ETHZ, BSc RW/CSE

Probeprüfung

Numerische Mathematik für RW WS 05/06

Prof. R. Jeltsch

Wichtige Hinweise

- Prüfungsdauer: 105 Minuten
- Zugelassene Hilfsmittel: 10 handgeschriebene A4 Blätter, MATLAB, Vorlesungsskript auf dem Rechner.
- Jede Aufgabe auf einer neuen Seite beginnen und jedes Blatt mit Namen und Aufgabennummer deutlich beschriften.
- Jede MATLAB-Programmdatei in der ersten Zeile mit einem Kommentar versehenen, der Namen und (Teil)Aufgabennummer enthält, Bsp:
  `% Hans Muster, Aufgabe 3f)`
- Ausdrucken der MATLAB-Programme während der Prüfung oder am Prüfungsende mit dem Skript
  print_m_files.bsh
  (Fasst alle *.m -Files im aktuellen Verzeichnis im File m_file_summary zusammen und schickt dieses an den Drucker.)
- Nicht durch Hardwareversagen entstandener Datenerlust ist vom Prüfungskandidaten zu verantworten.

Viel Glück!

— • —

1. Das lineare Gleichungssystem

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

mit $\boldsymbol{A} \in \mathbb{R}^{n,n}$ strikt diagonal dominant und $\boldsymbol{b} \in \mathbb{R}^n$, soll mit der folgenden Fixpunktiteration gelöst werden:

$$\boldsymbol{x}^{k+1} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{R})\boldsymbol{x}^k + \boldsymbol{D}^{-1}\boldsymbol{b}.$$

Dabei ist $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{R}$ die additive Zerlegung der Matrix $\boldsymbol{A}$ in die Diagonalmatrix $\boldsymbol{D} = diag(\boldsymbol{A})$, die untere Dreiecksmatrix $\boldsymbol{L}$ sowie die obere Dreiecksmatrix $\boldsymbol{R}$.

a) Sei $\boldsymbol{e}^k := \boldsymbol{x}^k - \boldsymbol{x}$ der Fehler im k-ten Schritt. Leite (von Hand) die Matrix $\boldsymbol{B} \in \mathbb{R}^{n,n}$ her, so dass für die Fehler gilt:

$$\boldsymbol{e}^{k+1} = \boldsymbol{B}\boldsymbol{e}^k.$$

Tipp: Verwende, dass für die Lösung $\boldsymbol{x}$ die Fixpunktgleichung erfüllt ist.

b) Zeige, dass die Fixpunktiteration in $||\,||_\infty$ kontraktiv ist. (Verwende, dass $\boldsymbol{A}$ strikt diagonal dominant ist.)

2. Die Matrix $\boldsymbol{A} \in \mathbb{R}^{n,n}$ sei von der Form

$$\boldsymbol{A} = \begin{pmatrix} * & * & * & \dots & * & * & * \\ * & * & * & \dots & * & * & * \\ 0 & * & * & \dots & * & * & * \\ 0 & 0 & * & \dots & * & * & * \\ \vdots & & \ddots & \ddots & * & * & * \\ 0 & & & & * & * & * \\ * & 0 & \dots & & 0 & * & * \end{pmatrix}. \tag{2}$$

(* bezeichnet ein Matrixelement $\neq 0$.)

a) Schreibe eine MATLAB-Routine

`function[H,Gi,Gj,rho]=Givenstransform(A),`

welche eine Matrix der Form (2) mit Givensrotationen auf obere Hessenbergform bringt.
Dabei ist `H` eine obere Hessenbergmatrix und `Gi(k),Gj(k),rho(k)` das Tripel zur Speicherung der k-ten Transformationsmatrix (siehe Skript: Konvention für Givens-Rotationen).
Tipp: Verwende Matlabfunktion `planerot`.

b) Überprüfung die Form von `H` aus Aufgabe a). Schreibe dazu ein Testprogramm, welches eine Matrix der Form (2) erzeugt, die Routine aus a) darauf anwendet und die Matrixstrukturen von $\boldsymbol{A}$ und $\boldsymbol{H}$ graphisch darstellt.

3. Betrachte die vier gekoppelten chemischen Reaktionen:

$$\begin{aligned} B &\xrightarrow{1.5} A \\ C &\xrightarrow{3} A \\ A + A &\xrightarrow{20} A + C \\ A + C &\xrightarrow{\lambda} B + C, \quad \lambda > 0 \end{aligned}$$

Das führt auf die Differentialgleichungen

$$\begin{aligned} \text{A:} \quad & y_1' = +1.5y_2 + 3y_3 - 20y_1^2 - \lambda y_1 y_3, && y_1(0) = 0.5 \\ \text{B:} \quad & y_2' = -1.5y_2 \qquad\qquad + \lambda y_1 y_3, && y_2(0) = 0.5 \\ \text{C:} \quad & y_3' = \qquad -3y_3 + 20y_1^2 \qquad , && y_3(0) = 0. \end{aligned}$$

Vom diesem System soll der stationäre Zustand berechnet werden.

a) Da für alle Zeiten

$$y_3 = 1 - y_2 - y_1$$

gilt, kann das Problem auf das Lösen eines zweidimensionalen Gleichungssystems zurückgeführt werden.
Leite von Hand das Gleichungssystem

$$\vec{f}(y_1, y_2, \lambda) = \vec{0} \tag{3}$$

her.

b) Für einen festen Parameter $\lambda > 0$ soll der stationäre Zustand $[y_1(\lambda),\ y_2(\lambda)]$ mit dem Newtonverfahren bestimmt werden.
Leite von Hand die Newtoniteration

$$\begin{pmatrix} y_1^{k+1} \\ y_2^{k+1} \end{pmatrix} = \vec{F}(y_1^k, y_2^k, \lambda)$$

her.

c) Implementiere eine MATLAB-Routine

```
function[y]=Newton_solve(y0,lambda,tol,maxit),
```

welche zu gegebenem Parameter λ die Lösung von (3) berechnet. Verwende dazu die Iterationsvorschrift aus b).
`y_0`: Startvektor
`tol`: Abbruchkriterium bzgl. Änderung der Lösung
`maxit`: Abbruchkriterium bzg. maximaler Anzahl Iterationsschritte

d) Schreibe eine MATLAB-Routine

```
function visualize_solution(),
```

welche die Kurven $y_1(\lambda)$ und $y_2(\lambda)$ graphisch darstellt. Wähle dazu $\lambda = [1, 1.5, ..., 20]$ und berechne für jeden dieser Werte die Lösung. Verwende als Startwert $[y_1,\ y_2] = [0.5,\ 0.5]$.

Beispiellösung 9

1. a)

$$\begin{aligned} \boldsymbol{e}_{k+1} &= \boldsymbol{x}_{k+1} - \boldsymbol{x} = \boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{R})\boldsymbol{x}^k + \boldsymbol{D}^{-1}\boldsymbol{b} - \boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{R})\boldsymbol{x} + \boldsymbol{D}^{-1}\boldsymbol{b} \\ &= \boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{R})\boldsymbol{e}^k. \\ &\Rightarrow \boldsymbol{B} = \boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{R}) \end{aligned}$$

b)

$$B_{ij} = \begin{cases} -A_{ij}/A_{ii}, & i \neq j, \\ 0 & i = j \end{cases}$$

$$||\boldsymbol{B}||_\infty = \max_i \sum_j |B_{ij}| = \max_i \sum_{j, i\neq j} \frac{|A_{ij}|}{|A_{ii}|}$$

Dies ist < 1, da $\boldsymbol{A}$ strikt diagonal dominant ist. Deshalb gilt:

$$||\boldsymbol{e}_{k+1}||_\infty = ||\boldsymbol{B}\boldsymbol{e}_k||_\infty \leq ||\boldsymbol{B}||_\infty ||\boldsymbol{e}_k||_\infty < ||\boldsymbol{e}_k||_\infty$$

2. a)

```
function[H,Gi,Gj,rho]=Givenstransform(A)

%dimension der Matrix A
n=size(A,1);

%Initialisierung
Gi=zeros(n-1,1);
Gj=zeros(n-1,1);
Grho=zeros(n-1,1);


for i=2:n-1

  j=n;

% Berechnung von gamma, sigma, rho
A([i,j],i-1)
G=planerot(A([i,j],i-1));
gamma=G(1,1);
```

Bitte wenden!

```
sigma=G(1,2);
 if gamma == 0
   rho = 1;
 else
   if abs(sigma)< abs(gamma);
     rho = sign(gamma)*sigma/2;
   else
     rho =  2*sign(sigma)/gamma ;
   end
 end

 Gi(j-1)=i;
 Gj(j-1)=j;
 Grho(j-1)=rho ;

 %Update der Matrix A
 A([i j],i-1:end)=G*A([i j],i-1:end)
end
H=A;
```

b)
```
function[]=test()
n=10;
A=triu(rand(n,n))+diag(rand(n-1,1),-1);
A(n,1)=rand(1,1);
figure;
spy(sparse(A))
figure
[H,Gi,Gj,rho]=Givenstransform(A);
[I,J,h]=find(abs(H)>1e-14)
M=sparse(I,J,h)
spy(M)
```

3. a) siehe c)

b) Siehe c)

c)
```
function[y]=Newtion_solve(y0,lambda,tol,maxit)


a=1.5;
b=3;
c=20;
```

Siehe nächstes Blatt!

```
f=@(x) [a*x(2)+b*(1-x(1)-x(2))-c*x(1)^2-lambda*x(1)*...
        (1-x(1)-x(2));...
        -a*x(2)+lambda*x(1)*(1-x(1)-x(2))];

J=@(x)[-b-2*c*x(1)-lambda*(1-x(1)-x(2))+lambda*x(1),...
       a-b+lambda*x(1);...
      lambda*(1-x(1)-x(2))-lambda*x(1),-a-lambda*x(1)];

 for it=1:maxit
   y1=y0-J(y0)\f(y0);
   if norm(y1-y0)<tol*norm(y0)
    y=y1;
    return
   end
   y0=y1;
 end
 y=y0;
```

d)
```
function visualize_solution()
y0=[0.5;0.5];
tol=1e-6;
maxit=100;
lambda=1:0.5:20;
for lambda_nr=1:length(lambda)
 y(lambda_nr,:)=Newton_solve(y0,lambda(lambda_nr),tol,maxit);
end
plot(lambda,y(:,1),'g',lambda,y(:,2),'r')
xlabel('\lambda')
legend('y_1(\lambda)','y_2(\lambda)')
```