Iterative Methods for Solving Linear Systems: Jacobi, Gauss–Siedel, SOR

1. Iterative Methods

Until now, we discussed only **direct methods** for solving linear systems, i.e., LU and Cholesky for solving

$$Ax^* \;=\; b. \tag{1}$$

Here, and throughout, x^* represents the exact solution of (1).

In this section, we will introduce three **iterative methods**: Jacobi, Gauss-Siedel, and SOR. The difference between direct and iterative methods is that, for iterative methods, an approximate solution to (1) is obtained by successive iterations which are stopped when a certain convergence criterion is satisfied.

The general form of an iterative method is:

Given an initial guess x_0, find subsequent iterates from

$$x_{n+1} \;=\; Rx_n \;+\; c, \quad \forall\; n \geq 0. \tag{2}$$

Several issues need to be addressed:

- When is iteration (2) convergent?
- If iteration (2) is convergent, under what conditions, i.e., for which matrix R, does the iteration converge to the solution x^* of (1)?

We will answer these question in the subsequent sections. Meanwhile, a simple example of an iterative method is the Richardson iteration

$$x_{n+1} \;=\; x_n \;+\; \alpha(b - Ax_n), \tag{3}$$

where $\alpha \neq 0$ is a real parameter. The convergence of this iteration depends on a good choice of α; for more details, see section 8.

Note that, if the Richardson iteration (3) converges, then it will converge to the exact solution x^* of $Ax^* = b$. Assume that $x_n \to \overline{x}$. Then, as $n \to \infty$, the equation (3) becomes

$$\overline{x} \;=\; \overline{x} \;+\; \alpha(b - A\overline{x}),$$

and therefore $A\overline{x} = b$, i.e., $\overline{x} = x^*$.

The Richardson iteration can be written in the general form (2) of an iterative method as follows:

$$x_{n+1} \;=\; (I - \alpha A)x_n + \alpha b, \tag{4}$$

which is the same as $x_{n+1} = RX_n + c$ for $R = I - \alpha A$ and $c = \alpha b$.

2. Convergence Criteria for Iterative Methods

The convergence of the iteration (2) depends on the norm of R, i.e.,

$$||R|| \;=\; \max_{v \neq 0} \frac{||Rv||}{||v||},$$

and the spectral radius of R, denoted by $\rho(R)$, defined as the largest absolute value of the eigenvalues of R. In other words, if $\lambda(R)$ is the set of the eigenvalues of R, then

$$\rho(R) \;=\; \sup_{\lambda \in \lambda(R)} |\lambda|.$$

The next theorem provides a sufficient convergence criterion:

Theorem 2.1. *If there exists a norm $||\cdot||$ such that $||R|| < 1$, then the iteration*

$$x_{n+1} \; = \; Rx_n \; + \; c, \quad \forall\; n \geq 0. \tag{5}$$

is convergent, for any initial guess x_0.

Proof. Write (5) for two consecutive values n and $n+1$:

$$\begin{aligned} x_{n+1} &= Rx_n \; + \; c \\ x_{n+2} &= Rx_{n+1} \; + \; c \end{aligned}$$

By subtracting these equations from each other, we obtain

$$x_{n+2} - x_{n+1} \; = \; R(x_{n+1} - x_n), \quad \forall\; n \geq 0. \tag{6}$$

Take the norm $||\cdot||$ on both sides of (6) and use the inequality $||Cx|| \leq ||C||\;||x||$ to find that

$$||x_{n+2} - x_{n+1}|| \; \leq \; ||R||\;||x_{n+1} - x_n||, \quad \forall\; n \geq 0. \tag{7}$$

We write (7) for n, $n+1$, ..., $n+k-2$ and multiply the equations. After cancelations, we end up with

$$||x_{n+k} - x_{n+k-1}|| \; \leq \; ||R||^{k-2}\;||x_{n+1} - x_n||, \quad \forall\; n \geq 0, \;\forall\; k \geq 2.$$

□

A necessary and sufficient convergence condition for the iteration scheme (5) can be given in terms of the spectral radius:

Theorem 2.2. *The iteration scheme (5) is convergent if and only if $\rho(R) < 1$.*

3. Stopping Criteria

A general question, independent of the choice of the iterative method, i.e., of R in (2), is when do we stop the iteration. In other words, how do we decide that x_n approximates the exact solution x^* of (1) well enough?

This is a subtle question, and many algorithms err because of wrong convergence criteria, either by not obtaining a good approximation of the exact solution, or by simply not getting a truly convergent algorithm.

Let *tol* be the tolerance, i.e., the approximation precision, of the iteration. A good choice for *tol* is 10^{-6} (one order of magnitude larger or smaller are still acceptable). Two convergence criteria are usually employed, none of them requiring a priori knowledge of the exact solution x^*.

(1) *Residual–based criterion:*
Let $r_n = b - Ax_n$ be the residual corresponding to x_n, and $r_0 = b - Ax_0$ be the initial residual. We stop the iteration when $||r_n|| < tol\;||r_0||$.

Pseudocode for the residual-based stopping criterion:

```
given A, b, $x_0$, tol
$r_0$ = b - A $x_0$; r = $r_0$; stop_iter_resid = tol * norm($r_0$);
while (norm(r) > stop_iter_resid)
    compute new value of x
    r = b-Ax;        // update the residual
end
```

(2) *Consecutive Approximation criterion:*
We stop the iteration when $||x_{n+1} - x_n|| < tol$, i.e., when two consecutive approximations of the exact solution are close enough. This criterion requires keeping two such approximations in separate vectors x_{old} and x_{new}.

Pseudocode for the consecutive approximation stopping criterion:

```
given A, b, $x_0$, tol
$x_{old} = x_0$; diff = 1;        // iteration should be initialized
while (norm(diff) > tol)
    compute $x_{new}$ given $x_{old}$
    diff = $x_{new}$ - $x_{old}$;
    $x_{old}$ = $x_{new}$;
end
```

4. A General Splitting Technique

To obtain an iterative method for solving (1), i.e., $Ax = b$, we split A as

$$A = M + N,$$

where M is a nonsingular matrix. Then $Ax^* = b$ can be written as $Mx^* = -Nx^* + b$, or, equivalently, as

$$x^* = -M^{-1}Nx^* + M^{-1}b. \tag{8}$$

Based on (8), the following iteration scheme is introduced: given x_0, compute x_1, x_2, ..., using the iteration:

$$x_{n+1} = -M^{-1}Nx_n + M^{-1}b, \quad \forall\, n \geq 0.$$

This corresponds to the general form of an iterative method $x_{n+1} = Rx_n + c$, for $c = M^{-1}b$ and

$$R = -M^{-1}N.$$

We recall that there are two issues that need to be addressed for a general iterative method: when is it convergent, and does it converge to the exact solution of the original linear equation?

Lemma 4.1. *Any iterative method obtained by the previously described splitting technique, if convergent, will converge to the exact solution of the original linear equation.*

In other words, if $A = M + N$ and if the sequence $\{x_n\}_{n \geq 0}$ is convergent, where

$$x_{n+1} = -M^{-1}Nx_n + M^{-1}b, \quad \forall\, n \geq 0,$$

then

$$\lim_{n\to\infty} x_n = x^*,$$

where x^* is the exact solution of $Ax^* = b$.

Proof. Assume that $x_n \to \overline{x}$. By letting $n \to \infty$ in (8), we find that

$$\overline{x} = -M^{-1}N\overline{x} + M^{-1}b.$$

Multiplying by M and solving for $\overline{x}$, we obtain that

$$(M + N)\overline{x} = b,$$

and, since $A = M + N$, that $A\overline{x} = b$. Thus, $\overline{x} = x^*$ and $x_n \to x^*$. □

For the iteration scheme (8) to be of practical value, it must converge (and converge fast, if possible) and it must be inexpensive to implement. In other words, we need

(1) $\rho(M^{-1}N) < 1$ for convergence; as a rule of thumb, the smaller $\rho(M^{-1}N)$ is, the faster the scheme (8) would converge;
(2) a decomposition $A = M + N$ such that M is a nonsingular matrix and matrix-vector multiplication by the matrix M^{-1} is easy to evaluate.

Note that we never compute M^{-1}. Instead, we solve the associated linear systems of the form $My = v$. Possible good choices for M could be M diagonal matrix or M triangular matrix.

5. Three Iterative Methods Based on the General Splitting Technique

There are two different ways of writing a decomposition for the matrix A. Let D be the diagonal part of A, i.e., a diagonal matrix having the same entries as A on the main diagonal and zeroes elsewhere. The remaining part of A below the main diagonal is denoted by L_A and the remaining part of A above the main diagonal is denoted by U_A. Thus,

$$A = L_A + D + U_A. \tag{9}$$

If we denote the part of A below the main diagonal by $-\widetilde{L}$ and the part of A above the main diagonal by $-\widetilde{U}$, we can write A as

$$A = -\widetilde{L} + D - \widetilde{U}. \tag{10}$$

For subsequent use, we define L and U by

$$\begin{aligned} L &= D^{-1}\widetilde{L}; \\ U &= D^{-1}\widetilde{U}. \end{aligned}$$

These decompositions are the basis for the Jacobi, Gauss-Siedel and SOR iteration presented in the following sections. The decomposition (9) will be most useful in writing pseudocodes for the iterative methods, while the decomposition (10) can be used to prove their convergence.

For every iterative method listed above, we will provide three different ways of writing the iterative step $x_{n+1} = Rx_n + c$:

(1) a general iteration formula to be used in the pseudocode, based on the decomposition (9) of A;
(2) an entry by entry formula for x_{n+1};
(3) a general iteration formula for proving the convergence of the iterative method, based on the decomposition (10) of A.

5.1. The Jacobi Iteration.

The easiest possible splitting of A of the form $A = M + N$ which results in a simple to invert matrix M is to choose $M = D$ and $N = L_A + U_A$. The resulting iteration scheme is called the Jacobi iteration.

5.1.1. *Jacobi: Recursion for Pseudocode.* .

Let $M = D$ and $N = L_A + U_A$ be the decomposition of $A = M + N$. From (8), it follows that the Jacobi iteration can be written as

$$x_{n+1} = -D^{-1}(L_A x_n + U_A x_n) + D^{-1}b. \tag{11}$$

For the pseudocode implementing the Jacobi iteration we use the residual-based stopping criterion from section 3 and (11). We need a routine mult_inv_D(x) to compute $D^{-1}x$ given x, (very cheap, since D is a diagonal matrix), and the routines mult_L_A(x) and mult_U_A(x) to compute $L_A x$ and $U_A x$ given x.

Pseudocode for Jacobi iteration:

```
given A, b, $x_0$, tol
$x = x_0$; $r_0$ = b - A $x_0$; r = $r_0$; stop_iter_resid = tol * norm($r_0$);
```

```
b_new = mult_inv_D(b);        // compute only once before the for loop
ic = 0;               // set iteration count to 0
while (norm(r) > stop_iter_resid)
     x = - mult_inv_D( mult_L_A(x) + mult_U_A(x) ) + b_new;
     r = b-Ax;          // update the residual
     ic = ic + 1;        // increase iteration count
end
```

5.1.2. *Jacobi: Entry by Entry Recursion.* .

Another way to express the dependence of x_{n+1} on x_n given by (11) is entry by entry. If $x_n \in R^p$, then $x_n = (x_n(1), x_n(2), \ldots, x_n(p))^T$ and iteration (11) can be written in vector form as

for $j = 1 : p$

$$x_{n+1}(j) \;=\; -\frac{1}{A(j,j)}\left(\sum_{k=1}^{j-1} A(j,k)x_n(k) \;+\; \sum_{k=j+1}^{p} A(j,k)x_n(k)\right) \;+\; \frac{b(j)}{A(j,j)} \tag{12}$$

end

5.1.3. *Jacobi: Recursion for Convergence Proof.* .

If we write $A = -\widetilde{L} + D - \widetilde{U}$, then the decomposition of A corresponding to the Jacobi iteration is $M = D$ and $N = -\widetilde{L} - \widetilde{U}$. From (8), it follows that the Jacobi iteration can be written as:

$$x_{n+1} \;=\; D^{-1}(\widetilde{L}x_n + \widetilde{U}x_n) + D^{-1}b. \tag{13}$$

Let L and U be defined by $L = D^{-1}\widetilde{L}$ and $U = D^{-1}\widetilde{U}$. Then $\widetilde{L} = DL$ and $\widetilde{U} = DU$ and (13) can be written as

$$x_{n+1} \;=\; (L+U)x_n + D^{-1}b, \tag{14}$$

or, using the general notation (2) for an iterative method, as

$$x_{n+1} \;=\; R_J x_n + b_J,$$

with $b_J = D^{-1}b$ and

$$R_J \;=\; L + U. \tag{15}$$

5.2. **The Gauss–Siedel Iteration.** Another matrix which is easy to invert is the lower triangular part of A. This corresponds to a splitting $A = M + N$ with $M = D + L_A$ and $N = U_A$. The resulting iteration scheme is called the Gauss–Siedel iteration.

5.2.1. *Gauss–Siedel: Recursion for Pseudocode.* .

Let $M = D + L_A$ and $N = U_A$ be the decomposition of $A = M + N$. From (8), it follows that the Gauss–Siedel iteration can be written as

$$x_{n+1} \;=\; -(D+L_A)^{-1}U_A x_n + (D+L_A)^{-1}b. \tag{16}$$

For the pseudocode implementing the matrix formulation (16) of the GS iteration, we use the residual–based stopping criterion from section 3. Let mult_U_A(x) be the routine which computes $U_A x$ given x, and let forward_subst(Q,z) be the forward substitution routine returning the solution y to the linear problem $Qy = z$, where Q is a lower triangular matrix.

Pseudocode for Gauss–Siedel iteration:

```
given A, b, $x_0$, tol
$x = x_0$; $r_0$ = b - A $x_0$; r = $r_0$; stop_iter_resid = tol * norm($r_0$);
b_new = forward_subst($D + L_A$, b);     // compute only once before the for loop
```

```
ic = 0;                  // set iteration count to 0
while (norm(r) > stop_iter_resid)
     x = - forward_subst($D + L_A$, mult_U_A(x)) + b_new;
     r = b-Ax;           // update the residual
     ic = ic + 1;        // increase iteration count
end
```

5.2.2. *Gauss–Siedel: Entry by Entry Recursion.* .

The entry by entry recursion corresponding to (16) is

for $j = 1 : p$

$$x_{n+1}(j) \;=\; -\frac{1}{A(j,j)}\left(\sum_{k=1}^{j-1} A(j,k)x_{n+1}(k) \;+\; \sum_{k=j+1}^{p} A(j,k)x_n(k)\right) \;+\; \frac{b(j)}{A(j,j)} \tag{17}$$

end

The difference between (12) and (17) resides in the fact that, when computing $x_{n+1}(j)$ in the GS iteration, we use the latest values for the entries $1 : (j-1)$ of the approximate solution of the linear system, i.e., $x_{n+1}(k)$, with $k = 1 : (j-1)$.

5.2.3. *Gauss–Siedel: Recursion for Convergence Proof.* .

If we write $A = -\widetilde{L} + D - \widetilde{U}$, then the decomposition of A corresponding to the GS iteration is $M = D - \widetilde{L}$ and $N = -\widetilde{U}$. Recall that $L = D^{-1}\widetilde{L}$ and $U = D^{-1}\widetilde{U}$, and therefore $\widetilde{L} = DL$ and $\widetilde{U} = DU$. From (8), it follows that the GS iteration can be written as:

$$\begin{aligned} x_{n+1} &= (D-\widetilde{L})^{-1}\widetilde{U}x_n + (D-\widetilde{L})^{-1}b \\ &= (D-DL)^{-1}DUx_n + (D-DL)^{-1}b \\ &= (I-L)^{-1}Ux_n + (I-L)^{-1}D^{-1}b. \end{aligned}$$

We conclude that

$$x_{n+1} \;=\; (I-L)^{-1}Ux_n + (I-L)^{-1}D^{-1}b, \tag{18}$$

or, using the general notation (2) for an iterative method, that

$$x_{n+1} \;=\; R_{GS}x_n + b_{GS},$$

with $b_{GS} = (I-L)^{-1}D^{-1}b$ and

$$R_{GS} \;=\; (I-L)^{-1}U. \tag{19}$$

5.3. **The SOR Iteration.** In general (and rather vague) terms, we can describe the Successive Overrelaxation Iteration (SOR) as a weighted average of the SOR approximation at step n and the Gauss–Siedel approximation at step $n+1$. The SOR iteration depends on a real parameter ω for the aforementioned weighted average.

We note that SOR and GS are identical if $\omega = 1$. In Theorem 6.1 we show that, for the SOR iteration to be convergent, it is necessary to have $0 < \omega < 2$.

The SOR iteration will be made precise in the following section.

5.3.1. *SOR: Entry by Entry Recursion.* .

The idea behind the Successive Overrelaxation Iteration (SOR) is easiest to understand if we look at the entry by entry recursions for Jacobi (12) and Gauss-Siedel (17).

The GS iteration was obtained from the Jacobi iteration by using the updated entries $x_{n+1}(1)$, $\dots$, $x_{n+1}(j-1)$ of the approximate solution x_{n+1} when computing $x_{n+1}(j)$. For SOR, we want to make a correction to this entry by computing a weighted average of the value of $x_{n+1}(j)$ computed from Gauss–Siedel and the value $x_n(j)$ corresponding to the previous iteration step, i.e.,

$$x_{n+1,SOR}(j) \;=\; (1-\omega)x_{n,SOR}(j) \;+\; \omega x_{n+1,GS}(j).$$

For the SOR iteration to be convergent, ω must be in the interval $(0,2)$. If $\omega = 1$, then the SOR iteration coincides with the GS iteration. If $0 < \omega < 1$, the SOR method is actually called successive underrelaxation, while if $1 < \omega < 2$ it is called overrelaxation.

The entry by entry dependence of x_{n+1} on x_n is given by:

for $j = 1 : p$

$$x_{n+1}(j) \;=\; (1-\omega)x_n(j) \;-\; \frac{\omega}{A(j,j)}\left(\sum_{k=1}^{j-1} A(j,k)x_{n+1}(k) \;+\; \sum_{k=j+1}^{p} A(j,k)x_n(k)\right) \;+\; \frac{\omega b(j)}{A(j,j)} \tag{20}$$

end

5.3.2. *SOR: Recursion for Pseudocode.* .

For SOR, the decomposition of A of the form $A = M + N$ is no longer obvious, as was the case for Jacobi and GS.

By multiplying the entry-by-entry recursion formula (20) by $A(j,j)$, we obtain

$$A(j,j)x_{n+1}(j) \;=\; (1-\omega)A(j,j)x_n(j) \;-\; \omega\left(\sum_{k=1}^{j-1} A(j,k)x_{n+1}(k) \;+\; \sum_{k=j+1}^{p} A(j,k)x_n(k)\right) \;+\; \omega b(j),$$

which can be written in matrix form as

$$Dx_{n+1} \;=\; (1-\omega)Dx_n \;-\; \omega(L_A x_{n+1} + U_A x_n) \;+\; \omega b.$$

By solving for x_{n+1} we find that

$$x_{n+1} \;=\; (D+\omega L_A)^{-1}\;\left((1-\omega)D - \omega U_A\right)x_n \;+\; \omega(D+\omega L_A)^{-1}b. \tag{21}$$

For the pseudocode implementing the matrix formulation (21) of the SOR iteration, we use the residual–based stopping criterion from section 3. Let mult_U_A(x) and mult_D(x) be the routines which compute $U_A x$ and Dx given x, respectively, and let forward_subst(Q,z) be the forward substitution routine returning the solution y to the linear problem $Qy = z$, where Q is a lower triangular matrix.

Pseudocode for the SOR iteration:

```
given A, b, $x_0$, tol, $\omega \in (0,\ 2)$
$x = x_0$; $r_0$ = b - A $x_0$; r = $r_0$; stop_iter_resid = tol * norm($r_0$);
b_new = $\omega$ forward_subst($D+\omega L_A$, b);     // compute only once before the for loop
ic = 0;                  // set iteration count to 0
while (norm(r) > stop_iter_resid)
     x = forward_subst($D+\omega L_A$, (1-$\omega$) mult_D(x) - $\omega$ mult_U_A(x)) + b_new;
     r = b-Ax;           // update the residual
     ic = ic + 1;        // increase iteration count
end
```

5.3.3. *SOR: Recursion for Convergence Proof.* .

Once again, we write A as $A = -\widetilde{L} + D - \widetilde{U}$. Recall that $\widetilde{L} = -L_A$ and $\widetilde{U} = -U_A$ and therefore

$$\begin{aligned} L &= D^{-1}\widetilde{L} = -D^{-1}L_A; \\ U &= D^{-1}\widetilde{U} = -D^{-1}U_A. \end{aligned}$$

We can write (21) as

$$x_{n+1} = (I + \omega D^{-1}L_A)^{-1} \left((1-\omega)I - \omega D^{-1}U_A\right) x_n + \omega(I + \omega D^{-1}L_A)^{-1}D^{-1}b$$

and therefore

$$x_{n+1} = (I - \omega L)^{-1} \left((1-\omega)I + \omega U\right) x_n + \omega(I - \omega L)^{-1}D^{-1}b. \tag{22}$$

Using the general notation (2) for an iterative method, we can express (22) as

$$x_{n+1} = R_{SOR}x_n + b_{SOR},$$

with $b_{SOR} = \omega(I - \omega L)^{-1}D^{-1}b$ and

$$R_{SOR} = (I - \omega L)^{-1} \left((1-\omega)I + \omega U\right). \tag{23}$$

6. Convergence of the Jacobi, Gauss–Siedel and SOR Iterations

Up to this point, we did not discuss for which matrices do the Jacobi, Gauss–Siedel and SOR iterations converge. First of all, it is clear that they cannot even be applied to arbitrary linear systems $Ax = b$. For example, Jacobi and GS involve D^{-1}. If one of the entries on the main diagonal of A is 0, the matrix D is not invertible and Jacobi and GS do not exist.

Before we move further, we show that $\omega \in (0, 2)$ is a necessary condition for the convergence of the SOR iteration.

Theorem 6.1. *If an SOR iteration is convergent, then*

$$0 < \omega < 2.$$

Proof. We recall from (23) that

$$R_{SOR} = (I - \omega L)^{-1} \left((1-\omega)I + \omega U\right).$$

It is easy to see that the determinant of R_{SOR} is

$$det(R_{SOR}) = (1-\omega)^p, \tag{24}$$

where p is the dimension of the matrix A. Since the determinant of any matrix is equal to the product of its eigenvalues, it follows that the largest eigenvalue in absolute value of R_{SOR}, denoted by $\lambda_{max}(R_{SOR})$, satisfies:

$$|det(R_{SOR})| = \prod_{i=1}^{p} |\lambda_i(R_{SOR})| \leq |\lambda_{max}(R_{SOR})|^p. \tag{25}$$

Recall that $\rho(R_{SOR}) = |\lambda_{max}(R_{SOR})|$. From (24) and (25) we find that

$$\rho(R_{SOR}) = |\lambda_{max}(R_{SOR})| \geq |1-\omega|.$$

We know from Theorem 2.2 that for convergence it is necessary (and sufficient, though we do not use that part here) to have

$$\rho(R_{SOR}) < 1.$$

Thus, $|1-\omega| < 1$, which is equivalent to $0 < \omega < 2$. □

A general convergence proof (which, however, does not tell us anything about how to choose ω for fastest convergence) exists for spd matrices.

Theorem 6.2. *If A is symmetric positive definite, then SOR converges for $\omega \in (0, 2)$. In particular, for $\omega = 1$, we obtain that GS is convergent as well.*

We are interested in applying Jacobi, GS, and SOR to matrices arising, e.g., from finite difference or finite element discretization of partial differential equations. These matrices are sparse, banded, and, most importantly, often have much larger elements on the main diagonal than anywhere else. To make this precise, we use the following definition:

Definition 6.3. The $p \times p$ matrix A is called weakly diagonally dominant if

$$|A(j,j)| \;\geq\; \sum_{k=1}^{j-1} |A(j,k)| \;+\; \sum_{k=j+1}^{p} |A(j,k)|, \quad \forall\, j = 1:p.$$

The matrix A is called *strictly* diagonally dominant if

$$|A(j,j)| \;>\; \sum_{k=1}^{j-1} |A(j,k)| \;+\; \sum_{k=j+1}^{p} |A(j,k)|, \quad \forall\, j = 1:p.$$

For strictly diagonally dominant matrices, it is not difficult to obtain the following convergence result:

Theorem 6.4. *If A is strictly diagonally dominant, then both the Jacobi and the Gauss–Siedel iterations converge, since*

$$||R_{GS}||_\infty \;\leq\; ||R_J||_\infty \;<\; 1.$$

Note that the result above does not imply that the Gauss–Siedel iteration will converge faster than the Jacobi iteration for any strictly diagonally dominant matrix A!

Since most of the matrices we will encounter will only be weakly diagonally dominant, we need to introduce new criteria for the convergence of iterative methods under weaker assumptions than those from Theorem 6.4.

Definition 6.5. The matrix A is called irreducible if the graph of A is strongly connected or, equivalently, if there is no permutation matrix P such that PAP^T has an upper block triangular structure.

Theorem 6.6. *If A is irreducible and weakly diagonally dominant, then Jacobi and Gauss–Siedel converge and*

$$\rho(R_{GS}) \;\leq\; \rho(R_J) \;<\; 1.$$

The next theorem indicates how should ω be chosen for SOR for fastest convergence.

Theorem 6.7. *If A is consistently ordered (e.g., comes from a Red–Black ordering), then*

$$\rho(R_{GS}) \;=\; \rho(R_J)^2.$$

Let

$$\omega_{opt} \;=\; \frac{2}{1 + \sqrt{1 - \rho(R_J)^2}}.$$

Then the SOR iteration corresponding to ω_{opt} satisfies

$$\rho(R_{SOR}) \;=\; \omega_{opt} - 1.$$

7. Decreasing the approximation error in an iteration

Let

$$x_{n+1} \ = \ Rx_n + c, \quad \forall\ n \geq 0 \tag{26}$$

be a convergent iterative method, i.e., cf. Theorem 2.2,

$$\rho(R) \ < \ 1.$$

Assume that the iteration (26) converges to the exact solution x^* of the linear problem $Ax^* = b$.

We want to find the connection between the speed of convergence of the iterative method (26) and the size of the spectral radius $\rho(R)$. In other words, the question we will answer in this section is:

How fast will the norm of the approximation error $x_n - x^*$ decrease by one order of magnitude?

We first need to understand the question that is being asked. To formalize it, note that since $x_n \to x^*$ as $n \to \infty$,

$$x^* \ = \ Rx^* + c. \tag{27}$$

By subtracting (27) from (26), we find that

$$x_{n+1} - x^* \ = \ R(x_n - x^*), \quad \forall\ n \geq 0.$$

We take norms on both sides and use the inequality $||Mv|| \leq ||M||\ ||v||$ to obtain

$$||x_{n+1} - x^*|| \ \leq \ ||R||\ ||x_n - x^*||, \quad \forall\ n \geq 0. \tag{28}$$

Writing (28) for n, $n+1$, ..., $n+s-1$, and multiplying the corresponding inequalities, we conclude that

$$||x_{n+s} - x^*|| \ \leq \ ||R||^s\ ||x_n - x^*||, \quad \forall\ n \geq 0,\ \forall\ s \geq 0. \tag{29}$$

We are going to replace $||R||$ in (29) by $\rho(R)$. This approximation is actually precise if R is a symmetric matrix and the norm is the 2-norm, i.e., $||R||_2 = \rho(R)$ for R symmetric. In general, there exists a norm $||\cdot||$ such that, for any $\epsilon > 0$, $||R|| < \rho(R) + \epsilon$. Thus, we infer from (29) that

$$\frac{||x_{n+s} - x^*||}{||x_n - x^*||} \ \leq \ \rho(R)^s, \quad \forall\ n \geq 0,\ \forall\ s \geq 1. \tag{30}$$

Our question can be rephrased as: find s such that

$$\frac{||x_{n+s} - x^*||}{||x_n - x^*||} \ \leq \ \frac{1}{10}, \quad \forall\ n \geq 0. \tag{31}$$

From (30) and (31) it follows that it is enough to find s such that

$$\rho(R)^s \ \leq \ \frac{1}{10}.$$

This is equivalent to

$$s \ \geq \ \frac{-\ln 10}{\ln \rho(R)}. \tag{32}$$

We recall that the Taylor expansion of $\ln(1-x)$ is

$$\ln(1-x) \ = \ -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots.$$

Therefore,

$$\ln \rho(R) \ = \ \ln(1 - (1 - \rho(R))) \ \leq \ -(1 - \rho(R)) \tag{33}$$

From (32) and (33) we conclude that it is enough for s to satisfy

$$s \ \geq \ \frac{\ln 10}{1 - \rho(R)}$$

in order to decrease the approximation error by an order of magnitude. Thus, the following result was proved:

Theorem 7.1. *Assume that the iterative method* $x_{n+1} = Rx_n + c$ *is convergent and* $x_n \to x^*$ *as* $n \to \infty$*. Then the approximation error decreases by at least an order of magnitude every* $\ln 10/(1-\rho(R))$ *iterations, i.e.,*

$$\frac{||x_{n+s} - x^*||}{||x_n - x^*||} \le \frac{1}{10}, \quad \forall\, n \ge 0,$$

if

$$s \ge \frac{\ln 10}{1-\rho(R)}.$$

8. Convergence Analysis of the Richardson Iteration

We now return to the Richardson iteration which was mentioned as a first example of an iterative method in section 1:

$$x_{n+1} = x_n + \alpha(b - Ax_n), \tag{34}$$

where α is a real parameter. We can rewrite the Richardson iteration in the general form (2) as:

$$x_{n+1} = (I - \alpha A)x_n + \alpha b, \tag{35}$$

with $R = I - \alpha A$ and $c = \alpha b$.

Recall that, if the Richardson iteration (3) converges, then it will converge to the exact solution x^* of $Ax^* = b$.

In this section, we discuss how to choose the parameter α in order to obtain a convergent Richardson iteration and find an optimal value for α.

For simplicity, assume that A is a $p \times p$ symmetric positive definite matrix. Thus, all the eigenvalues of A are real and positive. Let

$$0 < \lambda_1 \le \lambda_2 \le \ldots \le \lambda_p$$

be the eigenvalues of A.

It is easy to see that the eigenvalues of $R = I - \alpha A$ are

$$\mu_j = 1 - \alpha\lambda_j, \quad \text{for} \quad j = 1 : p.$$

For convergence it is necessary and sufficient to have $\rho(R) < 1$; cf. Theorem 2.2. If $\alpha < 0$, then $1 < \mu_1 < \mu_p$. Therefore $\rho(R) = \mu_p > 1$ and the Richardson iteration would not converge. If $\alpha \ge 0$, then

$$\mu_p \le \mu_{p-1} \le \ldots \le \mu_1$$

and therefore

$$\rho(R) = \max_{j=1:p} |\mu_j| = \max\{|\mu_1|, |\mu_p|\}.$$

The best possible value for α corresponds to the smallest possible value for $\rho(R)$ which is achieved when $|\mu_1| = |\mu_p|$. Then $\mu_p < 0 < \mu_1$ and $|\mu_1| = |\mu_p|$ is equivalent to $-\mu_p = \mu_1$, i.e., $\alpha\lambda_p - 1 = 1 - \alpha\lambda_1$. Solving for α, we obtain

$$\alpha_{opt} = \frac{2}{\lambda_1 + \lambda_p}. \tag{36}$$

We compute the corresponding value for $\rho(R)$, using the fact that $\rho(R) = \mu_1 = -\mu_p$ for α_{opt} satisfying (36), and obtain

$$\rho(R) = 1 - \alpha_{opt}\lambda_1 = \frac{\lambda_p - \lambda_1}{\lambda_p + \lambda_1}.$$

If A is spd, then $||A||_2 = \lambda_p$ and $||A^{-1}||_2 = 1/\lambda_1$. We find that

$$\kappa(A) \;=\; cond_2(A) \;=\; ||A||_2\, ||A^{-1}||_2 \;=\; \frac{\lambda_p}{\lambda_1}.$$

We conclude that

$$\rho(R) \;=\; \frac{\lambda_p/\lambda_1 - 1}{\lambda_p/\lambda_1 + 1} \;=\; \frac{\kappa(A) - 1}{\kappa(A) + 1} \;<\; 1$$

for the optimal choice of α_{opt}

$$\alpha_{opt} \;=\; \frac{2}{\lambda_1 + \lambda_p}.$$

One final note: the recipe for choosing the optimal α is based on previous knowledge of the largest and smallest eigenvalues of A. The largest eigenvalue of A can be found using the Power Method for A. This routine only involves matrix-vector multiplication by the matrix A.

However, computing the smallest eigenvalue of A requires to use the Power Method for A^{-1}. At each step of the power method iteration a matrix-vector multiplication by A^{-1} is required. This is equivalent to solving a linear system corresponding to A at each step, but this was the original problem that we were trying to solve using the Richardson iteration itself.

In other words, finding the optimal value of α is not practical in general, unless extra information is known about the matrix A a priori.