
Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Umgang mit der Matlab-Umgebung
 - Schreiben einfacher Skripte und Funktionen in Matlab
 - Bestimmung der Konvergenzrate verschiedener Iterationsverfahren
 - Visualisierung der Matrix-p-Normen
-

Praktikumsaufgabe 11 – Kontraktionsraten von Iterationsverf.

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

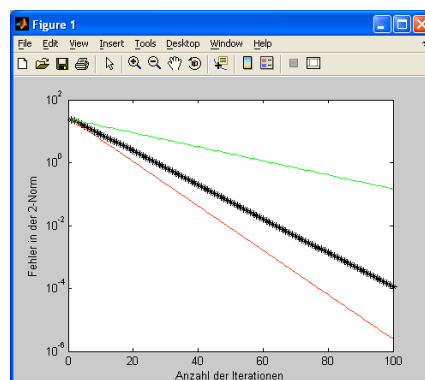
• Problembeschreibung

Rufen Sie das folgende Skript im Kommando-Fenster von Matlab auf.

```
% ex11x01.m
clear all
n = 8;
A = gallery('poisson',n);
%A = gallery('wathen',n,n);
dim = size(A,1);
fprintf('Dimension der Matrix A ist %i x %i.\n',dim,dim);

reply = input('Möchten Sie sich die Besetzungsstruktur der Matrix ansehen? Y/N [N]: ','s');
if reply=='Y' | reply=='y'
    figure(1);
    spy(A);
end
OK = 0;
while ~OK
    reply = input('Wieviele Iterationen möchten Sie ausführen? [10-100]','s');
    [max_it,OK]=str2num(reply);
end
b = ones(dim,1);
x = rand(dim,1);
x_exakt = A\b;
error = zeros(max_it,1);
ew = eig(full(A));
gRichardson = 2/(min(ew)+max(ew));
for s = 1:max_it
    %x = x + gRichardson * (b-A*x);    % Richardson-Iteration
    % rho = (ew_max - ew_min) / (ew_max + ew_min)
    %x = x + diag(diag(A)) \ (b-A*x); % Jacobi-Iteration
    x = x + triu(A) \ (b-A*x);        % Gauss-Seidel-Iteration
    error(s) = norm(x-x_exakt);
end
c = error(1);
rho_u = 0.95;
rho_o = 0.85;
semilogy(1:s,error,'k*',1:s,c*rho_o.^(0:s-1),'r',1:s,c*rho_u.^(0:s-1),'g')
xlabel('Anzahl der Iterationen')
ylabel('Fehler in der 2-Norm')
```

Sie sollten auf dem Bildschirm die folgende Ausgabe erhalten.



Das Programm berechnet zur rechten Seite $b=1$ und einer gegebenen Matrix A die Lösung x zu $Ax=b$ mittels Gauß-Seidel-Verfahren. (Richardson- und Jacobi-Verfahren sind ausdokumentiert.) Des weiteren wird der Fehler zur exakten Lösung bestimmt und in Abhängigkeit von der Iterationszahl graphisch dargestellt.

Der mittlere Graph ist der gemessene Fehler $x-x_s$ in der euklidischen Norm, der obere und untere Graph entsprechen der Funktion $c \rho^s$ mit $\rho=0.85$ bzw 0.95 .

- Modifizieren Sie die Werte von `rho_u` und `rho_o` im Skript durch „try and error“ so, dass Sie eine auf 2 Stellen genaue obere und untere Schranke zur asymptotischen Konvergenzrate für das Gauß-Seidel-Verfahren erhalten.
- Wiederholen Sie dies für das Jacobi-Verfahren und Richardson-Verfahren.
- Testen Sie, ob die drei Verfahren auch für die Matrix funktioniert, die in der Matlab-Galerie unter dem Kennwort WATHEN abgelegt ist.
- Testen Sie, ob die Matrizen (schwach) diagonaldominant sind. Arbeiten Sie vektorwertig mit den Befehlen `sum` und `diag`. Formulieren Sie die Abfrage als eine Anweisung.

• Vorlage

Die Datei `ex11x01.m` ist eine lauffähige Matlab-Funktion. Die Datei können Sie sich unter www.mathematik.uni-ulm.de/numerik/ downloaden. Folgen Sie dabei dem Link **Lehre->Numerik1...**

Praktikumsaufgabe 12 – Visualisierung der Matrix-p-Norm

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

• Problembeschreibung

Der folgende Satz soll in dieser Praktikumsaufgabe für $p=1,2,3,4,\infty$ und den Fall \mathbb{R}^2 visualisiert werden.

Satz A.2.9.

Die Matrix- p -Normen sind unter allen mit der Vektornorm $\|\cdot\|_p$ verträglichen Matrixnormen die kleinsten.

Gesucht ist also nach dem kleinsten Wert α_p , so dass

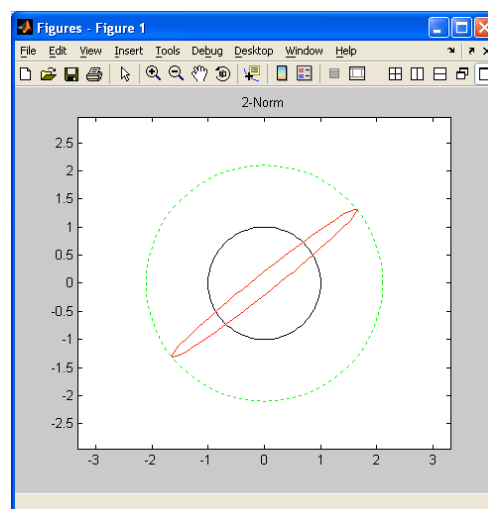
$$\|x\|_p \leq \alpha_p \|Ax\|_p \quad (x \in \mathbb{R}^2) \text{ gilt.}$$

Gelte $\|x\|_p = 1$, dann ist α_p der Wert, um den man den Einheitskreis in der p -Norm aufblasen muss, damit Ax darin enthalten ist.

Es soll also geplottet werden,

- der Einheitskreis in der p -Norm,
- das Bild des Einheitskreises und
- das $\|A\|_p$ -fache des Einheitskreises in der p -Norm.

Für $A = [63, 53; 43, 50]/50$; und $p=2$ sollten Sie folgende Grafik erhalten



%% ex12x01.m

```

clear all
A = [63,53; 43,50]/50;

N = 1+4*20;
t = linspace(0,2*pi,N);
vec = [cos(t);sin(t)];
ind = [1:N;2:N,1];

for p = 2
    x = [];
    for j = 1:size(vec,2)
        %%
        %% Vektoren in vec(:,j) sollen nach Normierung in x(:,j)
        %% gespeichert werden
        %%
    end
    Ax = A * x;
    lambda = norm(A,p);
    x = x'; Ax = Ax';
    plot(x(ind,1),x(ind,2),'k-', ...
         lambda*x(ind,1),lambda*x(ind,2),'g:', ...
         Ax(ind,1),Ax(ind,2),'r-')
    mx = max(max([Ax,x,lambda*x]));
    axis(1.4*mx*[-1,1,-1,1]); axis equal
    title([num2str(p),'-Norm'])
    pause
end

```

- Modifizieren Sie `ex12x01` so, dass Sie damit den Fall $p=2$ visualisieren können.
- Modifizieren Sie das erstellte Skript so, dass Sie damit den Fall $p=1,2,\infty$ darstellen können.
- Funktioniert der Befehl `norm` in Matlab auch für die 3- und 4-Norm? Ersetzen Sie diesen ggf. durch eine eigene von Ihnen zu erstellende Funktion `my_norm`.
(Für $p=1,2,\infty$ kann diese, den Befehl `norm` verwenden.)

• Vorlage

Die Datei `ex12x01.m` ist eine lauffähige Matlab-Funktion. Die Datei können Sie sich unter www.mathematik.uni-ulm.de/numerik/ downloaden. Folgen Sie dabei dem Link **Lehre->Numerik1...**