Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Wintersemester 2014/2015

Numerische Lineare Algebra - Matlab-Blatt 5
Lösung

(Besprechung in den MATLAB-Tutorien in KW 51/2)

Aufgabe 10 *(Numerische Berechnung der 2-Norm für Matrizen)* (10 Punkte)

Berechnen Sie näherungsweise die 2-Norm für 3×3-Matrizen mit Hilfe der Formel

$$\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2 \,.$$

Um das Supremum zu bestimmen diskretisieren wir die Einheitssphäre mit Kugelkoordinaten und bestimmen das Maximum über die diskrete Menge. Laden Sie dazu das Skript `mainNorm` und die Funktion `norm2` herunter und gehen Sie wie folgt vor:

(i) Vervollständigen Sie die MATLAB-Funktion `norm2`, die als Input eine 3×3-Matrix und einen Parameter $n \in \mathbb{N}$ erhält und $\max_{x \in D_n} \|Ax\|_2$ ausgibt. Die Menge D_n ist hierbei gegeben durch $D_n = M_n \cup \{(0,0,-1),(0,0,1)\}$ mit

$$M_n := \left\{(x,y,z) \in \mathbb{R}^3 : x = \sin\tfrac{\pi k}{n}\cos\tfrac{\pi \ell}{n},\ y = \sin\tfrac{\pi k}{n}\sin\tfrac{\pi \ell}{n}, z = \cos\tfrac{\pi k}{n};\ k = 1,\dots,n-1;\ \ell = 0,\dots,2n-1\right\} .$$

Gehen Sie dabei wie folgt vor:

- Speichern Sie alle Punkte in D_n in einer $n \times 3$-Matrix `points` (Zeilen 9).
- Berechnen Sie `b` = (`A*points`)' (Zeile 15). (In jeder Zeile von `b` steht jetzt Ax für einen Punkt $x \in D_n$.)
- Bestimmen Sie für jede Zeile von `b` die euklidische Norm (verwenden Sie dazu nicht die Matlab-Funktion `norm`) und schließlich das Maximum über alle Normen (Zeilen 20f).

(ii) Ergänzen Sie das Skript `mainNorm`, das den relativen Fehler

$$\frac{\|A\|_2 - \texttt{norm2(A,2\^{}k)}}{\|A\|_2}$$

für $k = 1,\dots,10$ berechnet (Zeile 13) . Zur Berechnung von $\|A\|_2$ dürfen Sie `norm(A,2)` verwenden. Stellen Sie den Fehler graphisch auf einer doppelt logarithmischen Skala dar (Abszisse = n-Werte). (Zeilen 17-20). Veranschaulichen Sie an der Graphik, dass die Formel

$$\frac{\|A\|_2 - \texttt{norm2(A,n)}}{\|A\|_2} \leq \frac{\pi^2}{4} \cdot \left(\frac{1}{n}\right)^2$$

gilt (Zeile 22-25). Achten Sie auf saubere Achsenbeschriftungen, Titel und Legenden.

Lösung:

(i) Die Funktion `norm2.m`

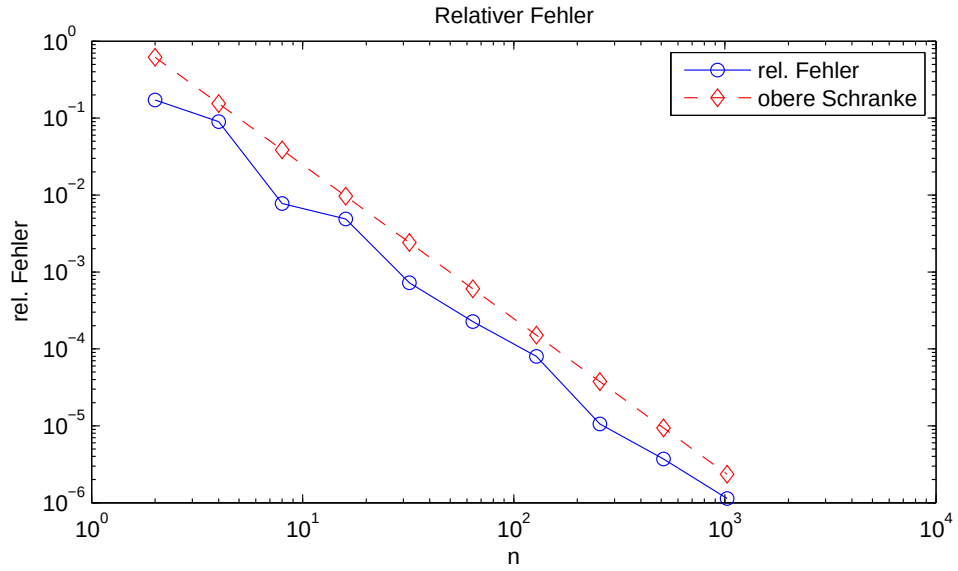
```
1  function val = norm2(A,n)
2
3  % Wertebereich der Indizes
4  k=(1:n-1)';
5  l=(0:2*n-1);
6
```

```
% Berechnung der x,y,z Werte an den Gitterpunkten (Achtung x,y,z sind Matrizen)
x=sin(pi*k/n)*cos(pi*l/n);
y=sin(pi*k/n)*sin(pi*l/n);
z=cos(pi*k/n)*ones(1,2*n);

% Aufstellen von D_n mit x-Wert in der 1.Spalte, y-Wert in der 2.Spalte, usw
points=[0 0 -1;0 0 1;x(:),y(:),z(:)];

% Berechnung von [x,y,z]*A'
b=points*A';

% Berechnung der Norm der Zeilen von b
% Berechne die Norm f r jede Zeile von b und bestimme das Maximum der Normen
val=max(b(:,1).^2+b(:,2).^2+b(:,3).^2);
val=sqrt(val);

% Zusatz: Punktegitter auf der Sphaere
% plot3(D_n(:,1)',D_n(:,2)',D_n(:,3)','.')
```

(ii) Das Skript `mainNorm.m`

```
clc, clear all, close all

A=hilb(3);

%*** exakte Norm
norm_exakt=norm(A,2);

%*** Berechne den Fehler fuer n=1,...,10
error = zeros(10,1);
n = 2.^(1:10);
for k=1:length(n)
    error(k)=1-norm2(A,n(k))/norm_exakt;
end

%*** Plotten des Fehlers
figure(1)
loglog(n,error,'-o')
hold on

%*** ueberpruefe die Ungleichung
y = pi^2/4./(n.^2);
loglog(n,y,'--dr')
hold off

title('Relativer Fehler')
xlabel('n')
ylabel('rel. Fehler')
legend('rel. Fehler', 'obere Schranke')
```

Wir erhalten das folgende Ergebnis:



Wir sehen, dass der relative Fehler unterhalb der oberen Schranke verläuft, somit ist die gegebene Ungleichung erfüllt.

Aufgabe 11 *(Richardson-Iteration)* (10 Punkte)

Das Richardson-Verfahren zum iterativen Lösen von linearen Gleichungssystemen ist gegeben durch

$$x_{k+1} = x_k - \omega\,(Ax_k - b) = (\mathrm{I} - \omega A)\,x_k + \omega\,b \quad (k = 0, 1, ...)$$

mit dem Relaxionsparameter $\omega \in \mathbb{R}$, der Matrix $A \in \mathbb{R}^{n\times n}$, der rechten Seite $b \in \mathbb{R}^n$ und einem Startwert $x_0 \in \mathbb{R}^n$. In dieser Aufgabe bestimmen wir numerisch den optimalen Parameter ω_{opt}.

(i) Schreiben Sie eine MATLAB-Funktion `x=richardson(A,b,x0,omega,N)`, die die Lösung des Linearen Gleichungssystems $Ax = b$ mit dem Richardson-Verfahren berechnet und die Näherungslösung nach N Schritten `x`$= x_N$ zurückgibt.

(ii) Laden Sie sich die das MATLAB-Skript `mainRichardson.m` von der Homepage. Vervollständigen Sie das Skript wie folgt.

- Berechnen Sie die exakte Lösung `x_exact` des LGS mit dem '\'-Operator von MATLAB (Zeile 11).
- Initialisieren Sie den Vektor `omega` mit 200 äquidistanten Werten von 0.25 bis 0.45 (Zeile 16).
- Berechnen Sie für jeden Wert von `omega` die Lösung x_{20} des LGS mit dem Richardson-Verfahren mit $N = 20$ Schritten und den Fehler $\|$`x_exact` $- x_{20}\|_2$. Zur Berechnung des Norm dürfen Sie den MATLAB-Befehl `norm` verwenden (Zeilen 21-22).
- Plotten Sie den Fehler über `omega` und beschriften Sie die Achsen (Zeile 26-29). Was beobachten Sie?
- Lassen Sie sich den optimalen Wert von `omega` ausgeben, d.h. den Wert von `omega`, für den der Fehler $\|$`x_exact` $- x_{20}\|_2$ minimal wird (Zeilen 32-33).
  (Verwenden Sie hierzu den `min`-Befehl von MATLAB und lesen sie in der MATLAB-Hilfe wie man Index des minimalen Elements erhält.)

Lösung:

(i) Die Funktion `richardson.m`

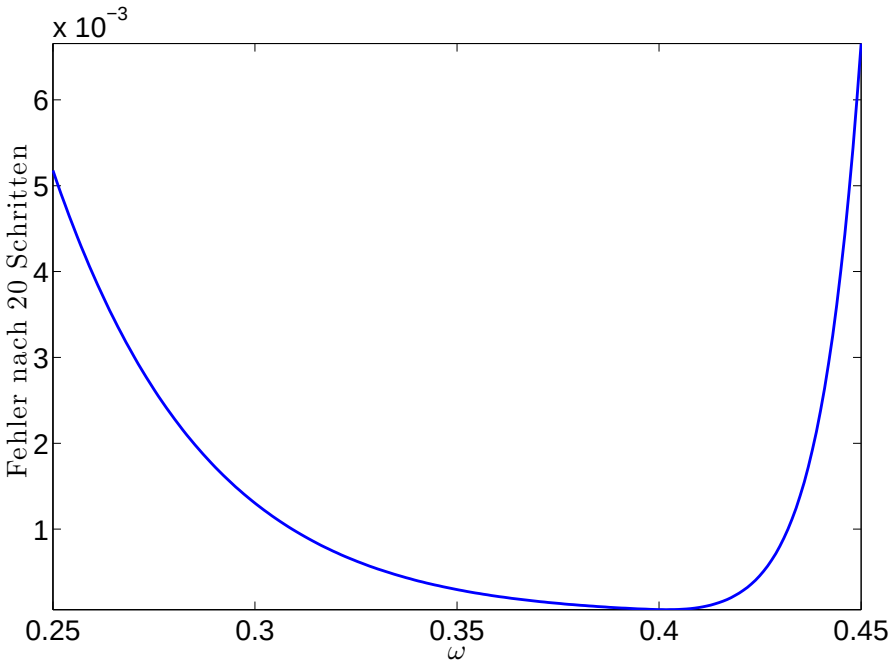
```
function x = richardson(A, b, x0, omega, N)
%*** startwert setzen
x = x0;
for k=1:N
  %*** Rekursion berechnen
  x = x - omega*(A*x-b);
end
```

(ii) Das Skript `mainRichardson.m`

```
clear all
clc, clear all, close all

%*** LGS definieren
n = 3;
A = [2 -1 1; -1 2 -1; 1 -1 2];
b = [2; 0; 2];


%*** LGS exakt loesen
x_exact = A \ b;

%*** Parameter fuer die Rekursion setzen
x0 = zeros(n,1);
N  = 20;
omega = linspace(0.25, 0.45, 200);

%*** fuer jeden Parameter das LGS loesen und relativen Fehlerberechen
err = zeros(length(omega),1);
for k=1:numel(omega)
    x_rich  = richardson(A, b, x0, omega(k), N);
    err(k)  = norm(x_rich - x_exact,2);
end
```

```
24
25 %*** Fehler ueber omega plotten
26 figure(1)
27 plot(omega,err,'Linewidth',2)
28 xlabel('$\omega$','interpreter','latex', 'fontsize',20)
29 ylabel('Fehler nach 20 Schritten','interpreter','latex', 'fontsize',20)
30 set(gca,'fontsize',20)
31 axis tight
32 %*** optimalen Parameter ausgeben lassen
33 [~,idx] = min(err);
34 omega_opt = omega(idx)
```

- Erklärung Zeile 33-34: Der `min`-Befehl liefert als erstes Rückgabe-Argument den minimalen Wert von `err` (interessiert hier aber nicht, deshalb wird dieser mit`~`unterdrückt), Der zweite Rückgabewert ist der Index `idx`, an dem das Minimum angenommen wird. Das optimale `omega` erhalten wir also durch `omega(idx)`.
- Wir erhalten das folgende Schaubild:



Wir sehen, dass der minimale Fehler für $\omega \approx 0.4$ angenommen wird. Man rechnet leicht nach, dass dieses numerische Resultat mit dem theoretischen Resultat aus der Vorlesung überein stimmt.