Joseph Dorfer 11911725

Übungsblatt 5

Aufgabe 1

Für $n \epsilon \mathbb{N}$, $n > 1$, sei die invertierbare Matrix A durch $A = D + L + U$ mit einer invertierbaren diagonalen Matrix D, einer streng unterdreieckigen Matrix L und einer streng oberdreieckigen Matrix U zerlegt. Für $\boldsymbol{x}, \boldsymbol{b} \,\boldsymbol{\epsilon}\, \mathbb{R}^{\boldsymbol{n}}$ wird die Jacobi-Methode $x^{(k+1)} = T_J x^{(k)} + M_J^{-1} b$, $k \epsilon \mathbb{N}$, auf Seite 79 im Skriptum verwendet, um das lineare Gleichungssystem $Ax = b$ iterativ zu lösen.

Teil (a)

Wenn A streng diagonal dominant ist, folgt $\|T_J\|_\infty < 1$.

Wissen: $\|T_J\|_\infty = \max\limits_{1<i<n} \sum_{j=1}^{n} |t_{ij}|$ Zeilensumme

$$T_J = -D^{-1}(L+U) = \begin{pmatrix} 0 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & & \ddots & \vdots \\ \vdots & & & \frac{a_{n-1\,n}}{a_{n-1\,n-1}} \\ \frac{a_{n1}}{a_{n\,n}} & \cdots & \frac{a_{n\,n-1}}{a_{n\,n}} & 0 \end{pmatrix}$$

Daraus folgt $\|T_J\|_\infty = \max\limits_{1<i<n} \sum_{j=1}^{n} |t_{ij}| = \max\limits_{1<i<n} \sum_{j=1, j\neq i}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| = \max\limits_{1<i<n} \frac{1}{|a_{ii}|} \sum_{j=1, j\neq i}^{n} |a_{ij}|$

Nun verwenden wir, dass A streng diagonal dominant ist, also dass gilt

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

Woraus folgt: $$\|T_J\|_\infty < \frac{1}{|a_{ii}|} * |a_{ii}| = 1$$

Damit ist Aussage (a) wahr.

Teil (b)

Wenn A streng diagonal dominant ist, folgt $\|T_J\|_1 < 1$.

Wissen: $\|T_J\|_1 = \max\limits_{1<j<n} \sum_{i=1}^{n} |t_{ij}|$ Spaltensumme

Gegenbeispiel: $A = \begin{pmatrix} 100 & 0 & 0 \\ 5 & 7 & 0 \\ 5 & 0 & 7 \end{pmatrix}$ ist streng diagonal dominant, aber

$$\|T_J\|_1 = \left\| \begin{matrix} 0 & 0 & 0 \\ 5/7 & 0 & 0 \\ 5/7 & 0 & 0 \end{matrix} \right\| = \frac{10}{7} > 1$$

Damit ist Aussage (b) falsch.

Teil (c)

Die Jacobi Iterierten $\{x^{(k)}\}_{k=0}^{\infty}$ können divergieren, wenn die folgende Iteration konvergiert

$$y^{(k)} = -(L+U)D^{-1}y^{(k-1)} + b, k \in \mathbb{N}$$

Wir betrachten die Jacobi Iteration:

$$x^{(k)} = -D^{-1}(L+U)x^{(k-1)} + D^{-1}b$$

Wir multiplizieren von links mit D und erhalten

$$Dx^{(k)} = -DD^{-1}(L+U)x^{(k-1)} + DD^{-1}b = -(L+U)x^{(k-1)} + b$$

Anschließend setzen wir $y^{(k)} = Dx^{(k)} \Leftrightarrow x^{(k-1)} = D^{-1}y^{(k-1)}$ und erhalten

$$y^{(k)} = -(L+U)D^{-1}y^{(k-1)} + b$$

Weil D invertierbar und die Multiplikation mit D somit bijektiv ist, handelt es sich bei den oben genannten Umformungen um Äquivalenzumformungen. Damit sind die beiden Gleichungen für $y^{(k)} = Dx^{(k)}$ äquivalent. Somit ist Aussage (c) falsch.

Teil (d)

Wenn A^T streng diagonal dominant ist müssen die Jacobi Iterierten $\{x^{(k)}\}_{k=0}^{\infty}$ zu $x = A^{-1}b$ konvergieren.

Wenn A^T streng diagonal dominant ist gilt $|a_{jj}| > \sum_{i \neq j} |a_{ij}|$

Wenn man T nun wählt wie in $y^{(k)} = -(L+U)D^{-1}y^{(k-1)} + b$

Woraus folgt $\|T\|_1 = \max_{1<j<n} \sum_{i=1}^{n} |t_{ij}| = \max_{1<j<n} \sum_{i=1, i \neq j}^{n} \left|\frac{a_{ij}}{a_{jj}}\right| = \max_{1<j<n} \frac{1}{|a_{jj}|} \sum_{i=1, i \neq j}^{n} |a_{ij}|$

$< \frac{1}{|a_{jj}|} * |a_{jj}| = 1$

Weil $\|.\|_1$ eine induzierte Norm ist, muss $\rho(T) < 1$ gelten.

Satz (S. 86 im Skriptum Stand 4.11.2020)

Wenn für das System $Ax^* = b$ gilt $x^* = Tx^* + c$, dann erfüllt das Verfahren $x^{(k+1)} = Tx^{(k)} + c$ die Konvergenz $\lim_{k \to \infty} x^{(k)} = x^*$ genau dann, wenn $\rho(T) < 1$.

Die Bedingung $\rho(T) < 1$ ist wie oben erörtert erfüllt, unter verwendung von $x^{(k)} = y^{(k)}$.

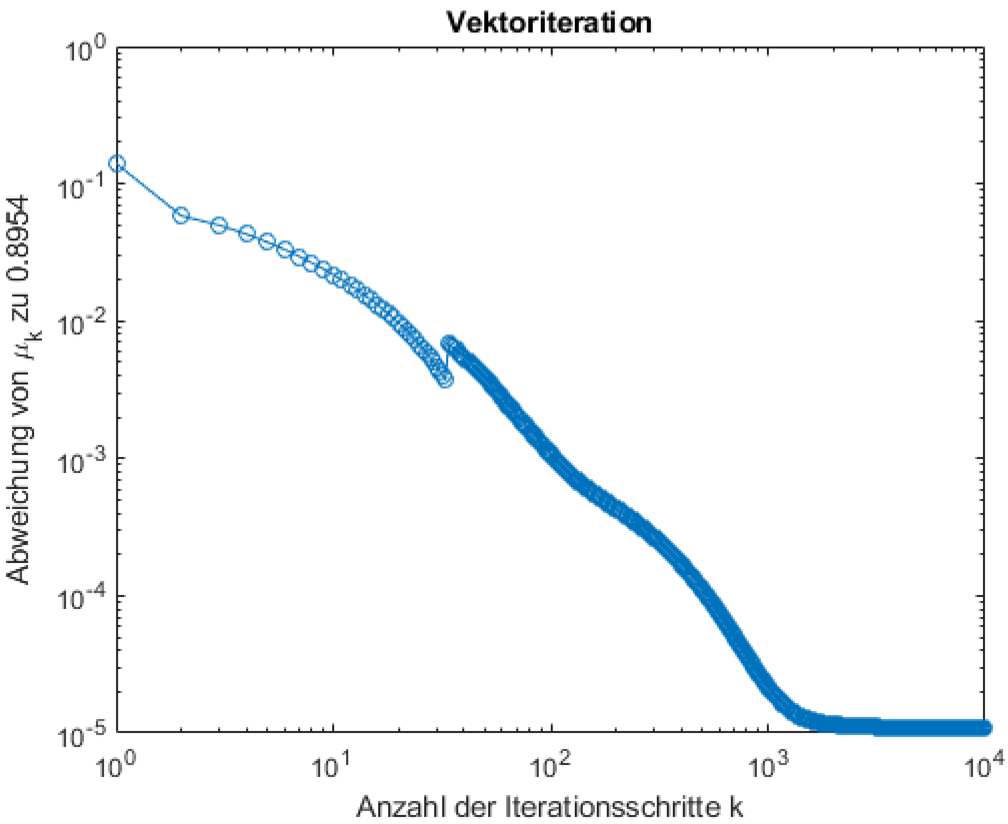
Damit konvergiert das Verfahren nach dem obigen Satz.

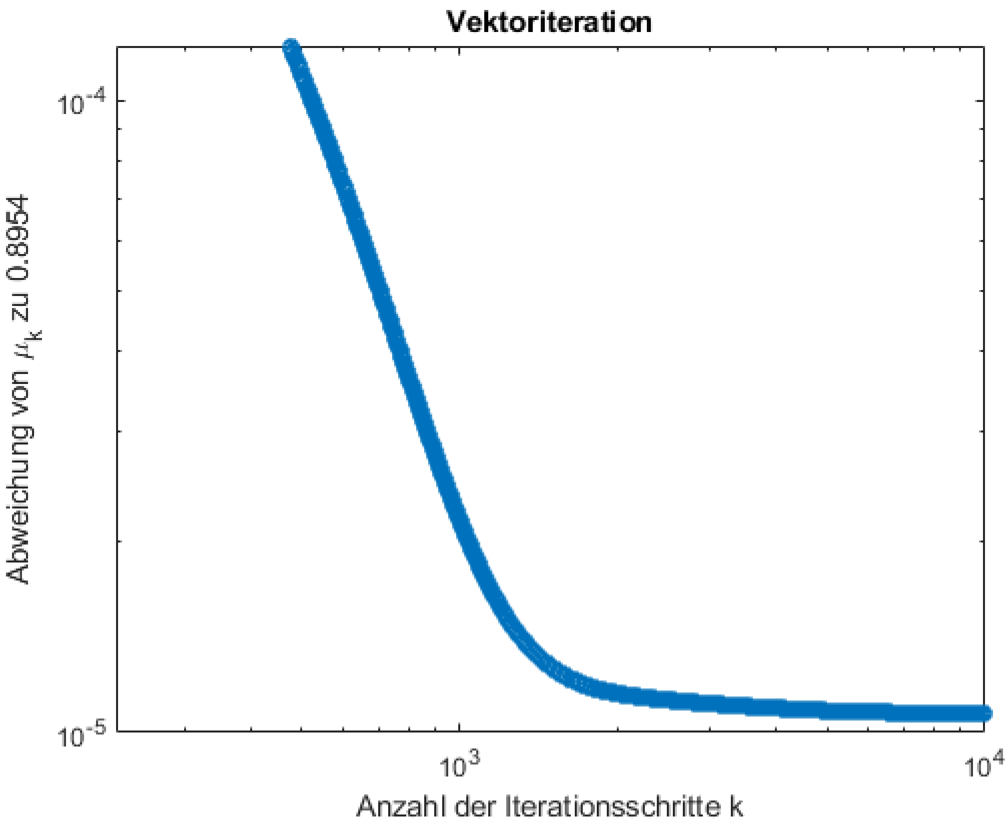
Damit ist Aussage (d) wahr.

```
clear all
close all
clc

N = 50;
N1 = N-1;
h = 1/N;
mu = 0.01;
Q = spdiags([-ones(N1,1),ones(N1,1)],[0,1],N1,N);
i = speye(N);
Dx = kron(i,Q);
Dy = kron(Q,i);
D = Dx'*Dx + Dy'*Dy;
I = speye(N*N);
v = sin(1:N*N)';
A = I + mu*D/h^2;
ustar = A \ v;

%Wir betrachten: Au=v
%***************************************************************************

L=tril(A,-1);
U=triu(A,1);
d=diag(A);
D=diag(d);

T_J=I-inv(D)*A;                         %Iterationsmatrix Jacobi
r=max(abs(eigs(T_J)));                  %Spektralradius
formel_rho=2/(1+sqrt(1-r^2))            %Übungsblstt

T_SSOR= @(omega) ((D+omega*U)\I)*(omega*L-(1-omega)*D)*((D+omega*L)\I)*(omega*U-(1-
omega)*D);        %Blatt 4
M_SSOR= @(omega) omega*(2-omega)*((D+omega*U)\I)*D*((D+omega*L)\I);

rho= @(omega) norm(eig(full(T_SSOR(omega))), inf);
[omega,fval] = fminbnd(rho,0,2);                    %untersucht das offene
Intervall auf ein Minimum/Infimum

[omega,fval]

%*
%formel_rho =

%    1.7518


%ans =

%    1.6788    0.8594

%Somit ist a) falsch.
```

```
%***************************************************************************
%dominanter Eigenwert Abschätzung

%omegastern=omega;
om=1.6788;

Tssor=om*U-(1-om)*D;                        %entspricht dem Ausdruck von Blatt 4
Tssor=(D+om*L)\Tssor;
Tssor=(om*L-(1-om)*D)*Tssor;
Tssor=(D+om*U)\Tssor;

x=ones(N*N,1);
x=x/norm(x,inf);
mu=1;
ma=0;

for k=1:10000
    y=Tssor*x;
    l1=find(abs(y) == norm(y,inf),1);
    xa=x;
    x=y/y(l1);
    if(k>1)
        ma=mu;
        mu=y(l2);
    end
    z(k)=mu-0.8594;
    l2=l1;
end

mu

loglog(1:10000,z,'-o')
xlabel('Anzahl der Iterationsschritte k')
ylabel('\mu_k'),
title ('Vektoriteration')

%mu =

%    0.8594                      %für ein genaueres Ergebnis wähle Format 'long'

%Somit ist b) richtig.

%***************************************************************************

 L=tril(A,-1);
 U=triu(A,1);
 d=diag(A);
 D=diag(d);

 om=0.5;

 DwL=D+om*L;
 DwU=D+om*U;
 DU=(1-om)*D-om*U;
```

```
 DL=(1-om)*D-om*L;

 u=ones(N*N,1);
 c=0;

 while((norm(u-ustar)/norm(ustar)) > (1/2^23))        %        %Folie 97, 05.11.2020

     y=DU*u;
     u=DwL\y + om*(DwL\v);
     z=DL*u;
     u=DwU\z + om*(DwU\v);

     c=c+1;
 end
 c

% c =
%
%    1431

%***************************************************************************

om=1.5;

DwL=D+om*L;
DwU=D+om*U;
DU=(1-om)*D-om*U;
DL=(1-om)*D-om*L;

u=ones(N*N,1);
d=0;

while((norm(u-ustar)/norm(ustar)) > (1/2^23))        %        %Folie 97, 05.11.2020

   y=DU*u;
   u=DwL\y + om*(DwL\v);
   z=DL*u;
   u=DwU\z + om*(DwU\v);

   d=d+1;
end
d

% d =
%
%    170
```


Vektoriteration
Abweichung von μ_k zu 0.8954
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^0
10^1
10^2
10^3
10^4
Anzahl der Iterationsschritte k

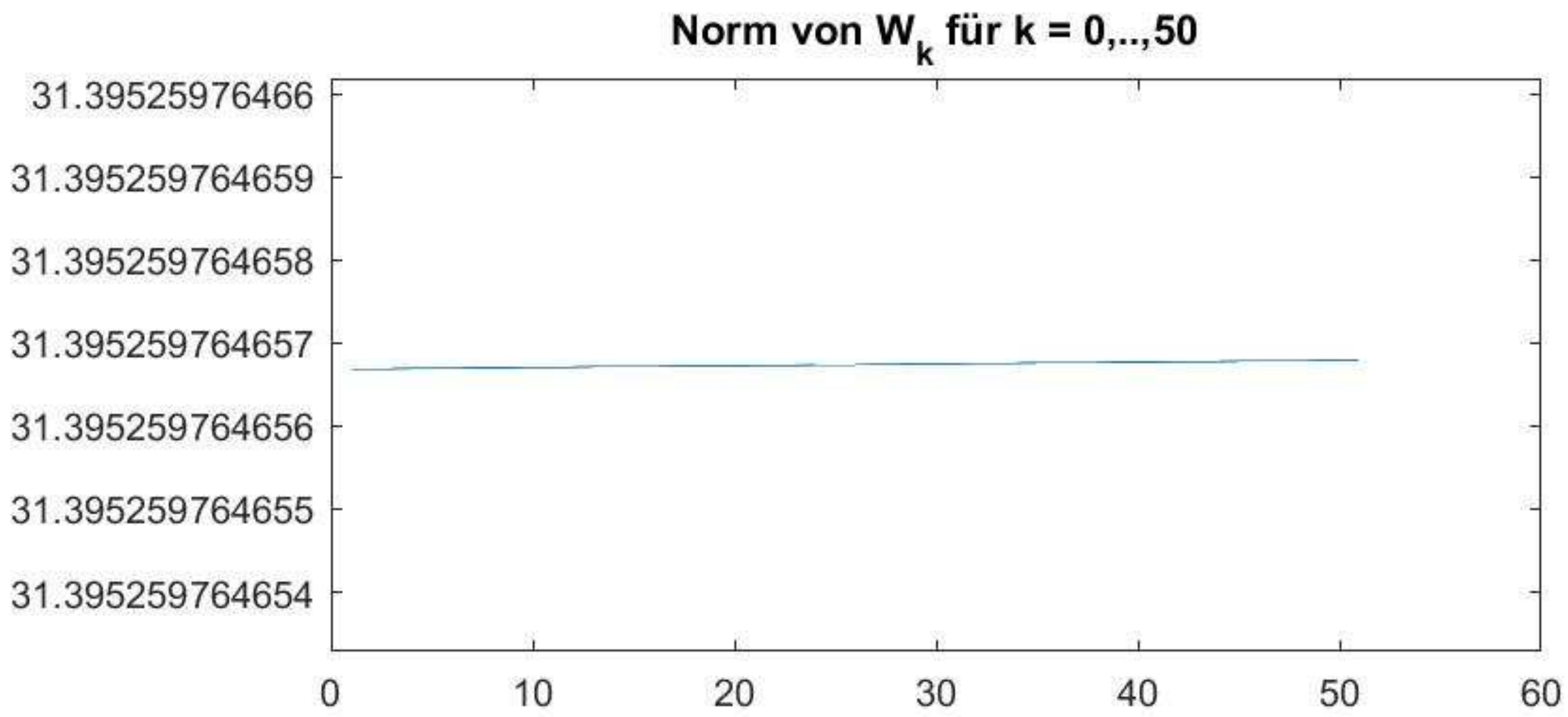

Vektoriteration
Abweichung von μ_k zu 0.8954
10^{-4}
10^{-5}
10^3
10^4
Anzahl der Iterationsschritte k

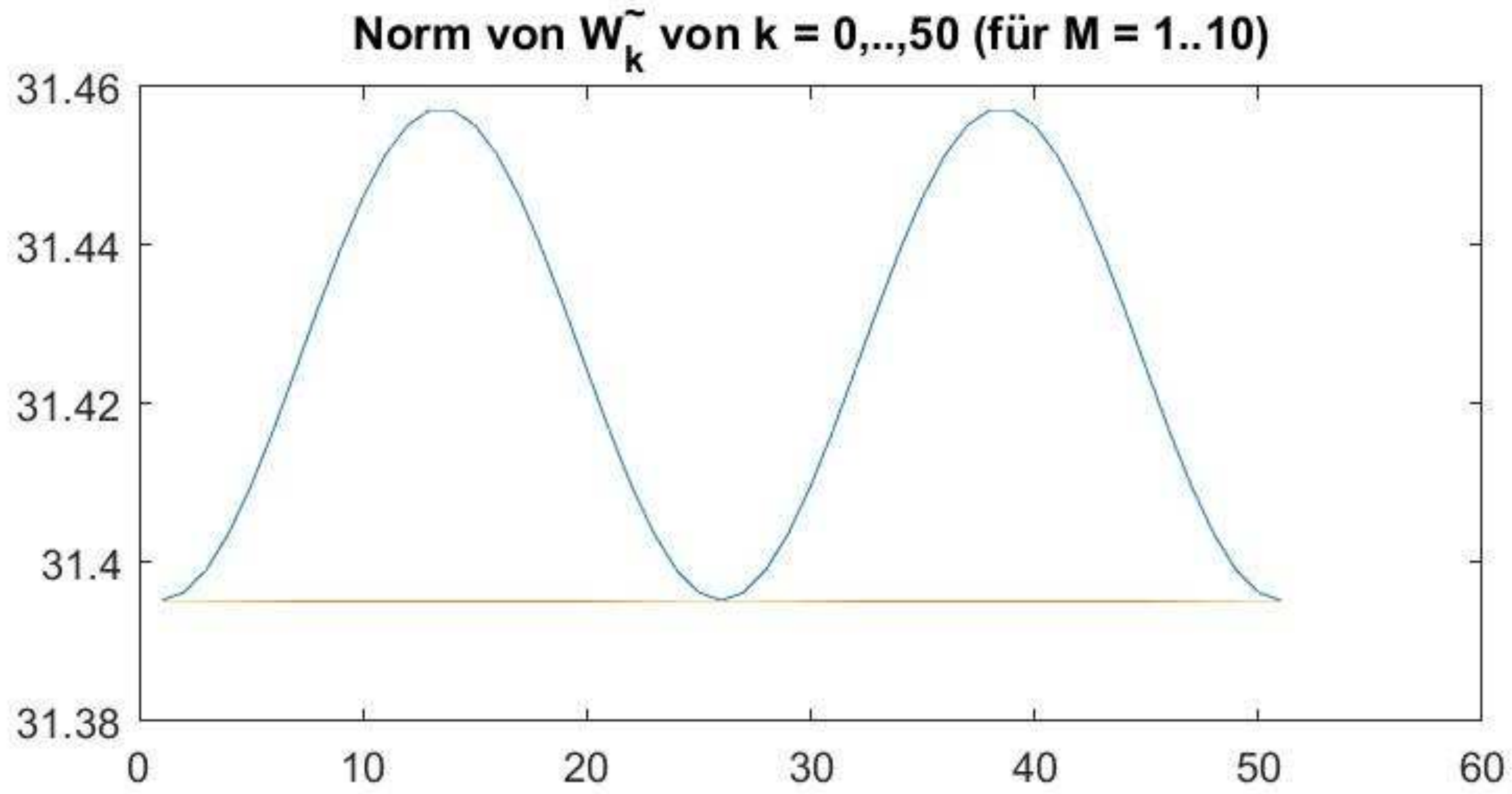
```
N = 50; N1 = N-1; h = 1/N;
mu = 0.01;
Q = spdiags([-ones(N1,1),ones(N1,1)],[0,1],N1,N);
i = speye(N);
Dx = kron(i,Q);
Dy = kron(Q,i);
D = Dx'*Dx + Dy'*Dy;
I = speye(N*N);
v = sin(1:N*N)';
A = I + mu*D/h^2;
ustar = A \ v;

L = tril(A,-1);
U = triu(A,1);
DD = A-L-U;

kmax = N;

%%%%%%%%%%%%% (a) %%%%%%%%%%%%%%

Ba = speye(length(A));

u = v;                              %x(0)
r = v - A*u;                        %v(1),r(0)
z = Ba \ r;                          %
r1 = z'*r; r2 = r1; a = 0*r;

Ua = zeros(N,1);

for k=1:kmax
    s = r1/r2;
    a = z + s*a;                    %a = r , B = I
    w = A*a;                        %A*r
    t = r1/(a'*w);                  %<r,r>/<r,Ar>
    u = u + t*a;
    r = r - t*w;
    z = Ba \ r;
    r2 = r1;
    r1 = z'*r;
    Ua(k) = sqrt(sum((ustar - u).^2));
end

max(abs(u-ustar))

%%%%%%%%%%%%% (b) %%%%%%%%%%%%%%

Mj  = DD;
Mjh = sqrtm(full(Mj));
AJ  = Mjh\speye(N*N);
AJ  = A*AJ;
AJ  = Mjh\AJ;           % Mj^(-1/2)*A*Mj^(-1/2)
kappa_J = cond(full(AJ));

Msgs = DD\(DD+U);
Msgs = (DD+L)*Msgs;
Msgsh = sqrtm(full(Msgs));
Asgs = speye(N*N);
Asgs = Msgsh\Asgs;
Asgs = A*Asgs;
Asgs = Msgsh\Asgs;          %Msgs^(-1/2)*A*Msgs^(-1/2)
kappa_SGS = cond(full(Asgs));

kappa = cond(full(A));




%%%%%%%%%%%%%%%% (c) %%%%%%%%%%%%%%%%%%

Bc = DD;
```

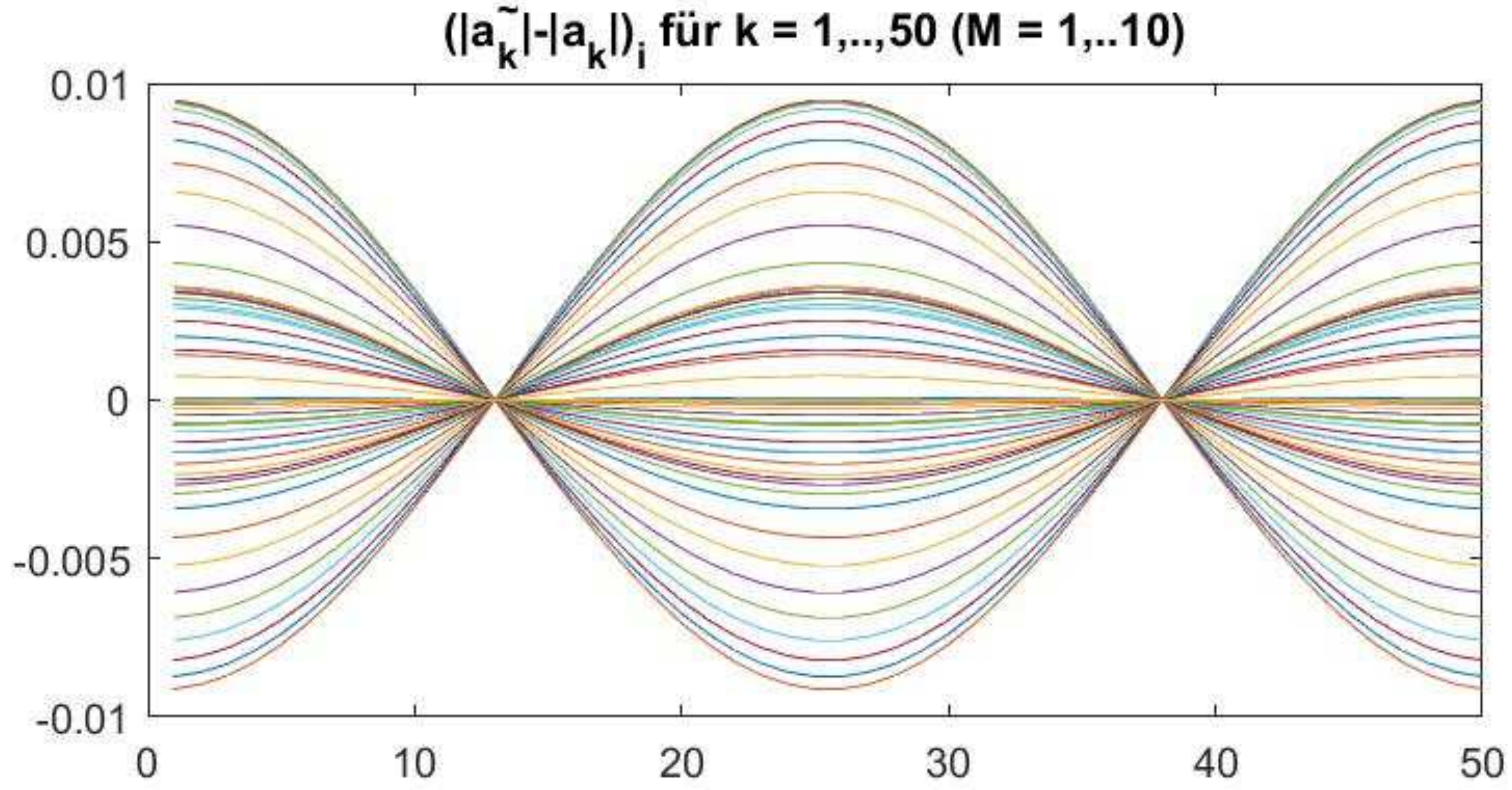
```
u = v;                          %x(0)
r = v - A*u;                    %v(1),r(0)
z = Bc \ r;                      %
r1 = z'*r; r2 = r1; a = 0*r;

Uc = zeros(N,1);

for k=1:kmax
    s = r1/r2;
    a = z + s*a;                %a = r , B = I
    w = A*a;                    %A*r
    t = r1/(a'*w);              %<r,r>/<r,Ar>
    u = u + t*a;
    r = r - t*w;
    z = Bc \ r;
    r2 = r1;
    r1 = z'*r;
    Uc(k) = sqrt(sum((ustar - u).^2));
end

%%%%%%%%%%%%%% (d) %%%%%%%%%%%%%%%


Bd = Msgs;

u = v;                          %x(0)
r = v - A*u;                    %v(1),r(0)
z = Bd \ r;                      %
r1 = z'*r; r2 = r1; a = 0*r;

Ud = zeros(N,1);

for k=1:kmax
    s = r1/r2;
    a = z + s*a;                %a = r , B = I
    w = A*a;                    %A*r
    t = r1/(a'*w);              %<r,r>/<r,Ar>
    u = u + t*a;
    r = r - t*w;
    z = Bd \ r;
    r2 = r1;
    r1 = z'*r;
    Ud(k) = sqrt(sum((ustar - u).^2));
end

figure
plot(1:N,Ua,'r')
hold on
plot(1:N,Uc,'g')
hold on
plot(1:N,Ud,'b')
```

```matlab
clear;clc;

N = 50;
N1 = N-1;
h = 1/N;
tau = h;
nu = 1;

x = linspace(0,1,N+1);
x = x(1:N)' + h/2;

Q = spdiags([-ones(N1,1),ones(N1,1)],[0,1],N1,N);
B = (nu/h)*[sparse(N1,N1),Q;-Q',sparse(N,N)];
I = speye(2*N-1);
A = I - (tau/2)*B;
C = I + (tau/2)*B;

D = diag(diag(A));
L = tril(A,-1);
U = triu(A,1);

a = cos(pi*(2*x-1)); % a_0
at = a;

W = [(nu/h)*Q*a;0*a]; %entspricht W_0

WW = zeros(2*N-1,N+1); %hier werden alle W_k hineingespeichert
WW(:,1) = W;
for k = 1:N %Berechnung der W_k wie in 8c), ÜB3 mittels \
    Ws = W;
    W = A\(C*Ws);
    WW(:,k+1) = W;

    a = a + (tau/2)*(WW(N:(2*N-1),k+1) + WW(N:(2*N-1),k));
    a_k(:,k) = a; %speichert die a_k für k=1..N
end

for i = 1:N+1
    norm_Ww(i) = norm(WW(:,i),2);
end
subplot(2,2,1)
plot(norm_Ww)
title('Norm von W_k für k = 0,..,50')

for M = 1:10
    WT = zeros(2*N-1,N+1); %hier werden alle Wt_k abhängig von M hineingespeichert
    WT(:,1) = WW(:,1);
    at = cos(pi*(2*x-1));

    for k = 1:N
        Wt = WT(:,k);
        %Wt = (1.0e-5)*sign(WT(:,k));
        for l = 1:M
            Wt = (D+L)\(C*WT(:,k) - U*Wt);
            Wt = (D+U)\(C*WT(:,k) - L*Wt);
        end
        WT(:,k+1) = Wt;
%         a = a + (tau/2)*(WW(N:(2*N-1),k+1) + WW(N:(2*N-1),k));
        at = at + (tau/2)*(WT(N:(2*N-1),k+1) + WT(N:(2*N-1),k));
        at_k(:,k) = at; %speichert die at_k für k=1..N

%         subplot(2,2,4)
%         plot (x,a,'b',x,at,'g');
%         axis([0 1 -1 +1]);
%         legend('EX','SGS');
%         drawnow;
%         pause(0.1)
```

```
    end

    for i = 1:(N+1)
        norm_Wt(i) = norm(WT(:,i),2);
    end
    subplot(2,2,2)
    plot(norm_Wt)
    title('Norm von W^~_k von k = 0,..,50 (für M = 1..10)')
    hold on

    diff_a = abs(at_k)-abs(a_k);
    subplot(2,2,3)
    plot(diff_a)
    title('(|a^~_k|-|a_k|)_i für k = 1,..,50 (M = 1,..10)')
    hold on

end
infnorm_diff = norm(WT-WW,inf);
infnorm_WW = norm(WW, inf);

rel_err = infnorm_diff/infnorm_WW;
disp('Relativer Fehler:')
disp(rel_err)
disp('Schwelle zur einfachen Genauigkeit')
disp(2^(-23))

% a) ist falsch, weil für M = 1 gibt es k, sodass die 2-Norm von W~_k nicht
% gleich 31.4 ist (siehe Grafik oben rechts, die blaue Linie entspricht M=1)

% b) ist falsch (siehe Grafik unten links) -> die Ungleichung ist nicht für
% alle k,M erfüllt

% c) und d) sind richtig
% für c: setze obere Grenze für M auf 3
% für d: setze obere Grenze für M auf 780 und ändere Startwert (Zeil 54 statt 53)

%Um die Auslenkungen a_k und a~_k zu plotten (Zeile 61,65 - 70), muss zuerst
die äußere
%M-Schleife deaktiviert werden und M auf einen fixen Wert gesetzt werden
```


Norm von W_k für k = 0,..,50
31.39525976466
31.395259764659
31.395259764658
31.395259764657
31.395259764656
31.395259764655
31.395259764654
0
10
20
30
40
50
60


Norm von $W_k^{\sim}$ von k = 0,..,50 (für M = 1..10)
31.46
31.44
31.42
31.4
31.38
0
10
20
30
40
50
60


$(|a_k^{\sim}|-|a_k|)_i$ für k = 1,..,50 (M = 1,..10)
0.01
0.005
0
-0.005
-0.01
0
10
20
30
40
50