab sind die folgenden Befehle äquivalent

```
myfunction a b c
myfunction 'a' 'b' 'c'
myfunction('a' ,'b', 'c')
```

  Die Argumente werden automatisch in ' ' verpackt.
  Insbesondere müssen die Argumente alle strings sein.
  Die ersten Schreibweisen heissen command syntax, (vs. function syntax)

- Beachte
  `sin pi`
  ist ein Fehler. (Wieso?)

- Funktioniert `disp hallo welt` ?

File I/O

- Oft muss man Daten aus speziell formatierten Textdateien oder spreadsheets lesen/schreiben.
- Einfachster Weg, Daten eines Typs:

```
A=rand(4)
dlmwrite('myfile.txt', A);
%later
B=dlmread('myfile.txt')
```

- Trennzeichen (, ; blank) werden erkannt oder man legt sie manuell fest (2. Argument).

File I/O

- Will man Dateien mit verschiedenen Datentypen lesen/schreiben geht es i. Allg. nicht in einer Zeile

- Man öffnet eine Datei und schreibt Daten wie folgt:

```
fid = fopen('myfile2.txt', 'w')
fprintf(fid, 'Datum; Messwert\n')
fprintf(fid, '%s; %f\n', …
        datestr(now), 1.245)
fclose(fid)
```

File I/O

- Zum Einlesen von Daten:

```
fid=fopen('gas.csv');
HDR = textscan(fid,'%s %s',1, ...
                    'delimiter',';');
DATA = textscan(fid,'%s %f', ...
                    'delimiter',';');
fclose(fid);
```

- textscan gibt ein cell array zurück, mit Elementen entsprechend den Spalten. Die eigentlichen Daten bekommt man also wie folgt:

```
measval=DATA{2};
meastimes = datevec(DATA{1}, …
                    'dd.mm.yyyy HH:MM');
```

File I/O

- Die gezeigten Wege zum Lesen/Schreiben von Daten sind nicht die einzigen. Vgl. fscanf, fread, fwrite, importdata, xlsread, xlswrite
- Die Matlab GUI enthält auch einen Assistenten zum Import von Daten und zum Erzeugen von Import-Skripten. (Import Data Button)

Datenanalyse

- `mean, stdvar, cov` berechenen den Mittelwert, Standardabweichung und Varianz von Daten

- Sind die Daten in Gruppen eingeteilt, kann man mit grpstats die Grössen für die einzelnen Gruppen berechenen.

```
data = rand(50,1);
groups = randi(3,50,1);
m = grpstats(data, groups, 'mean')
ugrps = unique(groups)
```

Aufgaben

- Schreiben Sie ein Skript, dass die (realen) Wetterdaten im file 'wetter.txt' importiert.
- Plotten Sie die Daten sinnvoll.
- Berechnen und plotten Sie die Jahresmittel-Höchst- und Tiefsttemperaturen. Sieht man einen Effekt der Klimaerwärmung?

Lineare Gleichungssysteme

- A eine invertierbare nxn Matrix, b ein n-Vektor. Man löst das Gleichungssystem Ax=b durch x=A\b oder x=inv(A)*b (letzteres nicht empfohlen).
- In der Praxis:
  - Gleichungssysteme fast immer überbestimmt
  - Keine exakte Lösung dank Rauschen
  - Man will statt einer exakten Lösung das x, dass den Fehler norm(A*x-b) minimiert.

Lineare Gleichungssysteme

- Dieses x wird ebenfalls durch
  x= A\b
  berechnet.

Aufgabe

- Betrachten Sie die Hilfe für die Funktion `polyfit`, und schreiben Sie selbst eine äquivalente Funktion.
- Fitten Sie ein Polynom durch die Temperaturdaten des letzten Jahres und plotten Sie das Ergebnis. Wie genau können Sie das Wetter “vorhersagen”.

Komplexität

- Seien f,g Funktionen auf den natürlichen Zahlen. Dann sagt man f=O(g) falls ein c und ein n0 existiert, so dass f(n) < c g(n) für alle n> n0.
- Man benutzt die Notation, um die Laufzeit eines Algorithmus als Funktion der Grösse n der Eingansdaten abzuschätzen.
- Beispiel: Algorithmus zum Suchen des grössten Elements einer Liste. n=Länge der Liste. Laufzeit O(n).

Komplexität

- Naive Multiplikation von 2 nxn Matrizen: Man braucht O(n^3) Rechenoperationen.

Peformance

- Laufzeitmessung von Funktionen

```
A=rand(2000);
tic; svd(A); toc
```

- `tic` startet einen timer, `toc` gibt die seither vergangene Zeit zurück.

- Man kann mehrere timer starten

```
T1=tic;
% wait
T2=tic;
elapsed1=toc(T1)
elapsed2=toc(T2)
```

Aufgaben

- Schreiben Sie ein Programm, dass die Komplexitaet der Matrizenmultiplikation (ungefähr) misst.
- Schreiben Sie ein Programm, dass die Komplexität der Operation svd misst.

Performancekiller

- ```
  tic; x=[1:100000]; toc
  tic; x=[]; for k=1:100000, x(k)=k;
  end, toc
  ```
- Faktor 200 (!)
- Problem 1: Variable x wechselt in jeder Iteration die Grösse.
- Problem 2: Matlab ist interpretiert, d. h., Schleifen sind langsam.
  ```

Performancekiller

- ```
  tic; x=[1:100000]; toc
  tic; x=zeros(1,100); for
  k=1:100000, x(k)=k; end, toc
  ```
- Take-home-message:
  - Schleifen vermeiden
  - Variablen sollten in der Schleife die Grösse nicht ändern.
  ```

Aufgabe

- Schreiben Sie eine Funktion, die ein 4-dimensionales array mit 1en im Schachbrettmuster erzeugt, explizit mit 4 Schleifen, und ohne Schleifen. Um wieviel können Sie die Laufzeit reduzieren?
- Erweitern Sie Ihre Funktion so, dass sie ein n-dimensionales Schachbrett-array erzeugen kann

Symbolic Math Toolbox

- Man kann mit Matlab auch symbolisch rechnen
- Der Befehl

  ```
  syms x y
  ```

  deklariert symbolische Variablen mit Namen x und y und speichert sie in den Matlab-Variablen x,y

  ```
  class(x)
  ```

- Man kann dann mit x und y symbolisch rechnen, z.B. neue symbolische Ausdrücke erzeugen:

  ```
  cc=sqrt(x+y)
  class(cc)
  f(x) = sqrt(x)
  ```

Symbolic Math Toolbox

- Zahlen in symbolischer Darstellung (beliebige Präzision)

```
a=sym('1233123123123123.1232131231232131
23123123')
sym('10/3')
pp=sym('pi')
sin(pp)
```

- Numerische Auswertung eines symbolischen Ausdruckes

```
vpa(pp)
vpa(pp,500)
```

Symbolic Math Toolbox

- Ausdrücke vereinfachen

```
a=sin(x)^2+cos(x)^2
simplify(a)
```

- Werte ersetzen

```
a=subs(sin(x),x,3*pi/2)
```

- Ausmultiplizieren

```
expand((1+x)^10)
```

- Zerlegung in Faktoren

```
factor(x^10-1)
```

Symbolic Math Toolbox

- Viele Operationen der Analysis sind in Matlab implementiert

```
limit(x*sin(x),x,0)
diff(x^5,x,2)
int(x^5,x)
int(1/x,x,1,2)
```

Symbolic Math Toolbox

- Lösen von Gleichungen

```
solve(x^3+2*x==1)
[a,b]=solve(x+y==1,x-y==-1)
```

- ... und Differentialgleichungen

```
syms y(t)
dsolve(diff(y,2)==-y,y(0)==0)
```

Symbolic Math Toolbox

- Matrizen

```
A=sym('A',[2,2])
inv(A)
eig(A)
```

Symbolic Math Toolbox

- Plotten

```
ezplot(sin(x),[0,pi])
syms y
ezsurf(sin(x)*cos(y),[0,pi,0,2*pi])
```

- Auch benutzbar für anonyme Funktionen

```
ezplot(@(s) sin(s),[0,2*pi])
ezplot(@cos,[0,2*pi])
```

Aufgaben

- Berechne die Millionste Stelle der Eulerzahl e
- Berechne die Integrale von:
  - sin(3x)cos(5x)
  - x ln(x) auf [2,5]
- Berechne den Grenzwert von
  sin(x)(1-cos(2x))/x^3
  für x-> 0.

```
% Vielen Dank für die Aufmerksamkeit

end
```