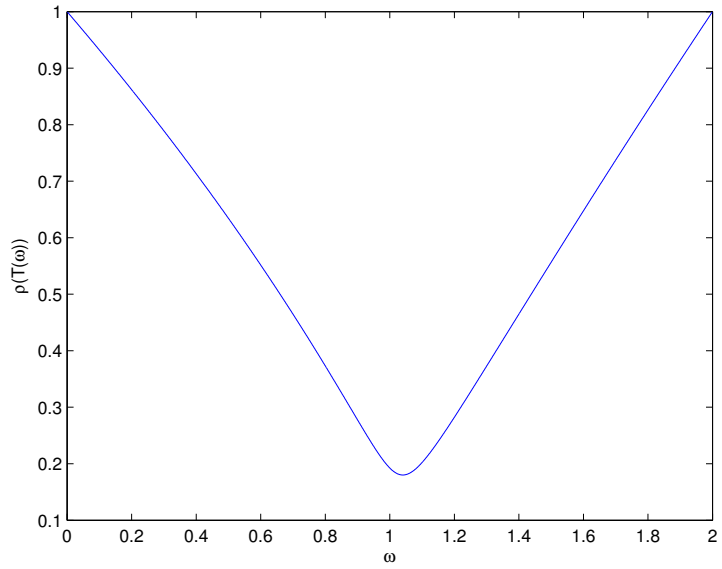
Musterlösung Serie 3

1. **a)** $T = (D + \omega L)^{-1} (-\omega R + (1 - \omega)D)$

```
A=[3 1 1;1 3 1;1 1 3];
D=diag(diag(A));
L=tril(A)-D;
R=triu(A)-D;
omega_vec=0.001:0.001:1.999;
rho=zeros(size(omega_vec));
for i=1:length(omega_vec)
 omega=omega_vec(i);
 T=-inv(D+omega*L)*(-omega*R+(1-omega)*D);
 rho(i)=max(abs(eig(T)));
end
plot(omega_vec,rho)
xlabel('\omega')
ylabel('\rho(T(\omega))')
```

Der Spektralradius ist immer kleiner als 1, das Verfahren konvergiert also für $0 < \omega < 2$. Der Spektralradius ist am kleinsten für $\omega \approx 1.04$.



Bitte wenden!

b) Für $\omega = 0.8$ ist $||T(\omega)||_2 = 0.49597$, für $\omega = 1.04$ ist $||T_\omega||_2 = 0.53653$. Es gilt

$$||e^k||_2 \leq ||T(\omega)||_2^k ||e^0||_2.$$

Um den Fehler um einen Faktor 10^{-10} zu reduzieren, muss also gelten:

$$\underline{\text{für } \omega = 0.8}\text{: } ||T(\omega)||_2^k \leq 10^{-10} \Rightarrow k \geq \frac{\log 10^{-10}}{\log ||T(\omega)||_2} \approx 32.836,$$

also sollte man im schlechtesten Fall 33 Iterationen benötigen.

$$\underline{\text{für } \omega = 1.04}\text{: } ||T(\omega)||_2^k \leq 10^{-10} \Rightarrow k \geq \frac{\log 10^{-10}}{\log ||T(\omega)||_2} \approx 36.981,$$

also sollte man im schlechtesten Fall 37 Iterationen benötigen. Numerische Berechnung (wir benützen zur Fehlerberechnung die exakte Lösung $x = (-\frac{1}{2}, 2, -\frac{3}{2})^\top$):

```
A=[3 1 1;1 3 1;1 1 3];
b=[-1;4;-3];
D=diag(diag(A));
L=tril(A)-D;
R=triu(A)-D;
omega_vec=[0.8 1.04];
x_exact=[-0.5;2;-1.5];
xstart=[-0.49; 1.9; -1.4];
nb_it=zeros(1,length(omega_vec));
tol_it=1e-10;
for i=1:length(omega_vec)
    omega=omega_vec(i);
    T=(D+omega*L)\(-omega*R+(1-omega)*D);
    xnew=xstart;
    k=0;
    while norm(xnew-x_exact,2) > ...
            tol_it*norm(xstart-x_exact, 2) && (k<5000)
        xnew=T*xnew+(D+omega*L)\(omega*b);
        k=k+1;
    end
    x_SOR=xnew;
    nb_it(i)=k;
end
nb_it
```

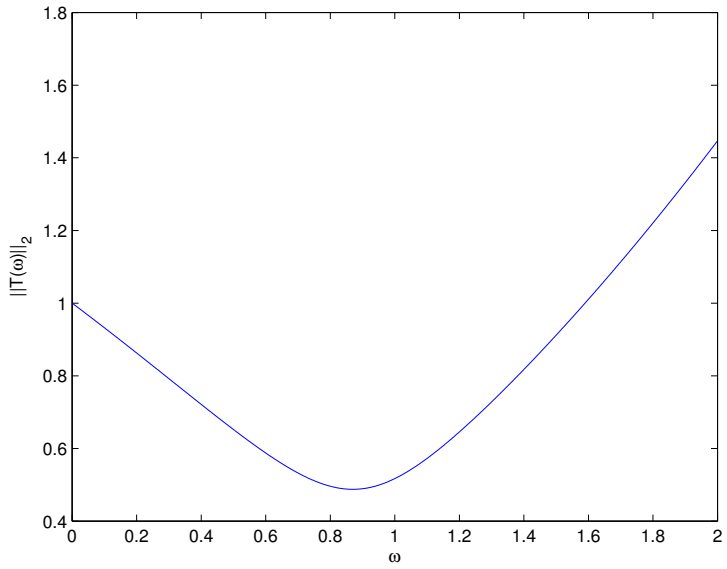
Für $\omega = 0.8$ braucht das Verfahren 24 Iterationen, für $\omega = 1.04$ nur 14. Der Grund dafür ist, dass $\varrho(T(0.8)) > \varrho(T(1.04))$ gilt.

c)

```
A=[3 1 1;1 3 1;1 1 3];
D=diag(diag(A));
L=tril(A)-D;
R=triu(A)-D;
omega_vec=0.001:0.001:1.999;
twonorm=zeros(1,length(omega_vec));
for i=1:length(omega_vec)
omega=omega_vec(i);
```

Siehe nächstes Blatt!

```
 T=inv(D+omega*L)*(-omega*R+(1-omega)*D);
 twonorm(i)=norm(T);
end
plot(omega_vec,twonorm)
xlabel('\omega')
ylabel('||T(\omega)||_2')
```

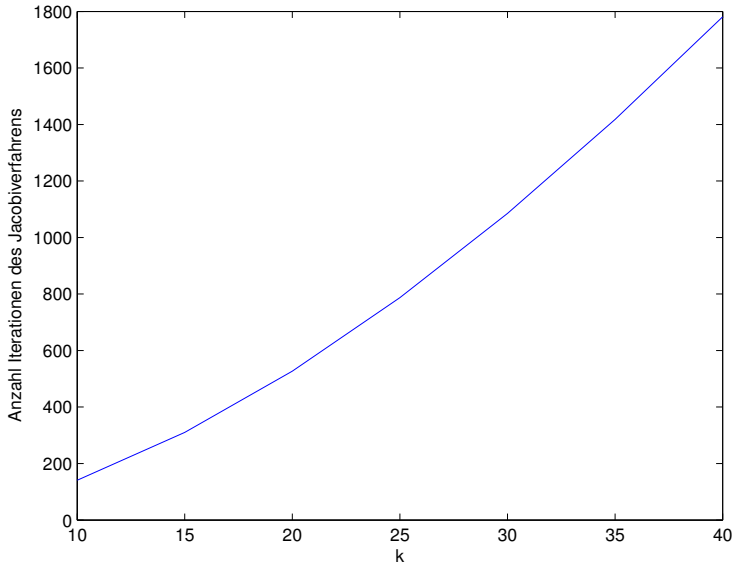


Obwohl $||T(\omega)||_2 < 1$ nur für $0 < \omega < 1.59$ gilt, konvergiert das SOR-Verfahren trotzdem für $0 < \omega < 2$, denn der Spektralradius und nicht die 2-Norm von $T(\omega)$ entscheidend ist. Für $\omega = 1.8$ konvergiert das SOR-Verfahren mit dem Code aus **b)** nach 118 Schritten, auch wenn $||T(1.8)||_2 = 1.2206 > 1$, denn $\varrho(T(1.8)) = 0.8261 < 1$.

2. a)
```
function [x, iter, time] = my_jac(A,b, x0, tol)
 tic
 D=diag(diag(A)); L=tril(A)-D; R=triu(A)-D;
 x=x0; % start with initial guess
 maxIter=2e3; iter=0;
 while (iter<maxIter)
  iter=iter+1;
  xold=x;
  x=D\((-L-R)*x+b); % Jacobi iteration without inverse
  if (norm(x-xold)/norm(x)<tol) break; end % converged!
 end
 time=toc;
 if (iter==maxIter)
  disp(['no convergence after ',int2str(iter),' iterations']);
 end;
```

b)
```
tol=1e-5;
```

```
k_vec=10:5:40;
iter_vec=zeros(1,length(k_vec));
 for i=1:length(k_vec)
  k=k_vec(i);
  A=model(k);
  b=rhs(k);
  x0=zeros(size(b));
  [x,iter]=my_jac(A,b,x0,tol);
  iter_vec(i)=iter;
 end
 plot(k_vec,iter_vec)
 xlabel('k')
 ylabel('Anzahl Iterationen des Jacobiverfahrens')
```



c)
```
A=[2 1 0;1 2 1;0 1 2];
B=[1 -1 0;5 2 0;0 0 1];
C=[3 2 1;2 3 2;1 2 3];
b=[1 1 1]';
x0=[0 0 0]';
[x,iter, time]=my_jac(A,b,x0,tol)
fehler1=norm(x-A\b)
[x,iter, time]=my_jac(B,b,x0,tol)
fehler2=norm(x-B\b)
[x,iter, time]=my_jac(C,b,x0,tol)
fehler3=norm(x-C\b)
```

Für die Matrix A konvergiert das Verfahren in 35 Schritten, für die Matrizen B und C konvergiert das Verfahren nicht, was mit der früheren Aufgabe übereinstimmt.

Siehe nächstes Blatt!

3.

$$\begin{aligned} F(u) &= \frac{1}{2}u^T Au + u^T b = \frac{1}{2}\left(a_{11}u_1u_1 + a_{12}u_1u_2 + a_{13}u_1u_3\right. \\ &+ a_{21}u_2u_1 + a_{22}u_2u_2 + a_{23}u_2u_3 + a_{31}u_3u_1 + a_{32}u_3u_2 + a_{33}u_3u_3) \\ &+ u_1b_1 + u_2b_2 + u_3b_3 \end{aligned}$$

Da die Matrix A symmetrisch ist, gilt $a_{ij} = a_{ji}$. Also

$$\begin{aligned} F(u) &= \frac{1}{2}u^T Au + u^T b \\ &= \frac{1}{2}\left(a_{11}u_1u_1 + a_{22}u_2u_2 + a_{33}u_3u_3 + 2a_{12}u_1u_2 + 2a_{13}u_1u_3 + 2a_{23}u_2u_3\right) \\ &+ u_1b_1 + u_2b_2 + u_3b_3 \end{aligned}$$

Der Gradient wird gebildet durch partielles Ableiten nach den einzelnen Komponenten des Vektors u: $\nabla F = \left(\frac{\partial F}{\partial u_1}, \frac{\partial F}{\partial u_2}, \frac{\partial F}{\partial u_3}\right)^\top$

$$\begin{aligned} \frac{\partial F}{\partial u_1} &= \frac{1}{2}\left(2a_{11}u_1 + 2a_{12}u_2 + 2a_{13}u_3\right) + b_1 = a_{11}u_1 + a_{12}u_2 + a_{13}u_3 + b_1 \\ \frac{\partial F}{\partial u_2} &= \frac{1}{2}\left(2a_{12}u_1 + 2a_{22}u_2 + 2a_{23}u_3\right) + b_2 = a_{21}u_1 + a_{22}u_2 + a_{23}u_3 + b_2 \\ \frac{\partial F}{\partial u_3} &= \frac{1}{2}\left(2a_{13}u_1 + 2a_{23}u_2 + 2a_{33}u_3\right) + b_3 = a_{31}u_1 + a_{32}u_2 + a_{33}u_3 + b_3, \end{aligned}$$

also

$$\nabla F = \begin{pmatrix} a_{11}u_1 + a_{12}u_2 + a_{13}u_3 \\ a_{21}u_1 + a_{22}u_2 + a_{23}u_3 \\ a_{31}u_1 + a_{32}u_2 + a_{33}u_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = Ax + b$$

4. **a)** Um das CG-Verfahren anwenden zu können, muss die Matrix symmetrisch positiv definit sein. Die beiden Matrizen sind offensichtlich symmetrisch. Zu überprüfen ist also die positive Definitheit, was gleichbedeutend ist mit $\lambda_i > 0$ für alle Eigenwerte λ_i der Matrix A.

i) $$\begin{aligned} \det(A - \lambda I) &= (2-\lambda)^3 - (2-\lambda) - 100(2-\lambda) \\ &= (2-\lambda)(\lambda^2 - 4\lambda + 4 - 1 - 100) = (2-\lambda)(\lambda^2 - 4\lambda - 97) \\ &\Rightarrow \quad \lambda_1 = 2, \lambda_2 = \frac{4 + \sqrt{16 + 4 \cdot 97}}{2} \approx 12.05, \\ \lambda_3 &= \frac{4 - \sqrt{16 + 4 \cdot 97}}{2} \approx -8.05 \end{aligned}$$

Da ein Eigenwert negativ ist, ist die Matrix NICHT positiv definit und CG NICHT anwendbar.

ii) $$\begin{aligned} \det(A - \lambda I) &= (2-\lambda)^3 - (2-\lambda) \\ &= (2-\lambda)(\lambda^2 - 4\lambda + 4 - 1) = (2-\lambda)(\lambda^2 - 4\lambda + 3) \\ &\Rightarrow \quad \lambda_1 = 2, \lambda_2 = \frac{4 + \sqrt{16 - 12}}{2} = 3, \lambda_3 = \frac{4 - \sqrt{16 - 12}}{2} = 1 \end{aligned}$$

Alle Eigenwerte sind positiv. Also ist die Matrix positiv definit und CG ist anwendbar.

Bitte wenden!

b) Ein Schritt gemäss Skript Seite 36 mit Startvektor $u_0 = (1,0,1)^T$

$$
\begin{aligned}
u_0 &= (1,0,1)^T, r_0 = Au_0 - b = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} \\
p_1 &= -r_0 = (-1,2,-1)^T, \varrho_1 = \frac{(r_0, r_0)}{p_1, Ap_1} = \frac{3}{8} \\
u_1 &= u_0 + \varrho_1 p_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \frac{3}{8} \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix} = \frac{1}{8} \begin{pmatrix} 5 \\ 6 \\ 5 \end{pmatrix} \\
r_1 &= r_0 + \varrho_1 Ap_1 = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} + \frac{3}{8} \begin{pmatrix} -4 \\ 5 \\ -2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{8} \\ \frac{1}{4} \end{pmatrix}
\end{aligned}
$$

c) Das CG-Verfahren konvergiert spätestens nach n Schritten, hier also nach 3 Schritten.