

Übung zur Vorlesung Lineare Algebra/Analytische Geometrie I

Übung 7 : MATLAB - Crashkurs

1. Man erzeuge in MATLAB die Matrizen $A = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{9 \times 9}$,
 $B = \text{diag}(i^2)_{i=1}^9 \in \mathbb{R}^{9 \times 9}$ und einen Zufallsvektor $b \in \mathbb{R}^9$. Für diese sollen

- die Matrix $A + 2B$,
- die Summe aller Elemente von A ,
- die Matrix A^{-1} ,
- die LU-Zerlegung von A
- die Cholesky-Zerlegung von A ,
- die Lösung von $Ax = b$

berechnet werden.

2. Visualisierung unter MATLAB:

- Man visualisiere die Funktion $f : \mathbb{R} \mapsto \mathbb{R}$, $f(x) = \sin x$ in $[0, 2\pi]$.
- Man füge Labels, Titel usw. hinzu und speichere den Plot in einem Vektorformat ab

Lösung der Aufgaben

1. Aufbau der Matrix A

1.Weg: „Einzelne“ Eingabe aller von Null verschiedenen Matrixeinträge über Indizes

```
A=zeros(n,n); % n x n Matrix mit Nullen
for i=1:n
    A(i,i)=2; % Hauptdiagonale
    if (i<n)
        A(i,i+1)=-1; % Nebendiagonalen
        A(i+1,i)=-1;
    end
end
```

2. Weg: Nutzen der Beziehung $A = 2I - V - V^T$, wobei I die Einheitsmatrix und V die Blockmatrix

$$V = \begin{bmatrix} \mathbf{0} & I_{n-1} \\ 0 & \mathbf{0} \end{bmatrix}$$

ist.

```

Hd=2*eye(n); % eye(n) ist die Einheitsmatrix
v=zeros(n-1,1);
V=-[ v, eye(n-1)
      0, v' ]; % Aufbau der Blockmatrix
A=Hd+V'+V; % V' ist die zu V transponierte Matrix

einfacher: V=diag(ones(n-1,1),1); A=Hd-V'-V;

```

Aufbau der Diagonalmatrix B . Dazu hat MATLAB den Befehl `diag`:

```

v=zeros(n,1);
for i=1:n
    v(i)=i^2;
end
B=diag(v);

```

Hinweis: Die Zählschleife kann mit `v(1:n)=(1:n).^2`; umgangen werden.

Vektor b : Ein zufälliger Vektor wird mit `rand(n,1)` erzeugt. Um auf einen normierten Vektor zu gelangen, muß noch durch die Norm dividiert werden:

```

b1=rand(n,1); % n x 1 Matrix mit zuf. Eintraegen
z=norm(b1) % Norm von b1
b=b1/z

```

- (a) Die Addition von Matrizen ist denkbar einfach $A+2*B$.
- (b) Entweder man schreibt eine Doppelschleife (langsam !) oder erinnert sich an die Beziehung $e'*A*e$, also `e=ones(9,1)`; $e'*A*e$.
Geht es auch mit der Funktion `sum(...)`? (siehe `help sum`)
- (c) Statt A^{-1} besser `inv(A)` verwenden.
- (d) Der Befehl `[L,U]=lu(A)` zerlegt die Matrix A in das Produkt aus einer unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix U .
`[L,U,P]=lu(A)` liefert zusätzlich eine Permutationsmatrix mit $P*A=L*U$.
- (e) `inv(A)` berechnet die Inverse von A .
- (f) `x=A\b` löst das LGS $Ax = b$.

2. Visualisierung unter MATLAB: Um eine Funktion $f : [a, b] \mapsto \mathbb{R}$ zu zeichnen, generiert man sich eine möglichst feine Unterteilung $a = x_1 < x_2 < \dots < x_n = b$ des Intervalls $[a, b]$. Die entsprechenden Gitterpunkte x_j schreibe man in einen Vektor $x = [x_j]_{j=1}^n$ und berechne für jeden Punkt x_j den Wert $f(x_j)$. Sei $f = [f(x_j)]_{j=1}^n$. Dann kann man mittels `plot(x,f)` die Funktion $f : [a, b] \mapsto \mathbb{R}$ graphisch darstellen.

```

x=0:0.01:2*pi; % 0 bis 2*pi mit geg. Schrittweite, oder:
x=linspace(0,2*pi,629) % mit geg. Anzahl von Punkten
y=sin(x); % Berechnung der Funktionswerte
plot(x,y) % Plotten

```

```

xlabel('x'); % Labels usw.
ylabel('sin(x)')
title('Plot von sin(x) im Intervall [0,2Pi]');

```

```

print -depsc sin.eps % Ausgabe als EPS (driver: eps-color)

```

Anhang zu Übung 7: MATLAB

Eigenes Arbeitsverzeichnis anlegen (z.B. `mkdir matlab`), um dort später Programme zu speichern. Interpretieren Sie die Bildschirm-Ausgaben nach Eingabe der folgenden MATLAB-Anweisungen. Achten Sie auf unterschiedliche Eingabemöglichkeiten (Bedeutung von `,` `;` `:`) und deren Auswirkungen aufs Ergebnis. — Notieren Sie sich ggf. die Bedeutung einzelner Anweisungen.

1. Eingabe, Verwendung von Variablen, Matrizen und Vektoren

```
> pwd
> cd matlab
> 3*4
> pi/2
> A = [1 2 3 4 5 6 7 8 9]
> A = [1 2 3; 4 5 6; 7 8 9]
> M = magic(6)
> B = [4, 5, 6.1], x=[ -1.3 sqrt(3) (1+2+3)^3 ]
> E = ones(3), F = zeros(3)
> e = ones(1,3), f = zeros(3,1)
> G = 3*diag(e) - 2*E
> E * G, E .* G, p1 = e*e', p2 = e'*e
> x(8) = -x(2); y=x';
```

2. Ausgabe von Ergebnissen in unterschiedlichem Format

```
> display(3*4) Standardausgabeformat mit Variablenname
> disp(3*4) nur Werte ohne Variablenname
> disp(' A B'), disp(rand(4,2))
> fprintf('Die Zahl pi lautet: %20.16f...\n',pi) mit Textformatierung
```

3. Verschiedene Zahlendarstellungen

```
> x, y
> format long
> x
> format long e
> x
> format +
> x
> format short
> omega = sqrt(-3)
> z1 = 10^20
> z2 = 1e20
> s=1/0, t=1/s, u=t*s
```

4. Verwendung des Doppelpunktes (Zähl-Intervalle, Indexbereiche)

```
> n=10
> 1:n
> 1:2:n
> 2:2:n
> linspace(0,10,6), cos(linspace(0,pi,5))
> y = x(3:5)
> y(4:6)=x(1:3)
> r = [10 11 12]
> q = [13:16]', p = [13;14;15;16]
> B = [ A; r ]
> C = [ B p ]
> C = C(:,4:-1:1)
> A = C(1:2,:)
> size(B)
```

```

> A = C(:,1:3)
> [n,m] = size(A)
> v = A(:), w=reshape(A,1,m*n)
> x = pi*(0:1/2:2) oder: x=linspace(0,2*pi,5)
> s = sin(x), c = cos(x)

```

5. Schleifen, Verzweigungen, Rechnen mit Matrizen

```

> % Tridiagonalmatrix, Variante a)
> n=5;
> A=zeros(n);      % n x n Matrix mit Nullen
> for i=1:n
>   A(i,i)=2;      % Hauptdiagonale
>   if (i<n)
>     A(i,i+1)=-1; % Nebendiagonalen
>     A(i+1,i)=-1;
>   end
> end
> disp(A)
> % Tridiagonalmatrix, Variante b)
> V=diag(ones(n-1,1));           Matrix aus verschobener Diagonale
> A=2*eye(n)-V-V';              eye(n) ist die n x n-Einheitsmatrix
> disp(A)
> E=eig(A)                       Eigenwerte von A
> [V,D]=eig(A)                   Eigenvektoren V1, ..., Vn und Diagonalmatrix der EW
> diag(D)
> diag(E)
> W=A*V-V*D                      sollte bei exakter Rechnung Null sein
> max(abs(W))
> max(max(abs(W)))
> c=rand(n,1)                    Zufallsvektor
> b=A*c
> % Loese lineares Gleichungssystem Ax=b :
> x=A^(-1)*b, oder: x=inv(A)*b,   aber besser:
> x=A\b oder dasselbe: x=(b'/A')'

```

6. Eigene Funktionen

```

> edit tridiag.m                ein sog. „M-file“ anlegen (auch mit belieb. Editor)

```

Inhalt der Datei tridiag.m:

```

function T=tridiag(n,a,b,c)
% tridiag(n,a,b,c) - erzeugt eine Tridiagonalmatrix mit
% konstanten Eintraegen:
% auf der unteren Subgonalen:  a
% auf der Hauptdiagonalen:    b
% auf der oberen Subdiagonalen: c
V=diag(ones(n-1,1));
T=b*eye(n)+c*V+a*V';

```

Nutzung der Funktion tridiag:

```

> help tridiag                  zeigt den selbst eingebauten Hilfetext an
> A=tridiag(n,2,-1,-1);

```

7. Grafische Darstellungen

```
> x=linspace(0,2*pi); c=cos(x); s=sin(x);
> plot(c)
> plot(x,c)
> subplot(2,1,1),plot(x,c),title('cos')
> subplot(2,1,2),plot(x,s),title('sin')
> subplot(1,1,1),plot(c,s)
> figure(2)
> plot(c), hold on, plot(s), hold off
> plot(x,c,x,s)
> plot(x,c,'o',x,s,':')
> Y = [ c; s ]; plot(x,Y)
> title('Funktionen')
> xlabel('x-Achse')
> ylabel('f(x)')
> grid
> x=-8:0.5:8; y=x';
> X=ones(size(y))*x; Y=y*ones(size(x));
> R=sqrt(X.^2+Y.^2)+eps;
> Z=sin(R)./R;
> mesh(Z);
> view(45,45)                                3D-Ansicht (Winkel in Kugelkoordinaten)
> view(0,90)
> view(45,90)
> view(0,0)
> view(3)                                    (=Standardwerte für 3D-plot: (-37.5,30))
```

8. Hilfsfunktionen

```
> diary                                     Ein-/Ausschalten der Protokollierung (in eine Datei)
> who                                       Liste aller Variablen
> whos                                     Variablenliste mit Speicherbedarf
> save temp                                alle Variablen retten als 'temp.mat'
> clear; whos                              alle Variablen löschen
> load temp A; whos                        nur Matrix A wieder einlesen
> load temp                                 alle geretteten Variablen wiederherstellen
> who
> save tmpmat A                             nur die Matrix A speichern
> save a.dat A -ascii                       dasselbe, aber lesbar
> dir
> clear A; who
> load a.dat; who                            Matrix A heißt jetzt a
```

Hilfe allgemein und zu einzelnen Kommandos:

```
> help
> help [
> help punct
> help inv
> help det
> help diag
> ...
```

9. Mitgelieferte Demos

```
> intro
> demo
```