Yannis Bähni
yannis.baehni@uzh.ch

# SOLUTIONS SHEET 3

**Exercise 1.** First, consider the *Jacobi method*. As in the script of Prof. Dr. Stefan Sauter*, I define for a nonsingular matrix $A \in M_n(\mathbb{R})$ and $b \in \mathbb{R}^n$ an abstract iteration method as

$$x^{(k+1)} = x^{(k)} - N(Ax^{(k)} - b) \qquad k = 1, 2, \dots \tag{1}$$

and $x^{(0)} := x_0 \in \mathbb{R}^n$. Further the *Iteration matrix* of such an abstract method is defined by $K := I - NA$. We have the

**Theorem 1.1.** *An iteration method of the form 1 is convergent if and only if $\varrho(K) < 1$ where $\varrho(A) := \{|\lambda| | \lambda$ eigenvalue of $A\}$ for $A \in M_n(\mathbb{R})$ is the so called* spectral radius *of $A$.*

Since the *Jacobi method* is given by

$$\begin{cases} x^{(0)} = x_0 \\ x^{(k+1)} = x_i^{(k)} - \left( \sum_{j=1}^{n} a_{ij} x_j^{(k)} - b_i \right) \Big/ a_{ii} \quad k = 1, 2, \dots \end{cases} \tag{2}$$

we get

$$\begin{bmatrix} x_1^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix} - \begin{bmatrix} a_{11}^{-1} & & \\ & \ddots & \\ & & a_{nn}^{-1} \end{bmatrix} \left( \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix} - \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \right) \tag{3}$$

For the given matrix $A$, we get

$$\begin{aligned} K_{\text{JAC}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -\frac{1}{10} & 0 \\ 0 & \frac{1}{5} \end{bmatrix} \begin{bmatrix} -10 & 2 \\ \beta & 5 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & -\frac{1}{5} \\ \frac{\beta}{5} & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \frac{1}{5} \\ -\frac{\beta}{5} & 0 \end{bmatrix} \end{aligned}$$

Further $p_{K_{\text{JAC}}} = X^2 + \frac{\beta}{25}$. Thus $\lambda_{1,2} = \pm\sqrt{-\frac{\beta}{25}}$. We get $\varrho(K_{\text{JAC}}) = \left|\sqrt{-\frac{\beta}{25}}\right|$. There are two cases to distinguish. $\underline{\beta \leqslant 0:}$ Then $\sqrt{-\frac{\beta}{25}} \in \mathbb{R}$ and so $-25 < \beta \leqslant 0$. $\underline{\beta > 0:}$ Then $\sqrt{-\frac{\beta}{25}} \in \mathbb{C}$ and $\sqrt{-\frac{\beta}{25}} = \pm i\frac{\sqrt{\beta}}{5}$. Hence we get $\left|i\frac{\sqrt{\beta}}{5}\right| = \frac{\sqrt{\beta}}{5}$. We get $\beta < 25$ in the case $\beta > 0$. In total $|\beta| < 25$ implies $\varrho(K_{\text{JAC}}) < 1$. The *Gauss-Seidel method* is given by

$$\begin{cases} x^{(0)} = x_0 \\ x^{(k+1)} = \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii} \quad k = 1, 2, \dots \end{cases} \tag{4}$$

*http://www.math.uzh.ch/index.php?file&key1=31712, last accessed March 15, 2016.

Yannis Bähni
yannis.baehni@uzh.ch

Thus, by considering the calculations provided on page 97 in the script, we have $K_{\text{GS}} = I - (D+E)^{-1}A$. Hence we get

$$\begin{aligned} K_{\text{GS}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -10 & 0 \\ \beta & 5 \end{bmatrix} \begin{bmatrix} -10 & 2 \\ \beta & 5 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{50} \begin{bmatrix} 5 & 0 \\ -\beta & -10 \end{bmatrix} \begin{bmatrix} -10 & 2 \\ \beta & 5 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -1 & \frac{1}{5} \\ 0 & -\frac{\beta}{25} - 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \frac{1}{5} \\ 0 & -\frac{\beta}{25} \end{bmatrix} \end{aligned}$$

Thus we get $\varrho(K_{\text{GS}}) = \left|\frac{\beta}{25}\right|$. It follows, that also $|\beta| < 25$ and thus in total, both method converges if and only if $|\beta| < 25$.

**Exercise 2.** Let $Ax = b$ a system of linear equations with $\det(A) \neq 0$.

- For the specified values we get

$$\begin{aligned} x_1^{(1)} &= \frac{1}{7}(3-2) = \frac{1}{7} \\ x_2^{(1)} &= \frac{1}{10}\left(4 + \frac{3}{7} - 2\right) = \frac{17}{70} \\ x_3^{(1)} &= -\frac{1}{15}\left(2 - 1 - \frac{17}{10}\right) = \frac{7}{150} \end{aligned}$$

- For the specified values we get

$$\begin{aligned} x_1^{(1)} &= \frac{1}{7}(3-2) = \frac{1}{7} \\ x_2^{(1)} &= \frac{1}{10}(4+3-2) = \frac{1}{2} \\ x_3^{(1)} &= -\frac{1}{15}(2-14)\,\frac{12}{15} \end{aligned}$$

**Exercise 3.** **a.** There is basically nothing to say. The *SOR method* is given by

$$\begin{cases} x^{(0)} = x_0 \\ x^{(k+1)} = x_i^{(k)} - \omega \left( \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij} x_j^{(k)} - b_i \right) \Big/ a_{ii} \quad k = 1, 2, \dots \end{cases} \tag{5}$$

**b.** The optimal relaxation parameter is given by

$$\omega_{\text{opt}} = 1.064981474202953 \tag{6}$$

This is the minimum of $K_{\text{SOR}}(\omega)$ for $\omega \in ]0, 2[$.

**c.** As a remark, the solution of the given system is $1 \in \mathbb{R}^7$. We get

Yannis Bähni
yannis.baehni@uzh.ch

```
function [ x0, count,err,cpu ] = Jacobi( A,b,x0,tol,maxiter,xstar )
count = 0;
tic;
while (norm(b - A * x0)/norm(b) >= tol) && (count < maxiter)
    x = x0 - diag(1./diag(A)) * (A * x0 - b);
    x0 = x;
    count = count + 1;
    err(count) = norm(xstar - x0)/norm(xstar);
    cpu(count) = toc;
end
```

Listing 1. `src/Jacobi.m`

```
function [ x0,count,err,cpu ] = GaussSeidel( A,b,x0,tol,maxiter,xstar)
[m,~] = size(A);
count = 0;
tic;
while (norm(b - A * x0)/norm(b) >= tol) && (count < maxiter)
    x = zeros(m,1);
    for i = 1:m
        x(i,1) = (b(i,1) - A(i,1:i-1) * x(1:i-1,1) - ...
            A(i,i+1:end) * x0(i+1:end,1))/A(i,i);
    end
    x0 = x;
    count = count + 1;
    err(count) = norm(xstar - x0)/norm(xstar);
    cpu(count) = toc;
end
```

Listing 2. `src/GaussSeidel.m`

$$x_{\text{JAC}} = \begin{bmatrix} 1.000000183397485 \\ 0.999999698350263 \\ 1.000000386220573 \\ 0.999999585155335 \\ 1.000000386220572 \\ 0.999999698350263 \\ 1.000000183397485 \end{bmatrix} \quad x_{\text{GS}} = \begin{bmatrix} 0.999999579248837 \\ 1.000000026856658 \\ 1.000000315083287 \\ 1.000000190483746 \\ 1.000000014283650 \\ 0.999999968023743 \\ 0.999999987754175 \end{bmatrix} \quad x_{\text{SOR}} = \begin{bmatrix} 0.999999655613379 \\ 1.000001511103757 \\ 1.000001125606356 \\ 0.999999788875074 \\ 0.999999537135906 \\ 0.999999921097913 \\ 1.000000050892852 \end{bmatrix}$$

From 2 we see, that the Jacobi method needs the most iterations whereas Gauss-Seidel and SOR are almost equal. This is due to the fact that $\omega_{\text{opt}} \approx 1$. To calculate the convegence order of the three methods, observe that the error is (almost) linear in a semi-logarithmic plot ordinate-wise. Hence the error can asymptotically be described by an *exponential function* of the form $Ca^k$ where $k \in \mathbb{N}$ are the iterations (or plotted as a continuum). Assume we have two values

Yannis Bähni
yannis.baehni@uzh.ch

```
function [ x0, count,err,cpu ] = SOR( A,b,x0,omega,tol,maxiter,xstar )
[m,~] = size(A);
count = 0;
tic;
while (norm(b - A * x0)/norm(b) >= tol) && (count < maxiter)
    x = zeros(m,1);
    for i = 1:m
        x(i,1) = x0(i,1) - omega * (A(i,1:i-1) * x(1:i-1,1)...
            + A(i,i:end) * x0(i:end,1) - b(i,1))/A(i,i);
    end
    x0 = x;
    count = count + 1;
    err(count) = norm(xstar - x0)/norm(xstar);
    cpu(count) = toc;
end
```

Listing 3. `src/SOR.m`

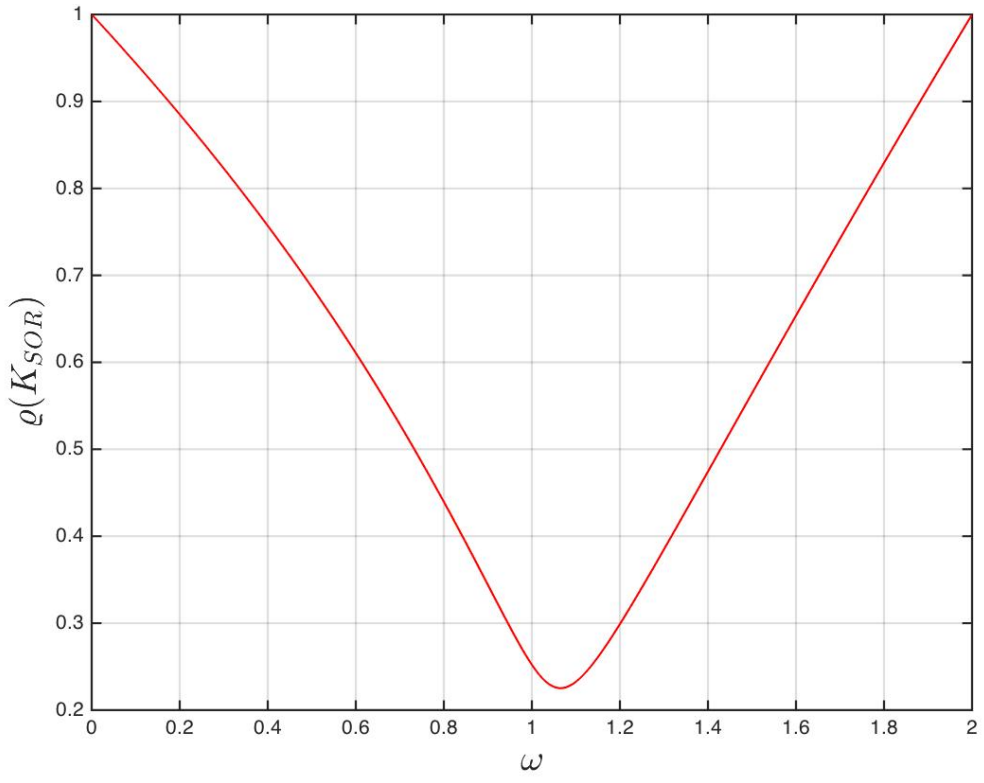


Figure 1. Plot of the spectral radius of $K_{\mathrm{SOR}} = I - \left(\frac{1}{\omega}D + E\right)^{-1} A$ on $]0, 2[$.

Yannis Bähni
yannis.baehni@uzh.ch

$$\begin{cases} \log(y_2) = \log\left(Ca^{k_2}\right) = \log(a)k_2 + \log(C) \\ \log(y_1) = \log\left(Ca^{k_1}\right) = \log(a)k_1 + \log(C) \end{cases} \tag{7}$$

Subtracting above equations yields

$$\log(a) = \frac{\log\left(\frac{y_2}{y_1}\right)}{k_2 - k_1} \tag{8}$$

or equivalently

$$\boxed{a = \exp\left(\frac{\log\left(\frac{y_2}{y_1}\right)}{k_2 - k_1}\right)} \tag{9}$$

Empirically we get for the SOR and Gauss-Seidel method an asymptotical behaviour of the error $O\left(0.25^k\right)$ and for the Jacobi method $O\left(0.78^k\right)$. I give the

**Definition 1.1.** *The* convergence order *of an iteration method of the form 1 is* $\varrho(K)$*, where* $K$ *is the iteration matrix.*

We have

$$\varrho(K_{\text{JAC}}) = 0.782253416945246 \qquad \varrho(K_{\text{GS}}) = 0.252194775253908 \tag{10}$$

So we get (almost) the expected convergence orders empirically. From 3 one observes, that the three iterative methods require almost the same amount of computational time. Remarkable is here, that the Jacobi method, which needs the most steps to reach the demanded accuracy performs almost the same with respect to time as the others. This is due to the calculations performed in the Jacobi methods, since there are less operations per iteration.

```
format long;
%Exercise 3 (b)
A = toeplitz([9,-3,1,0,0,0,0]);
SpecRad = @(omega) max(abs(eig(eye(length(A)) - ...
    inv(1./omega * diag(diag(A)) + tril(A,-1)) * A)));
val = arrayfun(@(omega) SpecRad(omega), 0.001:.001:1.999);
plot(0.001:.001:1.999, val, 'color','red');
grid on;
xlabel('$\omega$', 'interpreter','latex','fontsize',18);
ylabel('$\varrho(K_{SOR})$', 'interpreter','latex','fontsize',18);
saveas(gcf,'ex_3_b.jpg');
OmegaOpt = fminbnd(SpecRad,1,1.2)
disp('Convergence order');
JAC = max(abs(eig(eye(7) - diag(1./diag(A)) * A)));
GS = max(abs(eig(eye(7) - inv(tril(A)) * A)));
disp(JAC), disp(GS);
```

Listing 4. `src/ex_3_b.m`

Yannis Bähni
yannis.baehni@uzh.ch

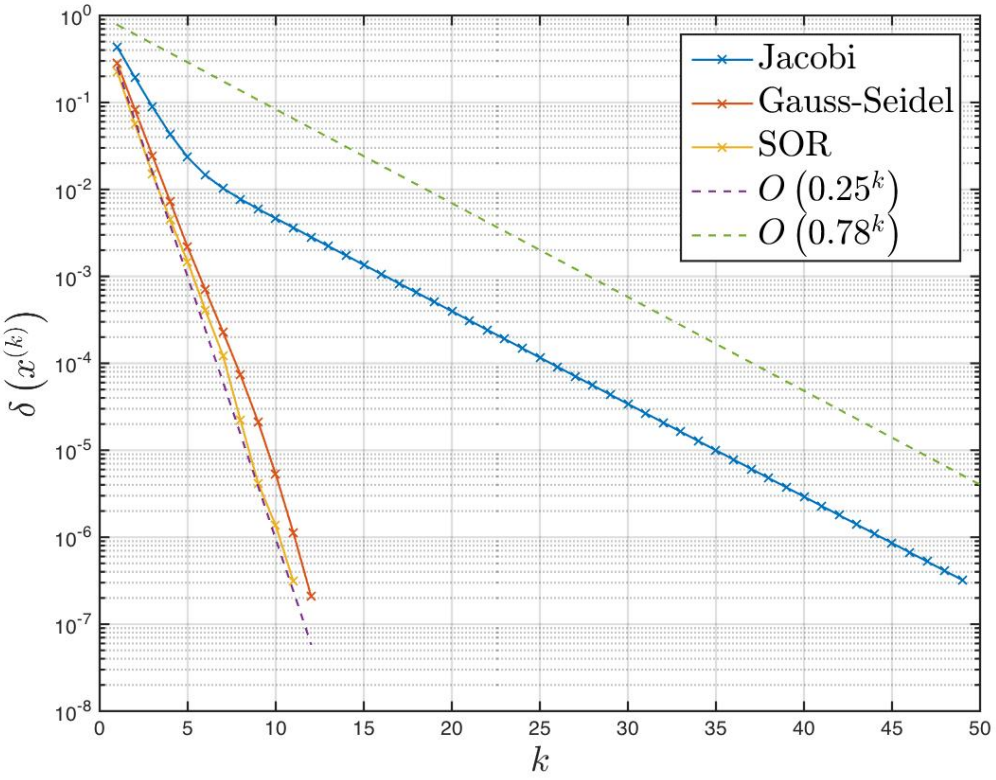


FIGURE 2. Plot of the errors of the different iterative solvers aplied to the given system $Ax = b$, where $A$ is a *Toeplitz matrix* and the convergence orders against iteration.

Yannis Bähni
yannis.baehni@uzh.ch

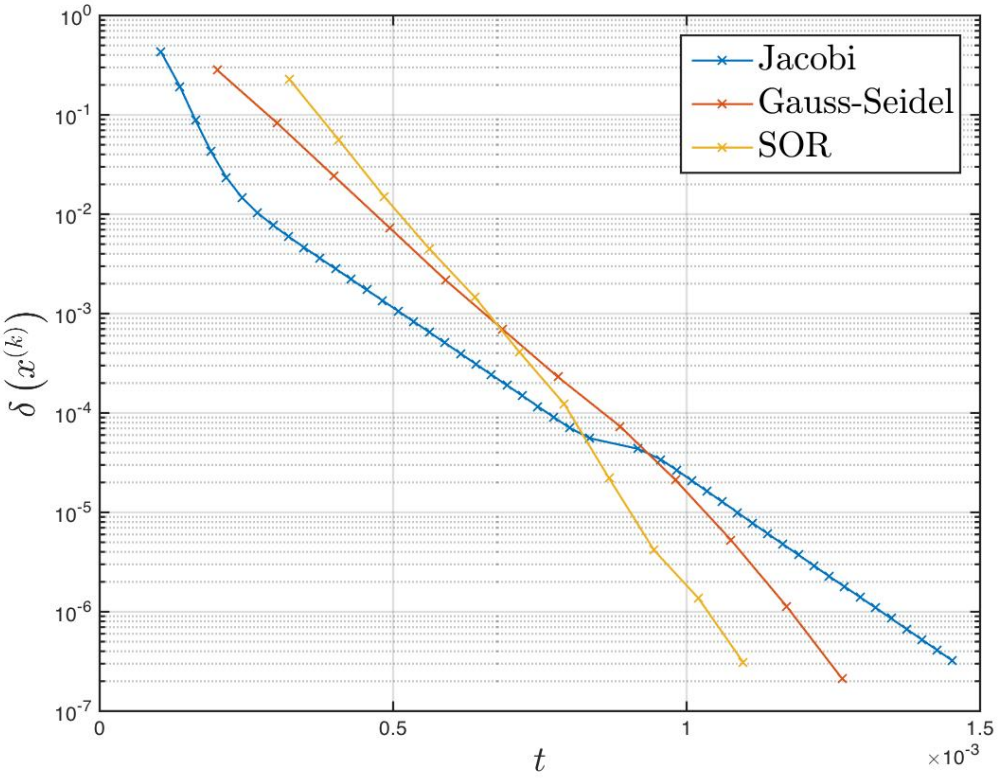


FIGURE 3. Plot of the errors of the different iterative solvers aplied to the given system $Ax = b$, where $A$ is a *Toeplitz matrix* and the convergence orders against cpu time.

Yannis Bähni
yannis.baehni@uzh.ch

```
%Exercise 3 (c)
A = toeplitz([9,-3,1,0,0,0,0]);
b = diag(fliplr(toeplitz([5,5,5,5,4,4,7])));
SpecRad = @(omega) max(abs(eig(eye(length(A)) - ...
    inv(1./omega * diag(diag(A)) + tril(A,-1)) * A)));
OmegaOpt = fminbnd(SpecRad,1,1.2);
x0 = zeros(length(A),1);
tol = 1e-6;
maxiter = 1e+3;
xstar = ones(length(A),1);
methods = {@Jacobi, @GaussSeidel};
for i = 1:length(methods)
    disp(methods{i});
    [x,~,err,cpu] = methods{i}(A,b,x0,tol,maxiter,xstar);
    iterations = 1:length(err);
    disp(exp(diff(log(err))./diff(iterations)));
    figure(1);
    semilogy(iterations,err,'-x');
    hold on;
    figure(2);
    semilogy(cpu,err,'-x');
    hold on;
end
[x,~,err,cpu] = SOR(A,b,x0,OmegaOpt,tol,maxiter,xstar);
iterations = 1:length(err);
disp(exp(diff(log(err))./diff(1:length(err))));
figure(1);
semilogy(iterations,err,'-x');
hold on;
semilogy(1:12, .25.^(1:12),'--');
hold on;
semilogy(1:50, .78.^(1:50),'--');
l = legend('Jacobi', 'Gauss-Seidel','SOR','$O\left( 0.25^k\right)$', ...
    '$O\left( 0.78^k\right)$');
set(l,'fontsize',18,'interpreter','latex');
xlabel('$k$', 'interpreter','latex','fontsize',18);
ylabel('$\delta\left(x^{(k)}\right)$', 'interpreter','latex','fontsize',18);
grid on;
saveas(gcf,'ex_3_c_err.jpg');
figure(2);
semilogy(cpu,err,'-x');
l = legend('Jacobi', 'Gauss-Seidel','SOR');
set(l,'fontsize',18,'interpreter','latex');
xlabel('$t$', 'interpreter','latex','fontsize',18);
ylabel('$\delta\left(x^{(k)}\right)$', 'interpreter','latex','fontsize',18);
grid on;
saveas(gcf,'ex_3_c_cpu.jpg');
```

Listing 5. src/ex_3_c.m