

Einführung in Matlab

Einheit 1

Jochen Schulz

Georg-August Universität Göttingen 

6. September 2009

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

- MATLAB steht für **Mat**rix **lab**oratory; ursprünglich speziell für Matrizenrechnung entwickelt.
- Entwickelt von Cleve Moler Ende der 70'er in FORTRAN.
- Heutige Version ist in C/C++ programmiert.
- Interaktives System für numerische Berechnungen und Visualisierungen.
- Kein Computer-Algebra-System (Allerdings mittlerweile durch die `symbolic math toolbox` dahingehend erweiterbar)

Vorteile von MATLAB

- High-Level Sprache \Rightarrow . Das Programmieren ist leicht und führt zu schnellen Erfolgen. Sehr geeignet für Prototyping und Debugging.
- Vielfältige Visualisierungsmöglichkeiten.
- MATLAB-Programme sind vollständig portierbar zwischen Architekturen.
- Integration zusätzlicher Toolboxes (Symb. Math T., PDE T., Wavelet T.)
- Ausgereifte Oberfläche.



Matlab online-help :-).



Matlab Guide, D.J. Higham, N.J. Higham, SIAM, 2000,



Introduction to Scientific Computing, C.F. van Loan, Prentice Hall, New Jersey, 1997,



Scientific Computing with MATLAB, A. Quarteroni, F. Saleri, Springer, 2003,



Graphics and GUIs with MATLAB, P. Marchand, O.Th. Holland, Chapman & Hall, 2003, 3. Aufl.



MATLAB 7, C. Überhuber, St. Katzenbeisser, D. Praetorius, Springer 2005.



Using MATLAB, offizielle Handbücher.

- **Launch Pad:** Start von Werkzeugen, Dokumentationen und Demos der verschiedenen Toolboxes
- **Command Window:** Befehlseingabe und Standardausgabe.
- **Workspace:** Ansicht von Variablen und deren Art und Grösse; Ändern der Einträge
- **Grafikfenster:** Grafiken werden normalerweise in separaten Fenstern dargestellt.

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Erste Schritte

- MATLAB als Taschenrechner
(Ergebnis wird in `ans` gespeichert.)

```
>> 1+(sin(pi/2)+ exp(2))*0.5  
ans = 5.1945
```

- Eingabe von (Zeilen-)Vektoren

```
>> x = [1 2 3]
```

- Transponieren und speichern in Variable `b`

```
>> b = transpose(x)
```

Erste Schritte II

- Erzeugen einer Matrix

```
A = [0 2 3 ; 4 5 6; 7 8 9];
```

- Lösen des Gleichungssystems $A \cdot z = b$

```
>> z = A \ b
```

- Probe

```
>> A*z
```

Erste Schritte III

- Berechnen der Determinante von A

```
>> det(A)
```

- Hilfe zu det

```
>> help det
DET      Determinant.
      DET(X) is the determinant of the square matrix X.
      Use COND instead of DET to test for matrix singularity.
```

- Erzeugen einer Einheitsmatrix

```
>> B = eye(3,3)
```

- Matrizenoperationen

```
>> A+B, A-B, A*B, inv(A)
```

- Anwendung von Vektoren

```
>> y = sqrt(x)  
y =  
    1.0000    1.4142    1.7321
```

- Komponentenweise Multiplikation

```
>> y = x.*x  
y =  
    1    4    9
```

- Zeilenvektor mit Werten von 1 bis 100

```
>> a = [1:100];
```

- Berechne $\sum_{j=1}^{100} \frac{1}{j^2}$

```
>> (1./a)*transpose(1./a)  
ans = 1.6350
```

- Alle definierten (globalen) Variablen werden mit seinen Werten im Workspace gespeichert.
- Man kann während einer MATLAB-Sitzung auf alle Variablen im Workspace zugreifen.
- Auskunft über den Inhalt des Workspace gibt `whos` oder `who`.
- Löschen von Variablen : `clear <var>`; `clear` löscht den gesamten Arbeitsspeicher (Workspace).

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

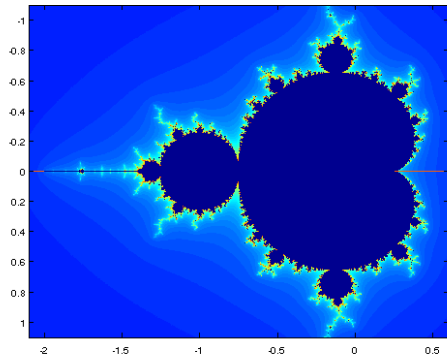
2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Die Mandelbrot-Menge



Die Mandelbrot-Menge ist die Menge von Punkten $c \in \mathbb{C}$ bei denen die Folge $(z_n)_n$, die durch

$$\begin{aligned} z_0 &:= c \\ z_{n+1} &= z_n^2 + c, \quad n \in \mathbb{N} \end{aligned}$$

definiert ist, beschränkt ist.

Ein nichttriviales Beispiel

```
% Mandelbrot set
% MANDEL.m

x = linspace(-2.1,0.6,601);
y = linspace(-1.1,1.1,401);

[X,Y] = meshgrid(x,y);
C = complex(X,Y);

it_max = 50;

Z = C; Anz = zeros(size(Z));

for k = 1:it_max
    Z = Z.^2+C;
    Anz = Anz + isfinite(Z);
    waitbar(k/it_max);
end,
image(x,y,Anz);
title('Mandelbrot Set', 'FontSize',16);
```

Verwendete Befehle

- `linspace(a,b,n)` ist ein Vektor mit n Einträgen der Form $a, a + (b - a)/(n - 1), \dots, b$
- `[X,Y]=meshgrid(x,y)` erzeugt Matrizen

$$X = \begin{pmatrix} x_1 & \dots & x_n \\ & \vdots & \\ x_1 & \dots & x_n \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 & \dots & y_1 \\ & \vdots & \\ y_n & \dots & y_n \end{pmatrix}$$

- `C=complex(X,Y)` erzeugt $C = (C(j, k))_{jk}$ mit $C(j, k) = X(j, k) + i Y(j, k)$

- `B=isfinite(A)` erzeugt eine Matrix B der gleichen Größe wie A . Die Einträge sind 1, wenn der entsprechende Eintrag von B finit ist und 0 sonst.
- `image(x,y,A)` erzeugt eine Grafik auf der Basis des Gitters (x,y) mit Werten A . Durch den entsprechenden Eintrag von A wird die Farbe bestimmt.
- Mit `title` lassen sich Grafiken mit einer Überschrift versehen.
- Durch `for`, `end` wird eine Schleife angegeben (Details später).

- Starten von MATLAB: Eingabe von `matlab &`.
- Verlassen von MATLAB: `quit` oder `exit`
- Editieren alter Eingaben: `↑`, `↓` (wie in Unix)
- aktuelles Verzeichnis: `pwd`
- Bei Definition von Variablen wird der Wert der Variablen ausgegeben.
Mit `;` am Ende wird dies unterdrückt.
- Inhalt des Arbeitsspeichers: `whos` oder `who`

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

- Erzeugen 'per Hand'

```
> b=[ 1 2 4]
b =
     1     2     4
```

- Abfragen der Einträge von b

```
>> b(2)
ans =     2
```

Index \equiv Position im Vektor

Achtung: Indizes beginnen immer mit 1!

Doppelpunkt - Notation

$x : s : z$ erzeugt einen Vektor der Form

$$(x, x + s, x + 2s, x + 3s, \dots, z).$$

```
>> a=2:11
a =
     2     3     4     5     6     7     8     9    10    11

>> c=-2:0.75:1
c =
-2.0000 -1.2500 -0.5000  0.2500  1.0000
```


- `length(a)` gibt die Länge des Vektors a an.
- `linspace(x1,x2,N)` erzeugt den Vektor

$$x_1, x_1 + \frac{x_2 - x_1}{N-1}, x_1 + 2\frac{x_2 - x_1}{N-1}, \dots, x_2$$

der Länge N .

```
>> linspace(1,2,4)
ans =
    1.0000    1.3333    1.6667    2.0000
```

- `logspace(x1,x2,N)` wie `linspace`, nur logarith. Skalierung

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Erzeugen von Matrizen

- Erzeugen 'per Hand'

```
>> B=[ 1 3 4; 5 6 7]  
B =  
     1     3     4  
     5     6     7
```

- Erzeugen von 'Einheitsmatrizen'

```
>> eye(2,3)  
ans =  
     1     0     0  
     0     1     0
```

`eye(n,m)` erzeugt eine $(n \times m)$ - Matrix mit 1 auf der Hauptdiagonalen und 0 sonst.

Erzeugen von Matrizen II

- `zeros(n,m)` bzw. `ones(n,m)` erzeugt eine $(n \times m)$ - Matrix mit 0 bzw. 1 als Einträge.
- Erzeugen von Blockmatrizen

```
>> C = [B zeros(2,2); eye(2,3) eye(2,2)]
```

```
C =
```

1	3	4	0	0
5	6	7	0	0
1	0	0	1	0
0	1	0	0	1

Achtung: Matrizen in einer Zeile müssen dieselbe Zeilenanzahl haben und Matrizen in einer Spalte dieselbe Spaltenanzahl.

Erzeugen von Matrizen III

- `repmat(A,n,m)` erzeugt eine Blockmatrix mit $(n \times m)$ aus A bestehenden Blöcken

```
>> D = repmat(B,1,2)
```

```
D =
```

1	3	4	1	3	4
5	6	7	5	6	7

- `blkdiag(A,B)` erzeugt eine Blockdiagonalmatrix.
- `diag(v,k)` erzeugt eine Matrix der Größe $(n + |k|) \times (n + |k|)$ mit den Einträgen des Vektors v auf der k -ten Nebendiagonalen

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- **Manipulation von Matrizen**
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Beispiel-Matrizen

- Einen Überblick erhält man durch `>> help gallery`
- Ein Beispiel

```
>> E = gallery('moler',4)
E =
     1     -1     -1     -1
    -1      2      0      0
    -1      0      3      1
    -1      0      1      4
```

- Hilfe zur Matrix 'moler' erhält man durch `help private/moler`
- weitere Matrizen: `magic`, `hilb`, `vander`

Zugriff auf Matrizen

```
>> A = [ 1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

Abfragen eines Eintrags

```
>> A(2,1)
ans =
     4
```

Abfrage von Blöcken

```
>> A(2:3,1:2)
ans =
     4     5
     7     8
```

Abfrage einer Zeile

```
>> A(2,:)
ans =
     4     5     6
```

Abfrage mehrerer Zeilen

```
>> A([1 3],:)
ans =
     1     2     3
     7     8     9
```



```
>> A = [ 1 2 3; 4 5 6; 7 8 9]  
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

Löschen einer Zeile

```
>> A(2,:) = []  
A =
```

```
    1    2    3  
    7    8    9
```

Löschen von Spalten

```
> A(:,[1 3]) = []  
A =
```

```
    2  
    5  
    8
```

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Matrizenoperationen

Standard-Matrix Operationen +, -, *

```
>> A = [1 2; 3 4]; B = 2*ones(2,2);
```

Multiplikation

```
>> A*B
```

```
ans =
```

```
     6     6
    14    14
```

Addition

```
>> A+B
```

```
ans =
```

```
     3     4
     5     6
```

Subtraktion

```
>> A-B
```

```
ans =
```

```
    -1     0
     1     2
```

- $A \setminus B$ berechnet die Lösung X von $A * X = B$.
- A / B berechnet die Lösung X von $X * A = B$.
- $\text{inv}(A)$ berechnet die Inverse von A .
- A' oder $\text{ctranspose}(A)$ ergibt die komplex Transponierte von A .
- $A.'$ oder $\text{transpose}(A)$ ist die Transponierte von A .
- A^z berechnet $\underbrace{A * A * \dots * A}_{z\text{-mal}}$ (quadratische Matrizen)
- Die Größe einer Matrix A wird durch $\text{size}(A)$ bestimmt.

Spezielle Operatoren

```
>> A = [1 2; 3 4]; B = 2*ones(2,2);
```

- $C = A.*B$ ergibt C mit $C(i,j) = A(i,j) * B(i,j)$.

```
C =  
    2    4  
    6    8
```

- $C = A./B$ ergibt C mit $C(i,j) = A(i,j)/B(i,j)$.

```
C =  
    0.5000    1.0000  
    1.5000    2.0000
```

- $C = A.B$ ergibt C mit $C(i,j) = B(i,j)/A(i,j)$.

```
C =  
    2.0000    1.0000  
    0.6667    0.5000
```

- $C = A.^B$ ergibt C mit $C(i,j) = A(i,j)^{B(i,j)}$

$C =$

1	4
9	16

Achtung: Bei diesen Operationen müssen A und B dieselbe Dimension haben. Matrizen können durch Skalare ersetzt werden, z.B. $A.^2$.

Skalarprodukt

- Vektoren $a = (a_1, \dots, a_n)$, $b = (b_1, \dots, b_n)$
- Skalarprodukt: ab^t
- Summe der Einträge von a : $(1, \dots, 1)a^t$

Beispiel:

```
>> a=1:100; b=linspace(0,1,100);  
>> a*transpose(b)  
ans =  
    3.3667e+03  
>> ones(1,100)*transpose(a)  
ans =  
    5050
```

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Zwei-Punkt-Randwert-Aufgabe

Suche eine Funktion

$$u : [0, 1] \rightarrow \mathbb{R},$$

so dass

$$\begin{aligned} -u''(x) &= e^x, & x \in (0, 1) \\ u(0) &= u(1) = 0 \end{aligned}$$

Problem: Es kann i.A. keine geschlossene Lösungsdarstellung angegeben werden.

Ausweg: Approximation der Lösung.

Finite Differenzen Verfahren

Diskretisierung: $0 = x_0 < \dots < x_n = 1$ mit $x_i = \frac{i}{n}$

Differenzenquotient:

$$u''(x_i) \sim \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1}))}{h^2}, \quad h := \frac{1}{n}$$

Einsetzen in $-u''(x) = e^x$ ergibt

$$-u(x_{i-1}) + 2u(x_i) - u(x_{i+1})) = h^2 e^{x_i}, \quad i = 1, \dots, n-1$$

Randbedingungen $\Rightarrow u(x_0) = u(x_n) = 0$.

\Rightarrow Lineares Gleichungssystem für $u(x_1), \dots, u(x_{n-1})$.

Diskretes Problem

Setze $z = (z_1, \dots, z_{n-1})^t = (u(x_1), \dots, u(x_{n-1}))^t$.

Löse das Gleichungssystem $Az = F$ mit

$$A := \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{pmatrix}, \quad F := h^2 \begin{pmatrix} e^{\frac{1}{n}} \\ \vdots \\ e^{\frac{n-1}{n}} \end{pmatrix}.$$

Lösung für $n = 21$

- Zerlegung des Intervalls $[0, 1]$

```
>> x=0:(1/21):1
```

- Eliminieren der Randpunkte

```
>> x_i=x(2:21)
```

- Erzeugen der Matrix A (Übungsaufgabe)

Lösung für $n = 21$

- Berechnen der rechten Seite:

```
F=(1/21)^2*transpose(exp(x_i));
```

- Lösen des lin. Gl.

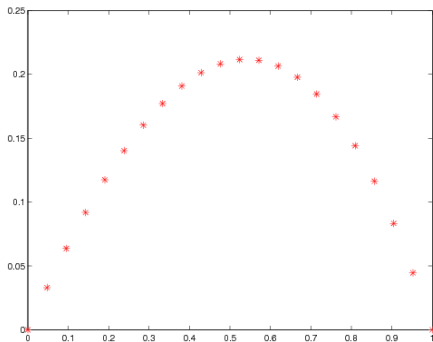
```
z_i=AF;
```

- Zufügen der Werte am Rand

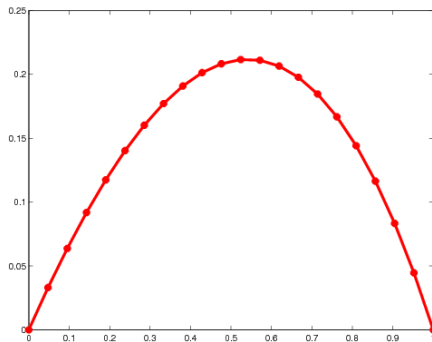
```
>> z=[0; z_i;0];
```

Lösung für $n = 21$

```
>> plot(x,z, 'r*', 'MarkerSize',8)
```



```
>> plot(x,z, 'r*-','LineWidth',3, 'MarkerSize',8)
```



1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Die p -Norm eines Vektors $x = (x_1, \dots, x_n)$

$$\|x\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

kann berechnet werden durch `norm(x,p)` (Default: $p = 2$)

- Die Norm ist definiert für $p \geq 1$.
- $p = \infty$ entspricht der Maximum-Norm

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i|.$$

Seien $A \in \mathbb{C}^{n \times m}$ und $p \geq 1$. Die *Matrixnorm* ist definiert durch

$$\|A\|_p = \sup_{x \in \mathbb{C}^m \setminus \{0\}} \frac{\|Ax\|_p}{\|x\|_p}.$$

- In MATLAB: `norm(A,p)` (Default $p = 2$).
- $p = \infty$ kann charakterisiert werden durch

$$\|A\|_\infty = \max_{1 \leq j \leq m} \sum_{i=1}^n |a_{ij}|, \quad \text{Zeilensummennorm.}$$

Kondition einer quadratischen Matrix A :

$$\text{cond}_p(A) := \|A\|_p \|A^{-1}\|_p.$$

- In MATLAB: $\text{cond}(A,p)$ (Default $p = 2$)
- Es gilt $\text{cond}_p(A) \geq 1$.
- Die Kondition mißt die Empfindlichkeit der Lösung x von $Ax = b$ gegenüber Störungen von A und b .
- Ist $\text{cond}_p(A) \gg 1$, so ist die Matrix beinahe singular. Die Matrix ist *schlecht konditioniert*.

- Vektornormen für $x = (1/100)(1, 2, \dots, 100)$

```
>> x = (1:100)/100; [norm(x,1) norm(x,2) norm(x,inf)]  
ans =    50.5000    5.8168    1.0000
```

- Matrixnorm für die Hilbert-Matrix $H = (\frac{1}{i+j-1})_{ij}$

```
>> H = hilb(10); [norm(H,1) norm(H,2) norm(H,inf)]  
ans =    2.9290    1.7519    2.9290
```

- Kondition der Hilbert-Matrix

```
>> H = hilb(10); [cond(H,1) cond(H,2) cond(H,inf)]  
ans =  
    1.0e+13 *  
    3.5354    1.6025    3.5354
```

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Lineare Gleichungssysteme

Seien $A \in \mathbb{C}^{n \times n}$ und $b \in \mathbb{C}^n$. Das lineare Gleichungssystem

$$Ax = b$$

wird in MATLAB gelöst durch $x=A \backslash b$.

```
>> x = ones(5,1); H = hilb(5); b = H*x; y = (H\b)'  
y =  
    1.0000    1.0000    1.0000    1.0000    1.0000
```

Warnung: Benutze nie $x=\text{inv}(A)*b$, da das Berechnen von A^{-1} sehr aufwendig sein kann.

Was bedeutet $A \setminus b$?

MATLAB berechnet die LU-Zerlegung von A (Gaussverfahren), d.h. es sucht eine obere Dreiecksmatrix U und eine untere Dreiecksmatrix L mit Einsen auf der Diagonalen, so dass $PA = LU$ gilt (P Permutationsmatrix). Dann wird das LGS durch Rückwärts- und Vorwärtseinsetzen gelöst.

```
>> [L,U,P]=lu(hilb(5)); norm(P*hilb(5)-L*U)
ans = 2.7756e-17
```

Inverse, Determinante

- Berechnung der Inversen

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

```
A =
```

1	1	1
1	2	3
1	3	6

```
>> X=inv(A)
```

```
X =
```

3	-3	1
-3	5	-2
1	-2	1

- Berechnung der Determinante

```
>> det(A)
```

```
ans = 1
```


Pseudoinverse

Berechnung der (Moore-Penrose) Pseudoinversen X von A (A singulär), d.h. X genügt

$$AXA = A, XAX = X, (XA)^* = XA, (AX)^* = AX.$$

```
>> pinv(ones(3,3))  
ans =  
    0.1111    0.1111    0.1111  
    0.1111    0.1111    0.1111  
    0.1111    0.1111    0.1111
```

1 Streifzug durch MATLAB

- Einleitung
- Erste Schritte
- Beispiele

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen
- Eine Anwendung

3 Numerische Lineare Algebra

- Normen
- Lösen linearer Gleichungssysteme
- Bestimmung von Eigenwerten

Sei $A \in \mathbb{C}^{n \times n}$. $\lambda \in \mathbb{C}$ ist Eigenwert von A , falls ein Vektor $x \in \mathbb{C}^n$ ungleich 0 existiert, so dass $Ax = \lambda x$ gilt. x heißt Eigenvektor.

- $x = \text{eig}(A)$ berechnet die Eigenwerte von A und schreibt sie in den Vektor x .
- $[V, D] = \text{eig}(A)$ D ist eine Diagonalmatrix mit den Eigenwerten auf der Diagonalen. Die Spalten von V bilden die zugehörigen Eigenvektoren.

Weitere Zerlegungen

- Mittels $[Q,R]=\text{qr}(A)$ wird zu einer $m \times n$ - Matrix A eine Zerlegung $A = QR$ erzeugt, wobei Q eine unitäre $m \times m$ -Matrix ist und R eine obere $m \times n$ Dreiecksmatrix ist.
- Eine Singulärwertzerlegung $A = U\Sigma V^*$ wird durch $[U,S,V]=\text{svd}(A)$ berechnet. Dabei ist $\Sigma \subset \mathbb{C}^{m \times n}$ eine Diagonalmatrix und $U \subset \mathbb{C}^{m \times m}$, $V \subset \mathbb{C}^{n \times n}$ sind unitäre Matrizen.
- Eine Cholesky-Zerlegung $A = R^*R$ zu einer hermiteschen, positiv definiten Matrix A wird durch $R=\text{chol}(A)$ berechnet. R ist eine obere Dreiecksmatrix mit reellen, positiven Diagonalelementen.

- LGS können auch mit Hilfe iterativer Verfahren gelöst werden, z.B. gmres, pcg, bicgstab.
- Ist $A \in \mathbb{C}^{n \times m}$, $n \neq m$, so ergibt $A \backslash b$ für $n > m$ (der überbestimmte Fall) die Least-Square Lösung, d.h. der Ausdruck $\text{norm}(A*x-b)$ wird minimiert. Ist $n < m$ (der unterbestimmte Fall) so wird eine Grundlösung berechnet.