EXAMPLES OF CLASSICAL ITERATIVE METHODS

In these lecture notes we revisit a few classical fixpoint iterations for the solution of the linear systems of equations. We focus on the algebraic and algorithmic ideas underlying those fixpoint methods, since understanding those gives the framework and orientation for any further going (often scattered) material in the literature.

The two most important ideas that become apparent in the following exposition are algebraic and algorithmic approaches in developing fixpoint methods:

- On the algebraic level, a fixpoint method is found by choosing a preconditioner P such that $P^{-1} \approx A^{-1}$. Any fixpoint method is uniquely defined by its preconditioner.
- On the algorithmic level, a fixpoint method can be found from another fixpoint method by rearranging the calculations, as is the derivation of the Gauss-Seidel method from the Jacobi method.

Furthermore, there are a few techniques of building new fixpoint methods from any existing fixpoint method; we mention explicitly *composition* and *relaxation*.

We do not dig too deep into convergence results. On the one hand, many of these fixpoint methods are often applied without further thought by practitioners. On the other hand, theoretical convergence results are readily available only for subclasses of matrices. Most of these results are motivated by applications in numerical partial differential equations: *finite difference methods* and *finite element methods* have produced large classes of matrices that have received much attention in theoretical numerical analysis.

It is sometimes claimed that the classical fixpoint iterations are only of historical interest today and have been superseded by other methods. That is not true; however, their usage has changed over time and mostly will change in the future. For example, the SOR method and SSOR method used to be the standard methods of choice for solving linear systems of equations in numerical partial differential equations. Though they have been superseded by other methods, they still remain important when used as *preconditioners themselves* within more complex numerical methods.

On the other hand, the relevance of numerical methods may change over time when technologies change: for example, even though the Gauss-Seidel method converges in many applications faster than the Jacobi method, the Gauss-Seidel method is intrinsically sequential. By contrast, the Jacobi method is almost trivially to parallelize.

1. Richardson Method

We begin this exposition by recalling the definition of the Richardson method on an algebraic level. The idea of the Richardson Method is to approximate the

matrix A by a single number $\theta \in \mathbb{R}$. Heuristically, $A \approx \theta$, so $A^{-1} \approx \theta^{-1}$. This works well if the matrix is indeed described, to some extent, by a single scalar.

Theorem 1
There exists a parameter $\theta \in \mathbb{C} \setminus 0$ such that the Richardson method

$$x^{(k+1)} := x^{(k)} + \theta^{-1} \left(b - Ax^{(k)} \right)$$

converges for any starting value $x^{(0)}$ if and only if all eigenvalues of A are contained within an open disk $D \subset \mathbb{C}$ that excludes the origin.

Proof. See homework. □

Theorem 2
Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. We can improve our initial convergence bound.

(1) Suppose that the eigenvalues of A are contained the positive interval $[a, b] \subset \mathbb{R}^+$. Show that the Richardson iteration converges with parameter $\theta = \frac{1}{2}(b + a)$.
(2) Let $\lambda_{\min} > 0$ and $\lambda_{\max} > 0$ be the smallest and the largest eigenvalues of A, respectively. Show that the spectral radius of the Richardson method's iteration matrix is minimal among all $\theta > 0$ for $\theta = \frac{1}{2}(\lambda_{\max} + \lambda_{\min})$.

Remark 1
For a positive definite matrix A, we thus can always estimate $0 \leq \lambda_{\min}$ and use upper bound for $\lambda_{\max}$ computed from Gerschgorin circles. This gives a method that, if nothing else, converges for any starting value.

Since these lectures will repeatedly dive into pseudocodes, we recall a pseudocode for the Richardson iteration as a simple warm-up. The Richardson iteration is described by:

FOR $i = 1, \ldots, n$ DO
  $y_i = \theta^{-1} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)$
END FOR
FOR $i = 1, \ldots, n$ DO
  $x_i = x_i + y_i$
END FOR

Example 1
To further improve our feeling for this fixpoint method, and as a preparation for the following methods, let us suppose that $A = \theta \operatorname{Id} + E$ where $\theta \in \mathbb{R}$ is non-zero and $E \in \mathbb{R}^{n \times n}$ is "small". In other words, let us suppose that A is the same as scaling by the non-zero parameter θ and a small linear perturbation E.

Then it is a reasonable guess that $A^{-1} \approx \theta^{-1}$. Whence, if we multiply the residual $b - Ax^{(k)}$ by the reciprocal θ^{-1}, then we should get a good approximation to the error $x^* - x^{(k)} = A^{-1}b - x^{(k)}$.

2. Splitting Methods, General Form

Many iterative schemes are based on splitting the matrix A into two parts $A = M + P$. We may think of P as the "dominant" part of A, and we use this part of the matrix as a preconditioner for a fixpoint iteration.

Suppose that $A = P + M$ with P invertible. We then find

$$\begin{aligned} T(x) &= x + P^{-1}(b - Ax) \\ &= x + P^{-1}(b - Mx - Px) \\ &= x + P^{-1}(b - Mx) - P^{-1}Px \\ &= x + P^{-1}(b - Mx) - x \\ &= P^{-1}(b - Mx). \end{aligned}$$

The iteration matrix of this fixpoint mapping is

$$\mathcal{H} = I - P^{-1}A = I - P^{-1}M - P^{-1}P = -P^{-1}M.$$

Example 2
Quite often, the preconditioner P is a part of the matrix, such as the diagonal part, and the matrix is literally split in terms of its entries. However, this explicitness of the splitting is not necessary.

Though it may be a moot point, let us interpret the Richardson iteration in terms of splitting the matrix A. We have $A = \theta \operatorname{Id} + M$ with $M = A - \theta \operatorname{Id}$, and the fixpoint mapping of the Richardson iteration can be written equivalently

$$T(x) = \theta^{-1}\left(b - Ax + \theta x\right).$$

In particular, this is an example where the preconditioner typically is not a "subpart" of the matrix.

3. Jacobi and Gauss-Seidel Methods

Some of the most well-known fixpoint iterations can be interpreted very naturally as splitting methods. We split the matrix $A \in \mathbb{R}^{n\times n}$ into

$$A = L + D + R.$$

Here, $L \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{n\times n}$ are the strict lower and strict upper triangular parts of A, respectively, and $D \in \mathbb{R}^{n\times n}$ is the diagonal part of A.

Example 3
The **Jacobi method** uses

$$A = P + M, \quad P = D, \quad M = L + R.$$

The resulting method is

$$\begin{aligned} T_{Jac}(x) &= x + D^{-1}(b - Ax) \\ &= (\mathrm{Id} - D^{-1}A)x + D^{-1}b \\ &= D^{-1}\left(b - Lx - Rx\right). \end{aligned}$$

The Jacobi method approximates the matrix A by its diagonal part D, that is, $A \approx D$, and hence $A^{-1} \approx D^{-1}$.

Example 4
The (forward) **Gauss Seidel method** uses

$$A = P + M, \quad P = L + D, \quad M = R.$$

The resulting method is

$$\begin{aligned} T_{GS,l}(x) &= x + (L + D)^{-1}(b - Ax) \\ &= (\mathrm{Id} - (L + D)^{-1}A)x + (L + D)^{-1}b \\ &= (L + D)^{-1}\left(b - Rx\right). \end{aligned}$$

The Gauss-Seidel approximates A by its lower triangular part $L+D$. With $L+D \approx A$ we get $A^{-1} \approx (L + D)^{-1}$.

Example 5
Conversely, the backward **Gauss-Seidel method** uses

$$A = P + M, \quad P = D + R, \quad M = L.$$

The resulting method is

$$\begin{aligned} T_{GS,r}(x) &= x + (D + R)^{-1}(b - Ax) \\ &= (\mathrm{Id} - (D + R)^{-1}A)x + (D + R)^{-1}b \\ &= (D + R)^{-1}\left(b - Lx\right). \end{aligned}$$

In this method, we approximate A by its upper triangular part $D + R$. With $D + R \approx A$ we get $A^{-1} \approx (D + R)^{-1}$.

3.1. **Pseudocodes.** It is illustrative to inspect pseudocodes associated with these fixpoint methods. For the Jacobi method, the code splits into two FOR loops and an auxiliary vector: we first compute the auxiliary vector $y = b - Lx - Rx$, and then solve $x = D^{-1}y$.

FOR $i = 1, \ldots, n$ DO
  $y_i = b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j$
END FOR
FOR $i = 1, \ldots, n$ DO
  $x_i = a_{ii}^{-1}y_i$
END FOR

Equivalently, with slight rearrangement of the computation

FOR $i = 1, \ldots, n$ DO

$y_i = a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right)$
END FOR
FOR $i = 1, \ldots, n$ DO
$x_i = y_i$
END FOR

For the Gauss-Seidel method, the first FOR loop computes the vector $y = b - Rx$, and the second FOR loop uses forwards substitution to solve $(L + D)x = y$:

FOR $i = 1, \ldots, n$ DO
$y_i = b_i - \sum_{j=i+1}^{n} a_{ij}x_j$
END FOR
FOR $i = 1, \ldots, n$ DO
$x_i = a_{ii}^{-1}\left(y_i - \sum_{j=1}^{i-1} a_{ij}y_j\right)$
END FOR

Note that this pseudocode for the Gauss-Seidel method is analogous to the first pseudocode for the Jacobi method. At this point we observe that the value of y_i and the new value of x_i do not depend on $x_1, \ldots, x_{i-1}$. So we can save some memory and overwrite the old entries of x with the new values, leading to the algorithm

FOR $i = 1, \ldots, n$ DO
$x_i = a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right)$
END FOR

This gives the following algorithmic motivation for the (forward) Gauss-Seidel method: we improve the Jacobi method by using the new values of the entries $x_1, \ldots, x_i$ already in the computation of the next entry x_{i+1}. For this reason, the Jacobi method is known as method of *simultaneous displacements* whereas the (forward or backward) Gauss-Seidel method is known as method of *successive displacements*.

Note that the second pseudocode for the Gauss-Seidel method can be obtained from the second pseudocode for the Jacobi method by reordering the calculations. In fact, a very naive implementation of the Jacobi method may actually implement the Gauss-Seidel method by accident. Following this algorithmic idea has led us to an algebraic idea.

A similar reasoning pertains to the backward Gauss-Seidel method.

3.2. **Composition.** The composition of two linear fixpoint mappings for the system $Ax = b$ yields another such fixpoint mapping. Suppose that we have two fixpoint mappings

$$T_1(x) = x + P_1^{-1}(b - Ax), \quad T_2(x) = x + P_2^{-1}(b - Ax).$$

Then their composition is

$$\begin{aligned} T_2\left(T_1(x)\right) &= x + P_1^{-1}(b - Ax) + P_2^{-1}\left(b - A\left(x + P_1^{-1}(b - Ax)\right)\right) \\ &= x + P_1^{-1}(b - Ax) + P_2^{-1}b - P_2^{-1}A\left(x + P_1^{-1}(b - Ax)\right) \\ &= x + P_1^{-1}(b - Ax) + P_2^{-1}(b - Ax) - P_2^{-1}AP_1^{-1}(b - Ax) \\ &= x + \left(P_1^{-1} + P_2^{-1} - P_2^{-1}AP_1^{-1}\right)(b - Ax). \end{aligned}$$

We note that composition in reverse order gives

$$T_1\left(T_2(x)\right) = x + \left(P_2^{-1} + P_1^{-1} - P_1^{-1}AP_2^{-1}\right)(b - Ax),$$

which is generally a different fixpoint mapping.

Remark 2
Note that the preconditioner P of the composed mapping corroborates our heuristic that the inverse of preconditioner should approximate the inverse of the system matrix A. Indeed, if we have the approximation $P_1^{-1} \approx A^{-1}$ and $P_2^{-1} \approx A^{-1}$, then

$$P^{-1} = P_1^{-1} + P_2^{-1} - P_2^{-1}AP_1^{-1} \approx A^{-1} + A^{-1} - A^{-1}AA^{-1} \approx A^{-1}.$$

Furthermore, the expression for P^{-1} involves only the matrix A and the mappings P_1^{-1} and P_2^{-1} explicitly, so we do not need to know P_1 and P_2 explicitly as long as their inverses are known explicitly.

On a related note that the preconditioner P in the above expression is not given explicitly, only its inverse P^{-1} is given in explicit terms. This suffices for implementing the method.

Example 6
The symmetric Gauss-Seidel method is given by

$$T_{SGS}(x) := T_{GS,r}\left(T_{GS,r}(x)\right).$$

According to the general formula above, or found via direct computation, the preconditioner P_{SGS} of this fixpoint mapping satisfies

$$P_{SGS}^{-1} = (L+D)^{-1} + (D+R)^{-1} - (D+R)^{-1} A (L+D)^{-1}.$$

By $A = L + D + R$, this simplifies to

$$\begin{aligned} P_{SGS}^{-1} &= (L+D)^{-1} + (D+R)^{-1} - (D+R)^{-1}(L+D+R)(L+D)^{-1} \\ &= (L+D)^{-1} - (D+R)^{-1} R (L+D)^{-1} \\ &= (D+R)^{-1}(D+R)(L+D)^{-1} - (D+R)^{-1} R (L+D)^{-1} \\ &= (D+R)^{-1} D (L+D)^{-1}. \end{aligned}$$

In the special case that A is symmetric we have $R = L^T$, and then it follows that

$$\begin{aligned} P_{SGS}^{-1} &= (L+D)^{-1} + (D+R)^{-1} - (D+R)^{-1}(L+D+R)(L+D)^{-1} \\ &= (L+D)^{-1} - (D+R)^{-1} R (L+D)^{-1} \\ &= (D+R)^{-1}(D+R)(L+D)^{-1} - (D+R)^{-1} R (L+D)^{-1} \\ &= \left(D+L^T\right)^{-1} D (L+D)^{-1}. \end{aligned}$$

In particular, the preconditioner (and its inverse) is symmetric again.

Remark 3
The symmetric Gauss-Seidel mapping is typically defined by first applying the forward Gauss-Seidel mapping and then the backward Gauss-Seidel mapping. Reversing the role of the forward and backward Gauss-Seidel mapping results in a different fixpoint mapping, even if the original matrix A was symmetric.

3.3. **Relaxation.** Another common approach of getting new fixpoint methods from old is known as *relaxation*. For any given fixpoint method with iteration mapping $T : \mathbb{R}^n \to \mathbb{R}^n$ and some parameter $\omega \in \mathbb{R}$ we may consider the following sequence. We let $x^{(0)} \in \mathbb{R}^n$ and define

$$x^{(k+\frac{1}{2})} := T(x^{(k)}), \quad x^{(k+1)} := \omega x^{(k+\frac{1}{2})} + (1-\omega)x^{(k)}.$$

We call ω the *relaxation parameter*. Note that above

$$x^{(k+1)} := x^{(k)} + \omega \left(x^{(k+\frac{1}{2})} - x^{(k)} \right).$$

For $\omega = 1$, we get our original method. For $\omega < 1$, we speak of *under-relaxation*, and for $\omega > 1$, we speak of *over-relaxation*.

This relaxated method can again be written as a fixpoint method. Suppose that our initial fixpoint mapping is

$$T(x) = x + P^{-1}(b - Ax).$$

By some algebraic manipulations, we get

$$\begin{aligned} x^{(k+1)} &= \omega x^{(k)} + \omega P^{-1} \left(b - Ax^{(k)} \right) + (1-\omega)x^{(k)} \\ &= x^{(k)} + \omega P^{-1} \left(b - Ax^{(k)} \right). \end{aligned}$$

So we get the relaxated fixpoint mapping simply by scaling the inverse of the preconditioner with the parameter ω. Let $M = A - \omega^{-1}P$, we can also write it as

$$x^{(k+1)} = \omega P^{-1} \left(b - Ax^{(k)} + \omega^{-1} P \right).$$

The fixpoint mapping is

$$\begin{aligned} T_\omega(x) &= x + \omega P^{-1}(b - Ax) \\ &= \omega P^{-1} \left(b - Ax + \omega^{-1} Px \right). \end{aligned}$$

Remark 4
The choice of the relaxation parameter may be interpreted as dampening or acceleration. The original fixpoint method may be interpreted as updating the approximate solution $x^{(k)}$ by adding the correction $P^{-1}(b - Ax^{(k)})$. Depending on the application, it may be wise to either *dampen* this update by scaling it with $\omega < 1$ or to *accelerate* this update by scaling it with $\omega > 1$.

At this point we remark that typically only $\omega > 0$ is used in the literature. If $\omega = 0$, then the relaxated fixpoint mapping is the identity mapping, and whenever $\omega < 0$ is a reasonable choice, then we should change our preconditioner.

Example 7
The Jacobi over-relaxation method, also known as *JOR*, uses the fixpoint mapping

$$\begin{aligned} T_{JOR}(x) &= (1-\omega)x + \omega T_{Jac}(x) \\ &= x + \omega D^{-1}(b - Ax). \end{aligned}$$

A pseudocode for this relaxation method is

FOR $i = 1, \ldots, n$ DO

$y_i = b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j$
END FOR
FOR $i = 1, \ldots, n$ DO
$x_i = (1-\omega) + \omega a_{ii}^{-1} y_i$
END FOR

Example 8
The relaxated Gauss-Seidel method, which does not really have a catchy name in the literature, uses the fixpoint mapping

$$\begin{aligned} T_{relGS,l}(x) &= (1-\omega)x + \omega T_{GS,l}(x) \\ &= x + \omega D^{-1}\left(b - Ax\right). \end{aligned}$$

A pseudocode for this relaxation method is

FOR $i = 1, \ldots, n$ DO
$y_i = b_i - \sum_{j=1}^{i-1} a_{ij}y_j - \sum_{j=i+1}^{n} a_{ij}x_j$
END FOR
FOR $i = 1, \ldots, n$ DO
$x_i = (1-\omega) + \omega a_{ii}^{-1} y_i$
END FOR

More precisely, this is the relaxation of the forward Gauss-Seidel method. The relaxated Gauss-Seidel method is derived analogously.

Example 9
We conclude with a rather trivial example of a relaxated method. Applying the relaxation construction to the Richardson iteration leads to the fixpoint mapping

$$T(x) = x + \omega\theta^{-1}\left(b - Ax\right).$$

This relaxated method yields nothing new since it has the form as the original Richardson method, only the parameter θ is changed to $\omega^{-1}\theta$. In other words, relaxation does not introduce anything substantially new in conjunction with the Richardson method.

One may think of the Richardson method as a relaxation of the fixpoint method $x \mapsto x + (b - Ax)$. The Richardson method is often simply called *relaxation method* in the literature.

3.4. **SOR Method.** The relaxation of the Gauss-Seidel method eventually inspires a fixpoint mapping that is known as successive over-relaxation in the literature. Similarly as the Gauss-Seidel method is inspired from the algorithm for the Jacobi method, we can interpret the SOR method as inspired from the algorithm of the Gauss-Seidel method.

Recall the relaxated (forward) Gauss-Seidel fixpoint mapping,

$$T_{relGS,l}(x) = x^{(k)} + \omega(L+D)^{-1}\left(b - Ax^{(k)}\right).$$

Using a pseudocode, this can be written in terms of two FOR loops using

FOR $i = 1, \ldots, n$ DO

$y_i = a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}y_j - \sum_{j=i+1}^{n} a_{ij}x_j\right)$
END FOR
FOR $i = 1, \ldots, n$ DO
$x_i = (1-\omega)x_i + \omega y_i$
END FOR

What about removing the auxiliary vector y and replacing the old entries of x by the new entries as soon as they are available? This leads to the pseudocode

FOR $i = 1, \ldots, n$ DO
$x_i = (1-\omega)x_i + \omega a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right)$
END FOR

In terms of components, we calculate $x^{(k+1)}$ from $x^{(k)}$ by

$$\begin{aligned} x_i^{(k+1)} &= (1-\omega)x_i^{(k)} + \omega a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \\ &= (1-\omega)x_i^{(k)} + \omega a_{ii}^{-1}b_i - \sum_{j=1}^{i-1} \omega a_{ii}^{-1}a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} \omega a_{ii}^{-1}a_{ij}x_j^{(k)}. \end{aligned}$$

This is equivalent to applying forward substitution to the triangular system

$$\left(I + \omega D^{-1}L\right)x^{(k+1)} = \omega D^{-1}b + (I-\omega)x^{(k)} - \omega D^{-1}Rx^{(k)}.$$

Reasonably assuming $\omega \neq 0$, we get

$$\left(\omega^{-1}D + L\right)x^{(k+1)} = b + (\omega^{-1}D - D)x^{(k)} - Rx^{(k)}.$$

We recognize this as fixpoint iteration iteration with preconditioner

$$P = \omega^{-1}D + L.$$

The matrix splitting works with

$$\begin{aligned} M = A - P &= A - \omega^{-1}D - L \\ &= R + (1-\omega^{-1})D. \end{aligned}$$

This method is known as *successive over-relaxation*, abbreviated *SOR*.

Remark 5
Despite the name, the relaxation parameter may not be classified as over-relaxation. It is entirely possible that $\omega < 1$. In particular, the (forward) Gauss-Seidel mapping is obtained in the special case $\omega = 1$.

The SOR method above is derived from the forward Gauss-Seidel method. Of course, a completely analogous construction is feasible from the backward Gauss-Seidel method too. There is no general terminology in the literature to distinguish between those two methods, but one may speak of forward SOR method and backward SOR method.

The forward and backward SOR methods are then given by

$$T_{SOR,l}(x) := \left(\omega^{-1}D + L\right)^{-1}(b - Rx - (1-\omega^{-1})Dx),$$
$$T_{SOR,r}(x) := \left(\omega^{-1}D + R\right)^{-1}(b - Lx - (1-\omega^{-1})Dx).$$

3.5. **Symmetric SOR Method.** The preconditioner of the Jacobi method is symmetric, which can be helpful as a building block in other iterative methods. For example, the eigenvalues of the Jacobi preconditioner are all real. The Gauss-Seidel and SOR method generally converge faster than the Jacobi method, but their preconditioners are generally non-symmetric. As a remedy, we can symmetrize the SOR method.

The symmetric successive over-relaxation method, also known as SSOR method, is defined as the composition

$$T_{SSOR,\omega}(x) := T_{SOR,r}\left(T_{SOR,l}(x)\right).$$

If A is symmetric, then one can show that

$$T_{SSOR}(x) = \frac{\omega}{2-\omega}\left(\frac{D}{\omega} + L\right)D^{-1}\left(\frac{D}{\omega} + L^T\right).$$

Remark 6
We notice that the symmetric Gauss-Seidel is obtained as a special case of the symmetric SOR method when setting $\omega = 1$. However, the symmetric SOR method is *not* the relaxation of the symmetric Gauss-Seidel method. For example, this is already evident for $A = D$ and most values of ω.

4. Concluding Remarks

We have not discussed the convergence properties of any fixpoint method except for Richardson iteration. As a rule of thumb, since most of the preconditioners rely on the diagonal to be invertible, we can reasonably expect their convergence properties to depend on properties and quantities associated with the diagonal part. As an example, we present the following two results without proof.

Theorem 3
Let $A \in \mathbb{R}^{n\times n}$ be strictly diagonally dominant. Then the iteration matrices of the Jacobi and the Gauss-Seidel iterations satisfy

$$\|\operatorname{Id} - P_{GS,l}^{-1}A\|_\infty \leq \|\operatorname{Id} - P_{Jac}^{-1}A\|_\infty < 1.$$

Theorem 4 (Ostrowski, Reich)
Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Then the symmetric successive over-relaxation method converges if and only if $\omega \in (0, 2)$.

The first result states the Jacobi method and the Gauss-Seidel are contractions with respect to the ∞-norm in the case of a strictly diagonal dominant matrix, with the Gauss-Seidel method being a better contraction than the Jacobi method. The second result states that the SSOR method converges precisely for the parameter

range $0 < \omega < 2$. Finding the optimal relaxation parameter to minimize the spectral radius is a delicate task.

Purely algorithmic concerns are of interest too in the context of these methods. One example is parallelism. The Richardson iteration and the Jacobi iteration are intrinsically parallel: in any of the pseudocodes for these methods given above, the steps within each FOR loop can be executed independently of each other and in any order. The same holds for the JOR method. Consequently, the steps in those FOR loops can be distributed among different processors.

By contrast, the Gauss-Seidel method and its variants, including the SOR method, are sequential by nature. The order of the execution of the steps in the FOR loops does matter and cannot be meaningfully parallelized in general. Sometimes a remedy is to reorder the coordinates the solution vector to give the matrix a special structure than is easy to exploit, but that is complicated optimization that is feasible only under special circumstances.

Hence, while the *successive* methods may have better convergence properties, it is easier to exploit parallelism with the more primitive *simultaneous* methods.