Chapter 6
Iterative Methods

In the previous chapter we considered methods which directly solve (SLE) using $\Theta(n^3)$ operations. In this chapter we will introduce iterative methods. These methods construct a sequence $(x_k)_{k \in \mathbb{N}}$ with

$$x_k \to x \quad \text{for} \quad k \to \infty,$$

where $x$ solves (SLE). These methods approximate the exact solution $x$ by the computed value $x_k$ for large $k$ and thus, even in the absence of rounding errors, the resulting algorithms only compute approximate solutions to (SLE). In practical applications this is no problem, since one can choose $k$ large enough such that the approximation error is of the same order of magnitude as the deviation caused by rounding errors in the “exact” methods.

The primary motivation for the use of such iterative methods arises for problems where $A \in \mathbb{C}^{n \times n}$ with $n \gg 1$, so that solving the problem by the direct methods of Chapter 5, with cost $\Theta(n^3)$, is prohibitively expensive. If $A$ has some special structure, for example if matrix-vector multiplication with $A$ is cheap to calculate, then iterative methods can have a substantial advantage over direct methods.

For the methods considered in this chapter, the sequence $(x_k)$ is constructed iteratively, i.e. starting with a value $x_0$ the value $x_k$, for each $k$, is computed from $x_{k-1}$. The map $x_{k-1} \mapsto x_k$ may be either linear or nonlinear; we discuss methods of both types. The two competing criteria for the iteration are:

- The calculation of $x_k$ from $x_{k-1}$ should be cheap, relative to the cost of solving (SLE).
- The convergence of $x_k$ to $x$ should be fast.

These two criteria are often contradictory to one another and the trade-off between them is very much dependent upon the matrix $A$ itself. To illustrate this trade off it is useful to have the following two extreme iterative methods in mind:

- $x_k = A^{-1}b$; and
- $x_k = x_{k-1}$.

The first is not cheap to implement — each step involves solving (SLE) by a direct method, and our assumption is that this is prohibitively expensive — but converges fast: in one step. The second is cheap to implement but does not converge at all. Successful iterative methods lie somewhere between these extremes.

In many cases, the cost of computing $x_k$ from $x_{k-1}$ is dominated by the cost of matrix-vector multiplication for a matrix $N$ which is derived from $A$. The resulting methods will be efficient if these multiplications can be computed efficiently. The most common structure leading to cheap matrix-vector multiplication is sparsity: where $A$ has only a small number of non-zero entries. Then matrix-vector multiplication by $A$, $A^*$ or any matrix $N$ where $N$ inherits the sparsity of $A$ will be cheap; the matrix $N$ might, for example, be formed from the off-diagonals of $A$.

In this chapter we will focus only on errors incurred through using a finite number of iterations but we will not discuss the effect of rounding errors. For each section describing a computational
approach there will be three subsections, one defining the method, the second studying error as a function of the number of iterations \( k \), and the third discussing computational cost. We will consider the cost of achieving an error of size \( \varepsilon \). We will show that, for some important problems, it is possible to compute an approximate solution in time \( \Theta(n^a) \) with \( a < 3 \) as the dimension \( n \) of the problem increases, thus beating the cost of the direct methods from the previous chapter.

6.1 Linear Methods

Iterative methods where \( x_k \) is computed from \( x_{k-1} \) by application of a linear map are called linear methods. In this section we describe results about linear methods in general. Sections 6.2 and 6.3 below introduce two specific instances of linear methods, namely the Jacobi method and the successive over relaxation scheme (SOR).

The Method

The basic idea of the methods described in this section is to write the matrix \( A \) as \( A = M + N \) where the matrix \( M \) is easier to invert than \( A \). Then, given \( x_{k-1} \), we can define \( x_k \) by

\[
Mx_k = b - Nx_{k-1}. \tag{6.1}
\]

If we assume for the moment that \( \lim_{k \to \infty} x_k = x \), then the limit \( x \) satisfies

\[
Mx = \lim_{k \to \infty} Mx_k = \lim_{k \to \infty} (b - N x_{k-1}) = b - Nx
\]

and thus we get \( Ax = b \). This shows that the only possible limit for this sequence is the solution of (SLE). Thus, once we know that the generated sequence converges for a given matrix \( A \) and a given initial value \( x_0 \), we can iteratively compute \( x_k \) for a "big" value of \( k \) to get an approximation for \( x \). In the following sections we will give sufficient criteria for convergence of the sequence \( (x_k)_{k \in \mathbb{N}} \).

This leads to the following algorithm.

Algorithm LI (linear iterative methods).

input: \( A = M + N \in \mathbb{C}^{n \times n} \), \( b \in \mathbb{C}^n \), \( x_0 \in \mathbb{C}^n \)
output: \( x_k \in \mathbb{C}^n \) with \( Ax_k \approx b \)

1: for \( k = 1, 2, 3, \ldots \) do
2: compute \( y_k = b - N x_{k-1} \)
3: solve \( Mx_k = y_k \)
4: end for

To actually implement this method one needs to choose \( M \) and \( N \) such that the method converges (we discuss different choices below). One also needs a stopping criterion to decide when to quit iterating and to return the resulting approximation \( x_k \).

Thus, the remaining problem is to choose a stopping criterion for the iteration. We consider the error

\[
e_k = x - x_k,
\]

and the residual vector

\[
r_k = b - Ax_k = Ae_k
\]

where \( x \) is the exact solution of (SLE) and \( x_k \) the approximation at step \( k \). We would like to choose \( k \) big enough such that \( \| e_k \| \leq \varepsilon \) for a given \( \varepsilon > 0 \) but, since \( e_k \) cannot be computed without knowledge of the exact solution \( x \), this is not a practical stopping criterion. Instead, we will stop the iteration once \( \| r_k \| \leq \varepsilon_r \) for some \( \varepsilon_r > 0 \). The residual error \( r_k \) can easily be computed during the iteration and from the estimate \( \| e_k \| = \| A^{-1} r_k \| \leq \| A^{-1} \| \| r_k \| \leq \| A^{-1} \| \varepsilon_r \)
we can get estimates for the error if needed.
Error Analysis

Since $Mx = b - Nx$ we get the relation

$$e_k = -M^{-1}Ne_{k-1}.$$  

The method converges if $e_k \to 0$ for $k \to \infty$. The following lemma characterises convergence with the help of the spectral radius of the matrix $R = -M^{-1}N$.

**Lemma 6.1.** Let $R \in \mathbb{C}^{n \times n}$, $e_0 \in \mathbb{C}^n$ and $e_k = R^k e_0$ for all $k \in \mathbb{N}$. Then $e_k \to 0$ for all $e_0 \in \mathbb{C}^n$ if and only if $\rho(R) < 1$.

**Proof.** Assume first $\rho(R) < 1$. Then by Lemma 2.10 we find an induced matrix norm $\| \cdot \|_S$ with $\| R \|_S < 1$ and we get

$$\| e_k \|_S = \| R^k e_0 \|_S \leq \| R \|_S^k \| e_0 \|_S \to 0$$

for $k \to \infty$.

On the other hand, if $\rho(R) \geq 1$, then there is an $e_0 \in \mathbb{C}^n \setminus \{0\}$ with $Re_0 = \lambda e_0$ for some $\lambda \in \mathbb{C}$ with $|\lambda| \geq 1$. For this vector $e_0$ we get, in any norm $\| \cdot \|$

$$\| e_k \| = \| R^k e_0 \| = |\lambda|^k \| e_0 \|$$

and thus $e_k$ does not converge to 0 as $k \to \infty.$

**Remarks.**

1. The lemma shows that the linear iterative method defined by (6.1) converges for every initial condition $x_0 \in \mathbb{C}^n$ if and only if $\rho(R) < 1$ for $R = -M^{-1}N$. The convergence is fast if $\rho(R)$ is small.

2. Since $\| \cdot \| \geq \rho(\cdot)$ for every matrix norm $\| \cdot \|$, a sufficient criterion for convergence of an iterative method is $\| R \| < 1$ for any matrix norm $\| \cdot \|$. On the other hand, whenever the method converges for every initial condition, there is an induced matrix norm $\| \cdot \|$ with $\| R \| < 1$ by Lemma 2.10.

**Computational Complexity**

When measuring computational cost for iterative methods it is of interest to know how many iterations are required to reduce the error to size $\varepsilon$. As before we will consider the relative error $\| x_k - x \| / \| x \| = \| e_k \| / \| x \|$. By Proposition 3.3 we have

$$\frac{\| e_k \|}{\| x \|} \leq \kappa(A) \frac{\| r_k \|}{\| b \|}. \quad (6.2)$$

Finding $k$ to achieve

$$\frac{\| r_k \|}{\| b \|} \leq \varepsilon, \quad (6.3)$$

and thus

$$\frac{\| e_k \|}{\| x \|} \leq \kappa(A) \varepsilon, \quad (6.4)$$

will lead to an estimate of the computational cost of using the iterative method.

Using the relation $r_k = Ae_k = AR^k e_0$ we can estimate the left hand side of (6.3) to get, from (6.2),

$$\frac{\| e_k \|}{\| x \|} \leq \kappa(A) \frac{\| A \| \| R \|^k \| e_0 \|}{\| b \|}$$

for any vector norm $\| \cdot \|$. If $\| \cdot \|$ is chosen so that $\| R \| < 1$ then choosing $k$ to be the smallest integer greater than

$$k^* = \frac{\ln \varepsilon^{-1} + \ln \| A \| + \ln \| e_0 \| - \ln \| b \|}{\ln \| R \|^{-1}} \quad (6.5)$$
will ensure (6.3).

In order to study the dependence of the cost on the dimension \( n \) of the problem we consider a family of matrices \( A = M + N \in \mathbb{C}^{n \times n} \) and vectors \( b \in \mathbb{C}^n \) and also a family of vector norms on the spaces \( \mathbb{C}^n \). The number \( k^\# \) depends on both \( \epsilon \) and \( n \); the \( n \) dependence can enter through \( \| R \|^{-1} \), \( \| A \| \), \( \| b \| \), and \( \| e_0 \| \), but the largest contribution is often through \( \| R \|^{-1} \), for families of problems in which \( \| R \| = 1 - \Theta(n^{-\beta}) \). We incorporate this into the following assumptions, which lead to a quantification of computational cost.

**Assumption 6.2.**

1. Calculation of \( Nx \) and \( M^{-1}x \) together costs \( \Theta(n^\alpha) \) for some \( \alpha > 0 \), uniformly for all \( x \in \mathbb{C}^n \).
2. \( \| R \| = 1 - \Theta(n^{-\beta}) \) for some \( \beta > 0 \).
3. \( \kappa(A), \| A \|, \| b \| \) and \( \| e_0 \| \) are bounded uniformly in \( n \).

**Theorem 6.3.** Under Assumption 6.2 the computational cost to achieve (6.4) by using the linear iterative method is bounded above by \( cn^{\alpha+\beta} \ln \epsilon^{-1} \), for some constant \( c \) independent of \( n \) and \( \epsilon \).

**Proof.** The first item of the assumption ensures that each step of the iteration costs \( \Theta(n^\alpha) \). The second item of the assumption implies that \( (\ln \| R \|^{-1})^{-1} = \Theta(n^\beta) \). Hence combining the second and third items and using (6.5) gives the desired result.

**Remarks.**

1. For the theorem to be practically useful, the norm in (6.3) must be readily computable. In some cases \( \| R \| < 1 \) in some computable norm such as \( \| \cdot \|_\infty, \| \cdot \|_2 \) or \( \| \cdot \|_1 \). Then the preceding theorem applies directly.

In other cases we only know that \( \| R \|_S < 1 \); as the construction of \( \| \cdot \|_S \) requires knowledge of the Jordan Canonical form, this is not a readily computable norm. However, by norm equivalence, we know that there is a constant \( c_2 \in (0, \infty) \) such that

\[
\frac{\| a \|}{\| b \|} \leq c_2 \frac{\| a \|_S}{\| b \|_S}.
\]

The constant \( c_2 \) may depend on \( n \), but in many important applications this occurs only in a polynomial fashion.

Iterating until we find a \( k \) so that

\[
\frac{\| r_k \|_S}{\| b \|_S} \leq \frac{\epsilon}{c_2}
\]

will ensure

\[
\frac{\| r_k \|}{\| b \|} \leq \epsilon
\]

and hence (6.4). Finding \( k \) to achieve (6.6) will incur cost

\[
k n^{\alpha+\beta} \left( \ln \epsilon^{-1} + \ln c_2(n) \right)
\]

by the methods given in the preceding theorem, if Assumption 6.2 holds in the norm \( \| \cdot \|_S \). Thus there is an increase in cost by only a logarithmic factor in \( n \) when \( c_2(n) \) is polynomial in \( n \).

2. In many applications \( \kappa(A), \| A \| \) and \( \| b \| \) do depend upon \( n \), but only polynomially. Again this leads to an increase in computational cost over the above, but only by a factor which is logarithmic in \( n \).
6.2 The Jacobi Method

In this and the next section we will consider specific methods which are obtained by choosing the matrices $M$ and $N$ in (6.1). For a matrix $A \in \mathbb{C}^{n \times n}$ define the three $n \times n$-matrices

$$L = \begin{pmatrix} a_{21} & a_{32} & \cdots & a_{n, n-1} \\ a_{31} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad D = \begin{pmatrix} a_{11} & a_{22} & \cdots & \cdots \\ a_{22} & a_{33} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{nn} \end{pmatrix},$$

$$U = \begin{pmatrix} a_{12} & a_{13} & \cdots & a_{1n} \\ a_{23} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1n} & \cdots & \ddots & a_{nn} \end{pmatrix}.$$  (6.7)

Then we have $A = L + D + U$. The matrices $L$ and $U$ are, respectively, strictly lower and upper triangular.

**The Method**

The iterative method obtained by choosing $M = D$ and $N = L + U$ in (6.1) is called Jacobi method. This choice of $M$ and $N$ leads to the iteration

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

for all $k \in \mathbb{N}$. Since $D$ is diagonal, the inverse $D^{-1}$ is trivial to compute.

**Error Analysis**

In order to study convergence of the Jacobi method using Lemma 6.1 we have to consider the matrix $R = -M^{-1}N = -D^{-1}(L+U)$. The method converges if and only if $\rho(R) < 1$. The following theorems give sufficient criteria for this to happen.

**Theorem 6.4.**

a) The Jacobi method is convergent for all matrices $A$ with

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

for $i = 1, \ldots, n$. (This condition is called the strong row sum criterion.)

b) The Jacobi method is convergent for all matrices $A$ with

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

for $j = 1, \ldots, n$. (This condition is called the strong column sum criterion.)

**Proof.**

a) The matrix $R = -D^{-1}(L+U)$ has entries

$$r_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}}, & \text{if } i \neq j, \text{ and} \\ 0, & \text{else.} \end{cases}$$

Using Theorem 1.28 we find

$$\|R\|_{\infty} = \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} \sum_{j \neq i} |a_{ij}|.$$ 

Thus the strong row sum criterion gives $\|R\|_{\infty} < 1$ which implies $\rho(R) < 1$ and the method converges.
b) If the strong column sum criterion (6.9) holds for $A$, then the strong row sum criterion (6.8) holds for $A^*$ and thus the method converges for $A^*$. From Lemma 6.1 we know then
\[ \rho(-(D^*)^{-1}(L^* + U^*)) < 1. \]
Since for every matrix $R$ the matrices $R, R^*$ and also $D^{-1}RD$ have the same eigenvalues we get
\[
\rho(-D^{-1}(L + U)) = \rho(-(L + U)D^{-1}) \\
= \rho(-(L + U))D^{-1} \\
= \rho(-(D^*)^{-1}(L^* + U^*)) < 1
\]
and so the method converges for $A$.

Remark. As well as showing convergence, the proof of the theorem estimates $\|R\|_\infty < 1$. This estimate can be used in Theorem 6.3 to analyse computational complexity.

Definition 6.5. A matrix $A \in \mathbb{C}^{n \times n}$ is called irreducible if there is no permutation matrix $P$ such that
\[
P^TAP = \begin{pmatrix}
\tilde{A}_{11} & \tilde{A}_{12} \\
0 & \tilde{A}_{22}
\end{pmatrix}
\]
where $\tilde{A}_{11} \in \mathbb{C}^{p \times p}$ and $\tilde{A}_{22} \in \mathbb{C}^{q \times q}$ are square matrices with $p, q > 0$ and $p + q = n$, $\tilde{A}_{12} \in \mathbb{C}^{p \times q}$, and $0$ is the $q \times p$ zero-matrix.

There is an alternative description of irreducibility, which is often easier to check than the definition given. To the matrix $A$ we associate the oriented graph $G(A)$ with vertices $1, \ldots, n$ and edges $i \rightarrow j$ for all $i, j \in \{1, \ldots, n\}$ with $a_{ij} \neq 0$. Then the matrix $A$ is irreducible if and only if the graph $G(A)$ is connected, i.e. if you can reach any vertex $j$ from any vertex $i$ by following edges.

Example. Consider the matrix
\[
A = \begin{pmatrix}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1
\end{pmatrix}.
\]
The associated graph is

![Graph 1](image)

The matrix $A$ is not irreducible (indeed $P = I$ in the definition is enough to see this) and since there is no path from 3 to 1, the graph $G(A)$ is not connected.

Example. In continuation of the previous example consider the modified matrix
\[
A = \begin{pmatrix}
1 & -1 & -1 \\
-1 & 2 & -1 \\
-1 & -1 & 1
\end{pmatrix}.
\]
The associated graph is

![Graph 2](image)

Now the graph $G(A)$ is connected and the matrix is thus irreducible.
Remarks.

The proof of equivalence of the two characterisations of irreducibility is based on the following observations.

1. For any permutation $P$ the graphs $G(A)$ and $G(PTAP)$ are isomorphic, only the vertices are numbered in a different way.

2. The block $\tilde{A}_{22}$ in the definition of irreducibility corresponds to a set of states from where there is no path into the states corresponding to the block $\tilde{A}_{11}$.

The characterisation of irreducibility through $G(A)$ shows that the diagonal entries of $A$ are irrelevant to determining irreducibility. Any matrix which has non-zero off-diagonal entries if and only if $A$ does so, will have the same graph. Hence $R = -D^{-1}(L + U)$ is irreducible if and only if $A$ is.

**Theorem 6.6.** If $A$ is irreducible and satisfies

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

(6.10)

for $i = 1, \ldots, n$ but

$$|a_{kk}| > \sum_{j \neq k} |a_{kj}|$$

(6.11)

for one index $k$, then the Jacobi method converges. (This condition is called the weak row sum criterion.)

**Proof.** We have to show that $\rho(R) < 1$ where $R = -D^{-1}(L + U)$. Define the matrix $|R| = (|r_{ij}|)_{ij} \in \mathbb{R}^{n \times n}$ and the vector $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$. Then we have

$$\langle |R| e \rangle_i = \left( \sum_{j=1}^n |r_{ij}| \cdot 1 \right)_i = \left( \sum_{j \neq i} \frac{|a_{ij}|}{a_{ii}} \right)_i \leq 1 = e_i.$$

Thus $|R| e \leq e$ where this and some of the following inequalities between vectors are to be read componentwise. Therefore we get

$$|R|^{l+1} e \leq |R|^{l} e \leq \cdots \leq e$$

for all $l \in \mathbb{N}$.

Let $t^{(l)} = e - |R|^{l} e \geq 0$. Then the vectors $t^{(l)}$ are componentwise increasing. Let $\tau_l$ be the number of non-vanishing components of $t^{(l)}$. We will show that $\tau_l$ is strictly increasing until it reaches the value $n$. Assume, for contradiction, that $\tau_{l+1} = \tau_l = k < n$. Since one row of $A$ satisfies the strict inequality (6.11) we have $|R| e \neq e$ and thus $k > 0$. Then without loss of generality (since we can reorder the rows and columns of $A$) we have

$$t^{(l)} = \begin{pmatrix} a \\ 0 \end{pmatrix}, \quad t^{(l+1)} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

where $a, b \in \mathbb{R}^k$ and $a, b > 0$. We can conclude

$$\begin{pmatrix} b \\ 0 \end{pmatrix} = t^{(l+1)} = e - |R|^{l+1} e$$

$$\geq |R| e - |R|^{l+1} e = |R| t^{(l)}$$

$$= \begin{pmatrix} |R_{11}| & |R_{12}| \\ |R_{21}| & |R_{22}| \end{pmatrix} \begin{pmatrix} a \\ 0 \end{pmatrix}.$$

From $a > 0$ we have $|R_{21}| = 0$. Hence $R$ is not irreducible. This implies that $A$ is not irreducible, and we have found the required contradiction. Thus we can conclude that $\tau_{l+1} > \tau_l$ whenever $\tau_l < n$. This proves $t^{(n)} > 0$ componentwise.

Hence $e \geq |R|^{n} e$ and we get, using Lemma 1.34,

$$\rho(R)^n \leq \rho(R^n) \leq \|R^n\|_\infty = \|R^n\|_\infty \leq \|R\|_\infty = \max_{i=1,\ldots,n} (|R|^{n} e)_i < 1$$

and thus $\rho(R) < 1$. This completes the proof. \qed
Computational Complexity

The sequence \((x_k)_{k \in \mathbb{N}}\) from the Jacobi method is defined by the relation
\[
x_k = -D^{-1}(b - (L + U)x_{k-1}).
\]

For each iteration we have to calculate
1) the product \((L + U)x_{k-1}\)
2) the difference \(b - \cdots\) (needs \(n\) subtractions)
3) the product \(D^{-1}\cdots\) (needs \(n\) divisions because \(D\) is diagonal)

For general matrices the first step needs \(\Theta(n^2)\) operations and thus dominates the computational cost. If the matrix \(A\) is sparse, i.e., if it contains many zeros, the matrix-vector multiplication can be performed using fewer operations.

**Example.** Assume that each row of \(A\) contains at most \(\ell > 0\) non-zero entries outside the diagonal. Then step 1) requires \(\Theta(n)\) operations and performing one step of Jacobi method has cost \(\Theta(n)\) as \(n \to \infty\). In this case we get \(\alpha = 1\) in Theorem 6.3.

### 6.3 The Gauss-Seidel and SOR Methods

**The Method**

The method obtained by using \(M = L + \omega D\) and \(N = U + (1 - \omega)D\) in (6.1), for the matrices \(L, D\) and \(U\) from (6.7) and some \(\omega \in \mathbb{R}\), is called the *Successive Over Relaxation* (SOR) scheme. The parameter \(\omega\) can be chosen to accelerate convergence. The special case \(\omega = 1\), i.e. \(M = L + D\) and \(N = U\), is called the *Gauss-Seidel method*. For the SOR scheme we get the iteration
\[
(L + \omega D)x_k = b - Ux_{k-1} - (1 - \omega)Dx_{k-1}.
\]

Since \((L + \omega D)\) is lower triangular we can use forward substitution to calculate \(x_k\) in each step.

**Error Analysis**

The error analysis of the SOR and Gauss-Seidel method is similar to the analysis for the Jacobi method from the previous section. The convergence properties of the SOR scheme are determined by the matrix
\[
R = -(L + \omega D)^{-1}(U + (1 - \omega)D).
\]

The following summarises some sufficient criteria for convergence of the Gauss-Seidel method.

**Theorem 6.7.** Assume either
a) \(A\) satisfies the strong row sum criterion or
b) \(A\) is irreducible and satisfies the weak row sum criterion.
Then the Gauss-Seidel (\(\omega = 1\)) method converges.

**Computational Complexity**

The speed of convergence and thus the cost of the method depends on the choice of the parameter \(\omega\). For linear systems arising from discretisation of certain elliptic PDEs, optimising can lead to an order of magnitude improvement in efficiency, when compared with Gauss-Seidel and Jacobi methods. This arises though decreasing \(\beta\) in Theorem 6.3 for appropriate choice of \(\omega\). In the same context Gauss-Seidel improves over Jacobi, but not by an order of magnitude.
6.4 Nonlinear Methods

In the remaining part of this chapter we will consider nonlinear methods for solving (SLE), i.e. methods where the map from \(x_k\) to \(x_{k+1}\) is no longer linear. We restrict ourselves to the case where the matrix \(A\) is Hermitian and positive definite. This section explains the general approach. The remaining two sections of the chapter examines two specific instances of nonlinear methods in detail.

The fundamental observation for the methods we consider here is that the vector equation \(Ax = b\) is equivalent to \(\|Ax - b\| = 0\) for any vector norm \(\|\cdot\|\). Since we assume that \(A\) is Hermitian and positive definite, \((x, y)_A = \langle x, Ay \rangle\) defines an inner product and \(\|x\|^2_A = \langle x, Ax \rangle\) defines the associated vector norm. The matrix \(A^{-1}\) is again Hermitian and positive definite and \(\|x\|^2_{A^{-1}} = \langle x, A^{-1}x \rangle\) defines the associated vector norm.

Lemma 6.8. Let \(A \in \mathbb{C}^{n \times n}\) be Hermitian and positive definite and \(b \in \mathbb{C}^n\). Then \(x \in \mathbb{C}^n\) solves \(Ax = b\) if and only if \(x\) is the minimiser of the function

\[
g(y) = \frac{1}{2} \|Ay - b\|^2_{A^{-1}} \quad \forall y \in \mathbb{C}^n.
\]

In this section we will consider iterative methods which solve (SLE) by constructing sequences \((x_k)_{k \in \mathbb{N}}\) which converge to the unique minimum of the function \(g\). Using the residual \(r_k = b - Ax_k\) and the error \(e_k = x - x_k\), where \(x\) is the exact solution of \(Ax = b\), we can write

\[
g(x_k) = \frac{1}{2} \|r_k\|^2_{A^{-1}} = \frac{1}{2} \|e_k\|^2_A.
\]

Thus minimising \(g\) corresponds to minimising the length of the residual vector in the \(\|\cdot\|_{A^{-1}}\)-norm or, equivalently, to minimising the error for the exact solution in the \(\|\cdot\|_A\)-norm.

The Method

The two methods that we study below both take the form

\[
x_k = x_{k-1} + \alpha_{k-1}d_{k-1},
\]

where \(d_{k-1} \in \mathbb{C}^n \setminus \{0\}\) is called the search direction and the scalar \(\alpha_{k-1} \in \mathbb{C}\) is called the step length. The step length is chosen so that, given \(x_{k-1}\) and \(d_{k-1}\),

\[
\alpha_{k-1} = \arg\min_{\alpha \in \mathbb{C}} g(x_{k-1} + \alpha d_{k-1}).
\]

Since \(g\) is a convex, quadratic function, the value \(\alpha_{k-1}\) is uniquely determined.

The minimum can be found using the fact the \(g\) satisfies

\[
g(y + \varepsilon z) = g(y) + \varepsilon \text{Re}(z, Ay - b) + \frac{\varepsilon^2}{2} \|z\|^2_A
\]

for all \(\varepsilon \in \mathbb{R}\) and \(z, y \in \mathbb{C}^n\), where \(\text{Re}\) denotes the real part of a complex number. Taking the derivative w.r.t. \(\varepsilon\) (along the real line) gives

\[
\frac{\partial}{\partial \varepsilon} g(y + \varepsilon z)|_{\varepsilon = 0} = \text{Re}(z, Ay - b).
\]

At the minimum we find

\[
0 = \frac{\partial}{\partial \varepsilon} g(x_{k-1} + \alpha_{k-1}d_{k-1} + \varepsilon \beta d_{k-1})|_{\varepsilon = 0}
= \text{Re} \beta (d_{k-1}, A(x_{k-1} + \alpha_{k-1}d_{k-1}) - b)
= \text{Re} \beta (d_{k-1}, r_{k-1} + \alpha_{k-1}d_{k-1})
\]

for all \(\beta \in \mathbb{C}\) and thus

\[
\alpha_{k-1} = \frac{\langle d_{k-1}, r_{k-1} \rangle}{\|d_{k-1}\|^2_A}.
\]

Hence we get the following algorithm.
Algorithm NI (nonlinear iterative methods).

**input:** \( A \in \mathbb{C}^{n \times n} \) Hermitian and positive definite, \( b \in \mathbb{C}^n \), \( x_0 \in \mathbb{C}^n \)

**output:** \( x_k \in \mathbb{C}^n \) with \( Ax_k \approx b \)

1. for \( k = 1, 2, 3, \ldots \) do
2. compute \( d_{k-1} \)
3. \( \alpha_{k-1} = \frac{\langle d_{k-1}, r_{k-1} \rangle}{\|d_{k-1}\|_A^2} \)
4. \( x_k = x_{k-1} + \alpha_{k-1} d_{k-1} \)
5. end for

The steepest descent method and the conjugate gradient method, discussed below, are instances of this algorithm, employing different choices for the search directions \( d_k \).

**Error Analysis**

The error analysis for nonlinear methods is done in two steps. We first study how fast the value \( \|e_k\|_A^2 = 2g(x_k) \) decays. This analysis depends on the specific choice of search directions \( d_k \) and is presented for the methods below, separately. In a second step we can use this result to estimate the error of the approximate solution \( x_k \) in the Euclidean norm:

**Lemma 6.9.** Assume \( \|e_k\|_A \leq cq^k\|e_0\|_A \) for all \( k \in \mathbb{N} \) for some constants \( q, c > 0 \). Then

\[
\|e_k\| \leq \sqrt{\kappa(A)c}q^k\|e_0\| \quad \forall k \in \mathbb{N}
\]

where \( \kappa(A) = \|A\|_2\|A^{-1}\|_2 \) is the condition number of \( A \) in the 2-norm.

**Proof.** Let \( \lambda_{\min} \) and \( \lambda_{\max} \) be the minimal and maximal eigenvalue of \( A \). Then we have \( \kappa(A) = \lambda_{\max}/\lambda_{\min} \) by (3.2) and from Lemma 1.21 we find

\[
\|e_k\|^2 \leq \frac{1}{\lambda_{\min}}\|e_k\|_A^2 \leq 2\lambda_{\min}q^k\|e_0\|_A^2 \leq \frac{\lambda_{\max}}{\lambda_{\min}}c^k\|e_0\|^2.
\]

This completes the proof. \( \square \)

For future reference we state the following relation, which describes how the residual error changes in each step:

\[
r_k = r_{k-1} - \alpha_{k-1}Ad_{k-1} \quad \forall k \in \mathbb{N}.
\]

This is a direct consequence of (6.13).

6.5 The Steepest Descent Method

**The Method**

The steepest descent method chooses a search direction which makes a step in the direction of the (locally) steepest descent for \( g \). From equation (6.14) we get

\[
g(x_k + \varepsilon d_k) = g(x_k) - \varepsilon \text{Re}\langle d_k, r_k \rangle + \frac{\varepsilon^2}{2}\|d_k\|^2_A
\]

for all \( \varepsilon \in \mathbb{R} \). Thus, the direction of steepest descent corresponds to the direction \( d_k \) which maximises \( \text{Re}\langle d_k, r_k \rangle \) while keeping the length of \( d_k \) fixed. Since \( d_k \) is only determined up to a scalar factor, we can choose

\[
d_k = r_k \quad \forall k \in \mathbb{N}_0
\]

in (6.13). With this choice, the step length becomes

\[
\alpha_k = \frac{\|r_k\|^2}{\|r_k\|_A^2}
\]

and we get the following algorithm.
Algorithm SD (steepest descent).
input: \( A \in \mathbb{C}^{n \times n} \) Hermitian, positive definite, \( b \in \mathbb{C}^n \), \( x_0 \in \mathbb{C}^n \), \( \varepsilon_r > 0 \)
output: \( x_k \in \mathbb{C}^n \) with \( Ax_k \approx b \)
1. for \( k = 1, 2, \ldots \) do
2. \( r_{k-1} = b - Ax_{k-1} \)
3. if \( \| r_{k-1} \| \leq \varepsilon_r \) then
   4. return \( x_{k-1} \)
5. end if
6. \( \alpha_{k-1} = \| r_{k-1} \| ^2 / \| r_{k-1} \| _A ^2 \)
7. \( x_k = x_{k-1} + \alpha_{k-1} r_{k-1} \)
8. end for

Error Analysis

The following result establishes the convergence of the method.

**Theorem 6.10.** Let \( A \) be Hermitian and positive definite. Then the rate of convergence of the steepest descent algorithm for solving (SLE) is given by

\[
\| x_k - x \|_A \leq \sqrt{1 - 1/\kappa(A)} \| x_0 - x \|_A \quad \forall k \in \mathbb{N},
\]

where \( \kappa(A) \) is the condition number of \( A \) in the 2-norm.

**Proof.** By substituting \( \varepsilon = \alpha_{k-1} \) into equation (6.17) we get

\[
g(x_k) = g(x_{k-1}) - \frac{1}{2} \frac{(d_{k-1}, r_{k-1})^2}{\| d_{k-1} \| _A ^2} = \left( 1 - \frac{(d_{k-1}, r_{k-1})^2}{\| d_{k-1} \| _A ^2 \| r_{k-1} \| _A ^2} \right) g(x_{k-1})
\]

for all \( k \in \mathbb{N} \). Together with the estimates from Lemma 1.21 we get

\[
g(x_k) \leq \left( 1 - \frac{\| r_{k-1} \| _A ^4}{\lambda_{\text{max}} \| r_{k-1} \| _A ^2 \lambda_{\text{min}} \| r_{k-1} \| ^2} \right) g(x_{k-1}) = \left( 1 - \frac{1}{\kappa(A)} \right) g(x_{k-1})
\]

for all \( k \in \mathbb{N} \) and thus

\[
g(x_k) \leq \left( 1 - \frac{1}{\kappa(A)} \right) ^k g(x_0).
\]

The result follows then with \( \| e_k \| _A ^2 = 2g(x_k) \).

**Remarks.**

1. If the matrix \( A \) is ill-conditioned then the method converges slowly. Ill-conditioning occurs in the 2-norm when the eigenvalues of \( A \) are very different: \( \lambda_{\text{max}} \gg \lambda_{\text{min}} \). In this case the steepest descent direction is almost orthogonal to the direction in which the solution lies.

2. Since the steepest descent method uses \( d_k = r_k \), the search directions satisfy \( r_k = r_{k-1} - \alpha_{k-1} A r_{k-1} \) by (6.16). Using induction we get

\[
x_k \in x_0 + \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\}
\]

for all \( k \in \mathbb{N} \).

**Computational Complexity**

We can achieve

\[
\| x_k - x \|_A \leq \varepsilon
\]

by choosing \( k \) to be the smallest integer greater than

\[
k^* = \frac{\ln \varepsilon^{-2} + \ln \| x_0 - x \| _A ^2}{\ln \left( 1 - \kappa(A)^{-1} \right)}
\]

77
Assumption 6.11.  \begin{enumerate}
\item Calculation of $Ax$ costs $O(n^2)$ uniformly in $x \in \mathbb{C}^n$.
\item The condition number of $A$ in the 2-norm satisfies $\kappa(A) = O(n^2)$.
\item $\|x-x_0\|_A$ is bounded uniformly in $n$.
\end{enumerate}

Theorem 6.12. Under Assumption 6.11 the computational cost of using the steepest descent method to achieve (6.19) is bounded above by $cn^{\max\{a,1\}+\beta}\ln\varepsilon^{-1}$, for some constant $c$ independent of $n$ and $\varepsilon$.

\begin{proof}
The first item of the assumption ensures that each step of the iteration costs $\Theta(n^{\max\{a,1\}})$. Hence, combining the second and third items and using (6.20) gives the desired result. \end{proof}

6.6 The Conjugate Gradient Method

Theorem 6.10 shows that the steepest descent method can behave quite poorly on ill-conditioned problems. The conjugate gradient method, and its pre-conditioned variants, go a long way to ameliorating this difficulty, at little extra cost when compared to the method of steepest descents.

Again, we assume that $A$ is Hermitian and positive definite. The conjugate gradient method is similar to the steepest descent method but instead of making steps in direction of the steepest descent it restricts the increments $x_k - x_{k-1}$ to lie in the Krylov subspace

$$K_k(A,r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^k r_0\}.$$  

Since the spaces $K_k(A,r_0)$ are increasing, this leads to $x_k \in x_0 + K_k(A,r_0)$ for all $k$ and, as we will see, the algorithm even chooses $x_k$ to minimise the function $g$ from (6.12) on this space.

An underlying, general approach to the design of iterative methods for systems of linear equations (and also for the eigenvalue problem) is as follows: seek approximate solutions which at the $k^{\text{th}}$ step of the iteration use only increments in $K_k(A,r_0)$ such methods are natural when matrix-vector multiplication by $A$ is cheap to compute. Equation (6.18) shows that the steepest descent method follows this principle. The conjugate gradient method can be implemented almost as easily as the method of steepest descents, but since it finds the optimal approximation to $x$ from $K_k(A,r_0)$ we expect it to perform better. Indeed, the CG method converges considerably faster on ill-conditioned problems. Many other iterative methods also use the idea of finding approximations from the Krylov subspace.

The Method

Definition 6.13. Given $A$ Hermitian positive-definite we say that the set $\{d_0, \ldots, d_k\}$ is $A$-orthogonal (or conjugate) if

$$\langle d_i, d_j \rangle_A = 0 \quad \forall i \neq j.$$  

The following Lemma shows that choosing the search directions $d_k$ to be $A$-orthogonal is advantageous for non-linear iterative methods.

Lemma 6.14. Assume that the search directions $\{d_0, \ldots, d_{k-1}\}$ form an $A$-orthogonal set. Then $x_k$, given by (6.13), minimises $g$ over $x_0 + \text{span}\{d_0, \ldots, d_{k-1}\}$.

\begin{proof}
Define $G: \mathbb{C}^n \rightarrow \mathbb{R}$ by

$$G(\gamma) = g(x_0 + \sum_{l=0}^{k-1} \gamma_l d_l) \quad \forall \gamma \in \mathbb{C}^n.$$  

Then $G$ is a convex quadratic function and has a unique minimum where all directional derivatives vanish. Using (6.15) with $y = x_0 + \sum_{l=0}^{k-1} \gamma_l d_l$ and $z = \beta d_m$ we find

$$\text{Re} \beta \langle d_m, Ar_0 + \sum_{l=0}^{k-1} \gamma_l Ad_l - b \rangle = 0.$$  

78
for all $\beta \in \mathbb{C}$ and thus
\[ \gamma_m = \frac{\langle d_m, r_0 \rangle}{\|d_m\|_A^2} \]
for $m = 0, \ldots, k - 1$.

Using (6.16) and the conjugacy of the $d_l$ we get
\[ \langle d_m, r_k \rangle = \langle d_m, r_{k-1} \rangle - \alpha_{k-1} \langle d_m, Ad_{k-1} \rangle = \langle d_m, r_{k-1} \rangle \]
for $k = 1, \ldots, m$ and thus $\langle d_m, r_0 \rangle = \langle d_m, r_m \rangle$. This shows that the minimum is, indeed, attained for $\gamma = \alpha$.

The steepest descent method also chooses $x_k \in x_0 + \text{span}\{d_0, \ldots, d_{k-1}\}$, but minimises $g$ only along a line instead of over all of the affine sub-space. Thus, any method with $A$-orthogonal search directions will be at least as good as the steepest descent method.

In general it is expensive to construct orthogonal sets something like a Gram-Schmidt orthogonalisation is required. Thus to make a viable method it remains to try and construct an iteration which iteratively generates an $A$-orthogonal set in a cheap fashion. The idea of the conjugate gradient method is to construct the $A$-orthogonal set by using the iteration
\[ d_k = r_k + \beta_k d_{k-1} \]
where the factor $\beta_k$ is chosen to enforce $\langle d_{k-1}, d_k \rangle_A = 0$, giving
\[ \beta_k = -\frac{\langle d_{k-1}, r_k \rangle_A}{\|d_{k-1}\|_A^2}. \]
This clearly ensures that $d_k$ is orthogonal to $d_{k-1}$; we will see that it is possible to choose $d_0$ such that the entire set $\{d_0, \ldots, d_k\}$ is an $A$-orthogonal set.

A preliminary form of the resulting algorithm is given by the following procedure: choose $x_0, d_0 \in \mathbb{C}^n$ and let $r_0 = b - Ax_0$. Then, for $k = 1, 2, \ldots$ define, in the given order,
\[ \alpha_{k-1} = \frac{\langle d_{k-1}, r_{k-1} \rangle}{\|d_{k-1}\|_A^2}, \quad x_k = x_{k-1} + \alpha_{k-1} d_{k-1}, \]
\[ r_k = r_{k-1} - \alpha_{k-1} Ad_{k-1}, \quad \beta_k = \frac{\langle d_{k-1}, r_k \rangle_A}{\|d_{k-1}\|_A^2}, \quad d_k = r_k + \beta_k d_{k-1}. \hspace{1cm} (6.21) \]
Here we have used (6.16) to compute $r_k$ inductively. This allows to compute all required quantities with only one matrix-vector multiplication per step (to calculate $Ad_{k-1}$).

**Lemma 6.15.** Let $x_1, \ldots, x_k$ be given by the conjugate gradient method (6.21) with $d_0 = r_0$. Assume $r_0, \ldots, r_k \neq 0$ and $d_0, \ldots, d_{k-1} \neq 0$. Then $d_k \neq 0$ and $\langle d_{k-1}, r_k \rangle = \langle r_{k-1}, r_k \rangle = 0$.

Furthermore
\[ \alpha_{k-1} = \frac{\|r_{k-1}\|^2}{\|d_{k-1}\|_A^2} > 0, \quad \beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|_A^2} > 0. \]

**Proof.** From the definition of $r_k$ and the definition of $\alpha_{k-1}$ we get
\[ \langle d_{k-1}, r_k \rangle = \langle d_{k-1}, r_{k-1} \rangle - \alpha_{k-1} \|d_{k-1}\|_A^2 = \langle d_{k-1}, r_{k-1} \rangle - \langle d_{k-1}, r_{k-1} \rangle = 0 \]
and thus, by Pythagoras’ theorem,
\[ \|d_k\|^2 = \|r_k\|^2 + \|d_{k-1}\|^2 - \alpha_{k-1} \|d_{k-1}\|_A^2 \geq \|r_k\|^2 > 0 \]
and $d_k \neq 0$. For $k = 1$ we have $\langle r_{k-1}, d_{k-1} \rangle_A = \langle d_0, d_0 \rangle_A = \|d_{k-1}\|_A^2$. For $k > 1$ we can solve the equation defining $d_{k-1}$ for $r_{k-1}$ to get $\langle r_{k-1}, d_{k-1} \rangle_A = \langle d_{k-1} - \beta_{k-1} d_{k-2}, d_{k-1} \rangle_A = \|d_{k-1}\|_A^2$. Hence
\[ \langle r_{k-1}, r_k \rangle = \langle r_{k-1}, r_{k-1} \rangle - \alpha_{k-1} \langle r_{k-1}, Ad_{k-1} \rangle = 0 \]
which is the second orthogonality relation from the claim.
For \( k = 1 \) we have \( \langle d_{k-1}, r_{k-1} \rangle = \langle r_0, r_0 \rangle = \| r_{k-1} \|^2 \). For \( k > 1 \) we can use the definition of \( d_{k-1} \) to get \( \langle d_{k-1}, r_{k-1} \rangle = \langle r_{k-1} + \beta_{k-1} d_{k-2}, r_{k-2} \rangle = \| r_{k-1} \|^2 \). Thus we get

\[
\alpha_{k-1} = \frac{\langle d_{k-1}, r_{k-1} \rangle}{\| d_{k-1} \|^2} = \frac{\| r_{k-1} \|^2}{\| d_{k-1} \|^2} > 0.
\]

Finally, since \( \alpha_{k-1} > 0 \), we can solve the definition of \( r_k \) for \( \beta_k \)

\[
\beta_k = \frac{\langle -A d_{k-1}, r_k \rangle}{\| d_{k-1} \|^2} = \frac{\langle r_k - r_{k-1}, r_k \rangle}{\alpha_{k-1}\| d_{k-1} \|^2} = \frac{\| r_k \|^2}{\| r_{k-1} \|^2} > 0.
\]

This completes the proof.

A consequence of Lemma 6.15 is that, assuming we have not already found the exact solution, we will have \( d_k \neq 0 \) and thus all the expressions in (6.21) make sense. We can now prove that the conjugate gradient method does indeed produce an \( A \)-orthogonal set of search directions.

**Lemma 6.16.** Let \( x_1, \ldots, x_k \) be given by the conjugate gradient method (6.21) with \( d_0 = r_0 \). Then \( \{d_0, \ldots, d_{k-1}\} \subseteq K_k(A, r_0) \) is an \( A \)-orthogonal set.

**Proof.** We first prove by induction that

\[
\text{span}\{r_0, r_1, \ldots, r_l\} = \text{span}\{d_0, d_1, \ldots, d_l\} = \text{span}\{r_0, Ar_0, \ldots, A^l r_0\} \tag{6.22}
\]

for \( l = 0, \ldots, k-1 \). Since \( d_0 = r_0 \), the statement holds for \( l = 0 \). Now let \( l > 0 \) and assume that the statement holds for \( l - 1 \). Then, since \( \alpha_{l-1} \neq 0 \), we can solve the definition of \( r_l \) for \( \beta_{l-1} \)

to get \( A d_{l-1} \in \text{span}\{r_{l-1}, r_l\} \) and thus

\[
A^l r_0 = A(A^{l-1} r_0) \in \text{span}\{Ad_0, \ldots, Ad_{l-1}\} \subseteq \text{span}\{r_0, \ldots, r_l\}.
\]

This shows \( \text{span}\{r_0, Ar_0, \ldots, A^l r_0\} \subseteq \text{span}\{r_0, r_1, \ldots, r_l\} \). From the definition of \( d_k \) we get

\[
r_l = d_l - \beta d_{l-1} \in \text{span}\{d_{l-1}, d_l\} \subseteq \text{span}\{d_0, d_1, \ldots, d_l\}
\]

and thus \( \text{span}\{r_0, r_1, \ldots, r_l\} \subseteq \text{span}\{d_0, d_1, \ldots, d_l\} \). Finally we have

\[
d_l = r_l + \beta d_{l-1} = -\alpha_{l-1} Ad_{l-1} + r_{l-1} + \beta d_{l-1} \in \text{span}\{r_0, \ldots, A^l r_0\}
\]

and thus \( \text{span}\{d_0, d_1, \ldots, d_l\} \subseteq \text{span}\{r_0, Ar_0, \ldots, A^l r_0\} \). This completes the proof of (6.22).

Assume for induction that \( \{d_0, \ldots, d_{k-1}\} \) is an \( A \)-orthogonal set. This is true for \( k = 1 \). Now let \( k > 1 \). From the definition of \( r_{j+1} \) we get

\[
\langle d_i, r_{j+1} \rangle = \langle d_i, r_j \rangle - \alpha_m \langle d_i, d_j \rangle A = \langle d_i, r_j \rangle
\]

for \( j = i+1, \ldots, k-1 \) and thus, by induction, \( \langle d_i, r_j \rangle = \langle d_i, r_{i+1} \rangle = 0 \) for all \( 0 \leq i < j \leq k \) where the last identity comes from Lemma 6.15. Since \( Ad_i \in \text{span}\{r_i, r_{i+1}\} \subseteq \text{span}\{d_0, \ldots, d_{i+1}\} \), we get

\[
\langle d_i, r_k \rangle A = \langle Ad_i, r_k \rangle = 0
\]

for \( i = 0, \ldots, k-2 \). Using this result we can prove that \( d_k \) is \( A \)-orthogonal to \( d_0, \ldots, d_{k-1} \): for \( i < k - 1 \) we get

\[
\langle d_i, d_k \rangle A = \langle d_i, r_k \rangle A + \beta_k \langle d_i, d_{k-1} \rangle A = \langle d_i, r_k \rangle A = 0
\]

and \( \langle d_{k-1}, d_k \rangle = 0 \) by the construction of \( \beta_k \). This completes the proof.

The CG-algorithm is normally implemented in the following form which exploits the expressions for \( \alpha_k, \beta_k \) as derived in Lemma 6.15. This allows the required number of arithmetic operations to be kept small.
Algorithm CG (conjugate gradient method).
input: $A \in \mathbb{C}^{n \times n}$ Hermitian, positive definite, $b \in \mathbb{C}^n$, $x_0 \in \mathbb{C}^n$, $\varepsilon_r > 0$
output: $x_k \in \mathbb{C}^n$ with $x_k \approx A^{-1}b$

1: $r_0 = b - Ax_0$, $d_0 = r_0$
2: for $k = 1, 2, 3, \ldots$ do
3: \[ \alpha_{k-1} = \frac{\|r_{k-1}\|^2}{\|d_{k-1}\|^2_A} > 0 \]
4: \[ x_k = x_{k-1} + \alpha_{k-1}d_{k-1} \]
5: \[ r_k = r_{k-1} - \alpha_{k-1}Ad_{k-1} \]
6: if $\|r_k\| \leq \varepsilon_r$ then
7: \[ \text{return } x_k \]
8: end if
9: \[ \beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2} > 0 \]
10: \[ d_k = r_k + \beta_kd_{k-1} \]
end for

Error Analysis
We have already seen, in Lemma 6.14, that any method with $A$-orthogonal search directions will have smaller errors than the steepest descent method. The following theorem shows a surprising result: these methods, even though iterative by construction, obtain the exact solution of (SLE) after at most $n$ iterations.

**Theorem 6.17.** If \{ $d_0, \ldots, d_n$ \} form an $A$-orthogonal set in $\mathbb{C}^n$ then, for any starting vector $x_0$, the sequence $(x_k)$ given by (6.13) reaches the exact solution $x$ of (SLE) in at most $n$ steps.

**Proof.** The set \{ $d_0, \ldots, d_n$ \} forms an $A$-orthogonal basis for $\mathbb{C}^n$ and so we may write

\[ x - x_0 = \sum_{k=0}^{n-1} \gamma_k d_k, \quad \gamma_k = \frac{(d_k, x - x_0)_A}{\|d_k\|^2_A}. \]

Also, by construction of the $x_k$,

\[ x_k - x_0 = \sum_{i=0}^{k-1} \alpha_i d_i. \]

Conjugacy gives

\[ (d_k, x_k - x_0)_A = 0 \]
and thus

\[ (d_k, x - x_0)_A = (d_k, x - x_0)_A + (d_k, x_0 - x_k)_A = (d_k, x - x_k)_A = (d_k, r_k) \]
for $k = 0, \ldots, n - 1$. Comparing the definitions of $\alpha_k$ and $\gamma_k$ we find $\alpha_k = \gamma_k$ which implies

\[ x - x_0 = \sum_{k=0}^{n-1} \alpha_k d_k = x_n - x_0 \]
so that $x = x_n$ as required. \qed

**Corollary 6.18.** The conjugate gradient algorithm, for any starting vector $x_0$, reaches the exact solution of (SLE) in at most $n$ steps.

**Proof.** Assume that the algorithm has not converged in $n - 1$ steps; otherwise the proof is complete. Then the set \{ $d_0, \ldots, d_{n-1}$ \} is $A$-orthogonal by Lemma 6.16. Hence the result follows from Theorem 6.17. \qed
Remark. When the computation is performed on a computer, rounding errors will cause \( \hat{r}_n \neq 0 \) where \( \hat{r}_n \) is the calculated value for the residual error \( r_n \). In practice one just treats the method as an iterative method and continues the iteration until the residual error \( \| \hat{r}_k \| \) is small enough. Typically the exit criterion will even be reached for \( k < n \).

In order to understand how the conjugate gradient method performs iteratively, we describe an error analysis based around the theory of polynomial approximation. In the following we let

\[ A(A) := \{ \lambda \in \mathbb{C} \mid \lambda \text{ is an eigenvalue of } A \} \]

and we let \( \mathcal{P}_k \) denote the set of all polynomials of degree \( k \) with \( p(0) = 1 \).

**Theorem 6.19.** If the conjugate gradient algorithm has not converged at step \( k \), then

\[
\| x_k - x \|_A = \inf_{p \in \mathcal{P}_k} \| p(A)(x_0 - x) \|_A \\
\leq \left( \inf_{p \in \mathcal{P}_k} \max_{\lambda \in A(A)} |p(\lambda)| \right) \| x_0 - x \|_A.
\]

**Proof.** Since \( x_k \in x_0 + \text{span}\{ r_0, A r_0, \ldots, A^{k-1} r_0 \} \) we can find \( \gamma_j \in \mathbb{C} \) with

\[
e_k = x - x_k = x - x_0 + \sum_{j=0}^{k-1} \gamma_j A^j r_0 = e_0 + \sum_{j=0}^{k-1} \gamma_j A^{j+1} e_0.
\]

Defining \( p \in \mathcal{P}_k \) by

\[ p(\lambda) = 1 + \sum_{j=1}^{k} \gamma_{j-1} \lambda^j \]

we obtain \( e_k = p(A)e_0 \) and thus

\[ \| x_k - x \|_A = \| p(A)(x_0 - x) \|_A. \]

The value \( x_k \) minimises \( \| x_k - x \|_A \) over \( x_k \in \mathcal{K}_k( A, b ) \), which is equivalent to minimising over all choices of \( \gamma_j \). Hence

\[ \| x_k - x \|_A = \inf_{p \in \mathcal{P}_k} \| p(A)(x_0 - x) \|_A. \]

To complete the proof let \( \{ \psi_j \} \) be a basis of \( \mathbb{C}^n \) composed of eigenvectors of \( A \) with corresponding eigenvalues \( \lambda_j \). If we let \( e_0 = \sum_{j=1}^{n} a_j \psi_j \), then

\[ e_k = p(A)e_0 = \sum_{j=1}^{n} a_j p(\lambda_j) \psi_j \]

and thus we get

\[ \| e_0 \|^2_A = \sum_{j=1}^{n} \lambda_j |a_j|^2, \quad \| e_k \|^2_A = \sum_{j=1}^{n} \lambda_j |a_j p(\lambda_j)|^2. \]

Therefore

\[ \| e_k \|^2_A \leq \max_{\lambda \in \Lambda(A)} |p(\lambda)|^2 \sum_{j=1}^{n} |a_j|^2 \lambda_j = \max_{\lambda \in \Lambda(A)} |p(\lambda)|^2 \| e_0 \|^2_A \]

and the result follows. \( \square \)

By choosing an appropriately scaled and shifted Chebyshev polynomial and applying the previous result, the following may be shown:

**Theorem 6.20.** Let \( \kappa \) be the condition number of a symmetric positive definite matrix \( A \) in the Euclidean norm. Then the error in step \( k \) of the conjugate gradient algorithm for solving (SLE) satisfies

\[
\| x_k - x \|_A \leq 2 \left[ \left( \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \right)^k + \left( \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \right)^{-k} \right] \| x_0 - x \|_A.
\]

**Proof.** See Exercise 6-6. \( \square \)
Computational Complexity

We make the same assumptions as for the steepest descent method, and show the decrease in computational cost that can be obtained by using the conjugate gradient method.

**Theorem 6.21.** Under Assumption 6.11 the computational cost of using the conjugate gradient method to achieve (6.19) is bounded above by $cn^{\max\{\alpha,1\}} + \frac{1}{2} \beta \ln \varepsilon^{-1}$, for some constant $c$ independent of $n$ and $\varepsilon$.

**Proof.** The first item of the assumption ensures that each step of the iteration costs $\Theta(n^{\max\{\alpha,1\}})$. If $\sqrt{\kappa} \gg 1$ then the Theorem 6.20 gives

$$\|x_k - x\|_A \lesssim 2 \left(1 - \frac{2}{\sqrt{\kappa}}\right)^k \|x_0 - x\|_A.$$ 

Combining the second and third items and using this expression gives the desired result. \qed

**Remark.** In an applied context the most important form of the conjugate gradient method is its preconditioned form. Roughly speaking the method is applied to the linear system

$$MAx = Mb$$

where the preconditioner $M$ is chosen to try and improve the conditioning of $MA$ over $A$, yet have $M$ easy to invert. In practical terms the algorithm differs from standard conjugate gradient only by the addition of one multiplication with $M$ per step.

**Bibliography**

Linear iterative methods for (SLE) are described in the book [Saa97]. Theorem 6.7 is proved in [SB02]. For discussion of Theorem 6.20 see [TB97]; for related results see [NW06]. The SOR method is analysed in detail in [LT88].

The nonlinear iterative methods that we discuss are constructed through minimisation or optimisation techniques. As such they are strongly related to the topic of optimisation in general. This topic is well overviewed in [NW06]; the exposition of the conjugate gradient method there forms the basis for ours. A good discussion of preconditioned versions of the conjugate gradient method may also be found in this book.

**Exercises**

**Exercise 6-1.** The purpose of this exercise is to introduce you to the idea of iteration. Many problems are simply too large to solve by the methods we have looked at so far — too much computer time or memory is required if the dimension $n$ of the matrix is large. For linear systems

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad b \in \mathbb{R}^n \quad (6.23)$$

this will lead to the idea of generating approximating sequences $\{x_k\}$ which hopefully converge to $x$ as $k \to \infty$. For eigenvalue problems which are large we will study methods which generate approximations to a subset of the eigenvalues, and not to all of them. (Note that for eigenvalue problems it is necessary to use iterative methods in all cases, provided the matrix has dimension bigger than 4; the main difference when the problem is large is that we solve for only a subset of the eigenvalues).

(i) To show why iterative methods are necessary construct a matrix $A$ with $n = 1000$ as follows: type

```matlab
>> A=0.0005*rand(1000);
>> b=rand(1000,1);
>> for i=1:1000; A(i,i)=1; end;
```

83
This creates a matrix \( A \) with 1 on the diagonal and random entries uniformly distributed in \([0, 5 \times 10^{-4}]\) on the off-diagonals. Now try solving (6.23). Type

\[
\texttt{>> x=A\backslash b;}
\]

Almost certainly your computer cannot handle a problem of this size. So type \texttt{ctrl c} to stop it from trying. (If you are lucky enough to be working on a machine where this problem is solved easily then repeat the exercise replacing 1000 by 10000 or by some matrix sufficiently large that \texttt{MATLAB} fails.) Keep the (original \( 1000 \times 1000 \)) matrix \( A \) for use in (iii) where we will use a more successful approach to solve this problem.

(ii) We look at the simplest iterative method, known as Jacobi. Any matrix \( A \) can be written uniquely as

\[
A = D + L + U
\]

where \( D \) is diagonal, \( L \) is zero on and above the diagonal and \( U \) is zero on and below the diagonal. Assuming that \( D \) is invertible, consider a sequence \( \{x_k\}_{k=1}^\infty \) satisfying

\[
x_{k+1} = D^{-1}[b - Lx_k - Ux_k].
\]

If \( x_k \to x^* \) as \( k \to \infty \) then what equation does \( x^* \) satisfy and why?

(iii) Implement the idea in (ii) as follows. Type

\[
\texttt{>> L=tril(A,-1);
>> U=triu(A,1);
>> 0=L+U;
}
\]

This creates \( L \) and \( U \) as defined in (ii) (a \texttt{MATLAB} question: why is this so?). Type

\[
\texttt{>> j=1;
>> y=ones(1000,1);
>> n(j)=2*j;
>> for i=1:n(j); y=b-0*y; end
>> rn(j)=norm(A*y-b)
}
\]

What does this tell you? Repeat the above 5 lines of code with \( j = 2, 3, \ldots, 5 \) and type:

\[
\texttt{>> plot(n,rn)}
\]

What do you observe? Also type

\[
\texttt{>> plot(n,log(rn))}
\]

(iv) Under what conditions on \( A \) will the Jacobi iteration converge to the correct solution of (6.23)?

**Exercise 6-2.** Prove that the Jacobi iteration that you implemented in Exercise 6-1 converges for the matrix \( A \) given there.

**Exercise 6-3.** Which of the following matrices are irreducible? For which of these matrices does the Jacobi-method converge?

\[
\begin{pmatrix}
4 & 2 & 1 \\
2 & 1 & \\
1 & & 
\end{pmatrix}, \quad \begin{pmatrix}
1 & 3 \\
3 & 25 & 4 \\
4 & 1 & 
\end{pmatrix}, \quad \begin{pmatrix}
8 \\
4 & 2 \\
2 & 1 & 1 
\end{pmatrix}
\]

**Exercise 6-4.** Show that the Jacobi method for the solution of

\[
\begin{pmatrix}
1 & 1 & 2 \\
1 & 10 & 2 \\
2 & 1 & 
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 
\end{pmatrix}
= \begin{pmatrix}
1 \\
2 \\
3 
\end{pmatrix}
\]

converges and, for the iteration starting with \( x_0 = 0 \), give an upper bound on the number of steps required to get the relative error of the result below \( 10^{-6} \).
Exercise 6-5. Consider the linear system $Ax = b$ where
$$A = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix},$$
where $b = (1, \varepsilon)^T$. Calculate $x_1$ from the steepest descents algorithm, starting from $x_0 = 0$. Which component of the solution is better approximated after one step? Why is this?

Exercise 6-6. Prove Theorem 6.20 by choosing a Chebyshev polynomial scaled and shifted so that its min/max properties hold on the interval $[\lambda_{\min}(A), \lambda_{\max}(A)]$. Then prove that
$$\|x_k - x\|_A \leq 2^k \|x_0 - x\|_A$$
where $\delta \in (0, 1)$ and
$$\delta \approx 1 - 2 \sqrt{\frac{\mu_{\min}}{\mu_{\max}}}$$
for $\kappa \gg 1$.

Exercise 6-7. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive-definite matrix. If $A = QM\tilde{Q}^T$ is the eigenvalue/eigenvector decomposition of $A$, then we define a square root of $A$ (one of many possible square roots) to be $A^{1/2} = QD\tilde{Q}^T$, where $D$ is a diagonal matrix whose elements are the square roots of the corresponding elements of $M$.

1. If $B \in \mathbb{R}^{n \times n}$ is a matrix such that $B$ and $A^{1/2}$ commute, prove that
$$\|Bx\|_A \leq \|B\|_2 \|x\|_A, \quad \forall x \in \mathbb{R}^n$$

2. Using your proof above, prove that
$$\inf_{p \in \mathcal{P}_k} \|p(A)y\|_A \leq \inf_{p \in \mathcal{P}_k} \max_{\lambda \in \Lambda(A)} |p(\lambda)| \|y\|_A$$
where $\mathcal{P}_k$ is the set of all polynomials $p$ of degree less than or equal to $k$ with $p(0) = 1$, $y \in \mathbb{R}^n$, and $\Lambda(A)$ is the set of eigenvalues of $A$.

Exercise 6-8. Define
$$\varphi(x) := \frac{1}{2} \langle x, Ax \rangle - \langle x, b \rangle$$
with $A$ symmetric positive definite. Show that minimising $\varphi(x)$ is equivalent to solving $Ax = b$.

Consider the algorithm
$$x_{k+1} = x_k + \alpha_k r_k$$
to minimise $\varphi(x)$. Find $\alpha_k$ which minimises $\varphi(x_{k+1})$ over all $\alpha_k \in \mathbb{R}$ given $x_k$. This choice of $\alpha_k$ defines the gradient or steepest descent algorithm.

For CG we have
$$x_{k+1} = x_k + \alpha_k d_k.$$