LINEAR SYSTEMS OF EQUATIONS

1. Introduction

We look for solutions of systems of equations of the form $Ax = b$, where A is an $n \times n$ matrix and x and b are $n-$column vectors

The matrix A will be assumed to be non-singular—i.e., A^{-1} exists—so that $x = A^{-1}b$

However, calculation of A^{-1} is too expensive for this approach to be used in practice

2. Gauss elimination

The matrix A is said to be upper triangular if $a_{ij} = 0,\ i > j,\ i.e.$ all terms below the diagonal are zero; similarly A is lower triangular if $a_{ij} = 0,\ i < j$; and diagonal if $a_{ij} = 0,\ i \neq j$.

If A is upper or lower triangular, then the system $Ax = b$ is solved using the *backward* or *forward substitution algorithm* respectively.

Backward substitution:

```
SUBROUTINE BACKSUB(A,B,X,N)
DIMENSION A(N,N),B(N),X(N)
X(N)=B(N)/A(N,N)
DO I=N-1,1,-1
     SUM=0.0
     DO J=I+1,N
          SUM=SUM+A(I,J)*X(J)
     END DO
     X(I)=(B(I)-SUM)/A(I,I)
END DO
RETURN
END
```

presuming $a_{ii} \neq 0$ for any i.

The *Gauss elimination algorithm* involves transforming the coefficient matrix A to upper triangular form using the *forward elimination algorithm*, and solving the resultant system using backward substitution. Forward elimination systematically eliminates all subdiagonal entries of A and adjusts b accordingly:

Forward elimination:

```
SUBROUTINE FORELIM(A,B,N)
DIMENSION A(N,N),B(N),M(N,N)
DO K=1,N-1 ! loop over the columns of A
     DO I=K+1,N  ! loop over the rows of A
          M(I,K)=A(I,K)/A(K,K)
          DO J=K+1,N
               A(I,J)=A(I,J)-M(I,K)*A(K,,J)
```

```
            END DO
            B(I)=B(I)-M(I,K)*B(K)
        END DO
    END DO
    RETURN
    END
```

Here again, it is assumed that the entry a_{kk} is never zero at any stage of the elimination; this is not true in general, and, in fact, even small values can cause problems when using floating point arithmetic.

Usually, the elements m_{ik}, which are called the *multipliers*, are stored in the sub-diagonal part of A; this is accomplished by replacing any reference to m_{ik} by a_{ik}, e.g., the statement

`M(I,K)=A(I,K)/A(K,K)`

in the above algorithm by the statement

`A(I,K)=A(I,K)/A(K,K)`

The upper triangular part of the array A naturally stores the upper triangular matrix resulting from the forward elimination.

2.1. Operation count. At the kth step of the forward elimination, A is in the form:

$$\begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & * & * & * \\ & & & * & * & * \end{pmatrix}$$

Work involved in kth step:

(1) Calculate multipliers:

$$m_{ik} = \frac{a_{ik}}{a_{kk}} \; i = k+1, \ldots, n$$

This involves $n-k$ divisions—or 1 division and $n-k$ multiplications.

(2) Elimination:

$$a_{ij} = a_{ij} - m_{ik}a_{kj} \; i = k+1, \ldots, n, \; j = k+1, \ldots, n$$

This takes $(n-k)^2$ (additions + multiplications).

(3) Right hand side:

$$b_i = b_i - m_{ik}b_k \; i = k+1, \ldots, n$$

$(n-k)$ (additions + multiplications).

Let $D = 1$ division; $A = 1$ addition; and $M = 1$ multiplication; then, summing the number of operations in each of the stages from $k = 1$ to $k = n-1$, we have:

the total operations calculating the multipliers =

$$\sum_{k=1}^{n-1}(n-k)D = \sum_{i=1}^{n-1} iD = \frac{n(n-1)}{2}D$$

Total operations in elimination =

$$\sum_{k=1}^{n-1}(n-k)^2(A+M) = \sum_{i=1}^{n-1} i^2(A+M) = \frac{n(n-1)(2n-1)}{6}(A+M)$$

Total operations for right hand side =

$$\sum_{k=1}^{n-1}(n-k)(A+M) = \frac{n(n-1)}{2}(A+M)$$

Similarly, the total number of operations for backward substitution =

$$\begin{aligned}\sum_{k=1}^{n}[(n-k)(A+M)+1D] &= \sum_{k=1}^{n-1}(n-k)(A+M)+nD \\ &= \frac{n(n-1)}{2}(A+M)+nD\end{aligned}$$

Conventionally, 1 operation $= 1(A+M) = 1D$; so, elimination of the lower triangular part of the matrix takes

$$\begin{aligned}\frac{n(n-1)(2n-1)}{6}+\frac{n(n-1)}{2} &= \frac{(2n^3-3n^2+n)+3n^2-3n}{6} \\ &= \frac{n^3-n}{3} \text{ operations.}\end{aligned}$$

For large n, $n^3 >> n^2 >> n$; so the matrix elimination takes approximately $n^3/3$ operations, whereas elimination of the right hand side and backward substitution together take $n(n-1)(A+M)+nD = n^2$ operations. For large n, this is insignificant in comparison to $n^3/3$.

3. Matrix factorisation

Given a *set* of linear systems:

$$Ax_1 = b_1,\ Ax_2 = b_2, \ldots, Ax_m = b_m$$

one efficient method of solution is to combine the vectors x_i and b_i $i = 1, \ldots, m$ into two $n \times m$ matrices:

$$X = (x_1, x_2, \ldots, x_m)\ \ B = (b_1, b_2, \ldots, b_m)$$

and solve $AX = B$ by Gauss elimination.
This cannot be done, however, if for example $b_i = f(x_{i-1}),\ i = 2, \ldots, m$.
In this situation, the problems must be solved consecutively; but repeated elimination of the matrix A is not necessary, if we use *LU-decomposition*:

3.1. *LU*-decomposition. The matrix A is factorised as the product of alower triangular matrix L and an upper triangular matrix U, so that $A = LU$.
So, to solve $Ax_i = b_i$ for any i, we solve $LUx_i = b_i$ in two stages:

$$\begin{aligned}Ly_i &= b_i \text{ by Forward Substitution} \\ Ux_i &= y_i \text{ by Backward Substitution}\end{aligned}$$

This takes only $n^2 + n$ operations. The matrix U is the upper triangular matrix obtained from Forward Elimination of A.

The matrix $L = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ m_{21} & 1 & 0 & 0 & \ldots & 0 \\ m_{31} & m_{32} & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ldots & \vdots \\ m_{n-1\,1} & m_{n-1\,2} & m_{n-1\,3} & \ldots & 1 & 0 \\ m_{n1} & m_{n2} & m_{n3} & \ldots & m_{n\,n-1} & 1 \end{pmatrix}$

where the elements m_{ik} are the multipliers from the forward elimination.

L and U are thus both obtained from the forward elimination, and are both stored in the array originally assigned to A, L—apart from its diagonal which does not need to be stored as all elements are 1—below the diagonal, and U in the remainder.

Example.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 5 \\ 3 & 5 & 9 \end{pmatrix}$$

Forward elimination:

$$k = 1 \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 0 & 2 & 6 \end{pmatrix}$$

$$k = 2 \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{pmatrix}$$

Note that

$$A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{pmatrix}$$

and LU is actually stored as $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 3 \\ 3 & 1 & 3 \end{pmatrix}$ So, to solve e.g., $Ax = \begin{pmatrix} 1 \\ 3 \\ 7 \end{pmatrix}$, solve $Ly = \begin{pmatrix} 1 \\ 3 \\ 7 \end{pmatrix}$:

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 7 \end{pmatrix}$$

So, $y_1 = 1$, $y_2 = 3 - 2y_1 = 1$, $y_3 = 7 - 3y_1 - y_2 = 3$, and then solve $Ux = y$:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}$$

So, $x_3 = 1$, $x_2 = (1 - 3x_2)/2 = -1$, $x_1 = 1 - x_2 - x_3 = 1$.

REMARK 3.1. This factorisation of A still requires that $a_{kk} \neq 0$ during the elimination.

REMARK 3.2. An alternative way to obtain L and U is to solve directly the equation $LU = A$, e.g., here

$$\begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 5 \\ 3 & 5 & 9 \end{pmatrix}$$

which gives:

$$\begin{aligned} u_{11} &= u_{12} = u_{13} = 1 \\ m_{21}u_{11} &= 2 \Rightarrow m_{21} = 2 \\ m_{21}u_{12} + u_{22} &= 4 \Rightarrow u_{22} = 2 \\ m_{21}u_{13} + u_{23} &= 5 \Rightarrow u_{23} = 3 \\ m_{31}u_{11} &= 3 \Rightarrow m_{31} = 3 \\ m_{31}u_{12} + m_{32}u_{22} &= 5 \Rightarrow m_{32} = 1 \\ m_{31}u_{13} + m_{32}u_{23} + u_{33} &= 9 \Rightarrow u_{33} = 3 \end{aligned}$$

as before. This variation is sometimes known as *Doolittle's method*.

3.2. Cholesky decomposition. A more useful variation can be used when the matrix is symmetric and positive-semidefinite—i.e. $x^TAx \geq 0 \forall x \neq 0$—in which case the matrix A may be decomposed 'symmetrically' as $A = LL^T$.

This is known as *Cholesky decomposition* and its advantages when it is applicable are twofold: computation of L involves approximately half the work of finding L and U; also, only the upper- (or lower-) triangular part of the original matrix A need be stored if it is symmetric; and if only L is stored there is no extra requirement for U.

Example. Find the Cholesky decomposition—$A = LL^T$—of the matrix

$$A = \begin{pmatrix} 2 & -2 & -3 \\ -2 & 5 & 4 \\ -3 & 4 & 5 \end{pmatrix}$$

and hence, solve the system of equations $Ax = b$, where $b = \begin{pmatrix} 7 \\ -12 \\ -12 \end{pmatrix}$.

Solution:

$$A = \begin{pmatrix} 2 & -2 & -3 \\ -2 & 5 & 4 \\ -3 & 4 & 5 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}$$

Thus $l_{11}^2 = 2 \Rightarrow l_{11} = \sqrt{2}$; $l_{11}l_{21} = -2 \Rightarrow l_{21} = -\sqrt{2}$; $l_{11}l_{31} = -3 \Rightarrow l_{31} = -3/\sqrt{2}$;
$l_{21}^2 + l_{22}^2 = 5 \Rightarrow l_{22} = \sqrt{5-2} = \sqrt{3}$; $l_{21}l_{31} + l_{22}l_{32} = 4 \Rightarrow l_{32} = (4-3)/\sqrt{3} = 1/\sqrt{3}$;
$l_{31}^2 + l_{32}^2 + l_{33}^2 = 5 \Rightarrow l_{33} = \sqrt{5 - 9/2 - 1/3} = 1/\sqrt{6}$
And so

$$L = \begin{pmatrix} \sqrt{2} & 0 & 0 \\ -\sqrt{2} & \sqrt{3} & 0 \\ -3/\sqrt{2} & 1/\sqrt{3} & 1/\sqrt{6} \end{pmatrix}$$

Thus $Ax = b \Rightarrow LL^Tx = b$. First solve $Ly = b$ i.e.,

$$\begin{pmatrix} \sqrt{2} & 0 & 0 \\ -\sqrt{2} & \sqrt{3} & 0 \\ -3/\sqrt{2} & 1/\sqrt{3} & 1/\sqrt{6} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 7 \\ -12 \\ -12 \end{pmatrix}$$

Thus, $y_1 = 7/\sqrt{2}$, $y_2 = (-12+7)/\sqrt{3} = -5/\sqrt{3}$, and $y_3 = (-12 + 21/2 + 5/3)\sqrt{6} = 1/\sqrt{6}$.

Then solve $L^T x = y$, i.e.,

$$\begin{pmatrix} \sqrt{2} & -\sqrt{2} & -3/\sqrt{2} \\ 0 & \sqrt{3} & 1/\sqrt{3} \\ 0 & 0 & 1/\sqrt{6} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7/\sqrt{2} \\ -5/\sqrt{3} \\ 1/\sqrt{6} \end{pmatrix}$$

Thus $x_3 = 1$, $x_2 = (-5/\sqrt{3} - 1/\sqrt{3})/\sqrt{3} = -2$, $x_3 = (7/\sqrt{2} - 2\sqrt{2} + 3/\sqrt{2})/\sqrt{2} = 3$

Thus the solution is $\begin{pmatrix} 3 \\ -2 \\ 1 \end{pmatrix}$ If the matrix A is not positive-semidefinite, some of the elements of L will be complex, so it is not often used in such a circumstance.

4. Error in solution

In solving the linear system of equations $Ax = b$, due to the presence of rounding error, we *may* obtain only an approximate solution $\bar{x}$ rather than the exact solution $\hat{x}$—i.e., $A\hat{x} = b$ but $A\bar{x} \neq b$.
The error vector $\varepsilon = \hat{x} - \bar{x}$ cannot be measured since $\hat{x}$ is not known. What can be measured, however, is the *residual*:

$$\begin{aligned} r &= b - A\bar{x} \\ &= A\hat{x} - A\bar{x} \\ &= A(\hat{x} - \bar{x}) = A\varepsilon \end{aligned}$$

If $r = 0$, then $\varepsilon = 0$—since A is assumed to be non-singular, and if the error ε is small, the residual r is small; however, the converse is not true, i.e., small r does *not* imply small ε.

4.1. Ill-conditioning. Example.

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix} \quad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}$$

Assume we have calculated $\bar{x} = \begin{pmatrix} 0.9911 \\ -0.4870 \end{pmatrix}$, using Gauss elimination;

then $r = b - A\bar{x} = \begin{pmatrix} 10^{-8} \\ 10^{-8} \end{pmatrix}$, which is "small".

However, the true solution $\hat{x} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$, so that the error $\varepsilon = \begin{pmatrix} 1.0089 \\ -1.513 \end{pmatrix}$, which is not small.
If we examine the elimination process, the one elimination step required for the matrix A gives:

$$\begin{pmatrix} 1.2969 & 0.8648 \\ 0 & 7.7 \times 10^{-9} \end{pmatrix}$$

The new a_{22} is almost 0, so A is very nearly singular and is said to be *ill-conditioned*—i.e., a small change in the term 0.1441 would lead to a big change in $\bar{x}$.

4.2. Norms. To quantify this idea we use some vector norm to measure the size of the vectors involved in some sense. The most easily calculable ones are

- The *Euclidean norm*: $\|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$. This norm is often denoted $\|x\|_2$.
- The *maximum norm*: $\|x\| = \max\limits_{1 \leq i \leq n} |x_i|$, which is often denoted $\|x\|_\infty$.

To any vector norm $\|x\|$ there corresponds a matrix norm called the *natural* or *induced* matrix norm:

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

which satisfies the basic norm properties and also, for any $n \times n$ matrices A and B:

$$\begin{aligned} \|Ax\| &\leq \|A\| \, \|x\| \\ \|AB\| &\leq \|A\| \, \|B\| \end{aligned}$$

The norm used for measuring the size of the error is the maximum norm; it can be shown that the corresponding matrix norm is the *maximum row-sum norm*:

$$\|A\| = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$$

This is also often denoted $\|A\|_\infty$

4.3. The condition number. Returning to the calculation of the error, we have the following two pairs of equations:

$$\begin{aligned} &1.(\text{a}) \; A\hat{x} = b \quad 1.(\text{b}) \; \hat{x} = A^{-1}b \\ &2.(\text{a}) \; A\varepsilon = r \quad 2.(\text{b}) \; \varepsilon = A^{-1}r \end{aligned}$$

Using the norm properties, we obtain the following:

$$\begin{aligned} 1.(\text{a}) \Rightarrow \|b\| &= \|A\hat{x}\| \leq \|A\| \, \|\hat{x}\| && 3. \\ 1.(\text{b}) \Rightarrow \|\hat{x}\| &= \|A^{-1}b\| \leq \|A^{-1}\| \, \|b\| && 4. \end{aligned}$$

$$\begin{aligned} 2.(\text{a}) \Rightarrow \|r\| &= \|A\varepsilon\| \leq \|A\| \, \|\varepsilon\| && 5. \\ 2.(\text{b}) \Rightarrow \|\varepsilon\| &= \|A^{-1}r\| \leq \|A^{-1}\| \, \|r\| && 6. \end{aligned}$$

Combining 5 and 6 $\Rightarrow$

$$\frac{\|r\|}{\|A\|} \leq \|\varepsilon\| \leq \|A^{-1}\| \, \|r\|$$

and combining 3 and 4 $\Rightarrow$

$$\frac{1}{\|A^{-1}\| \, \|b\|} \leq \frac{1}{\|\hat{x}\|} \leq \frac{\|A\|}{\|b\|}$$

and combining these two inequalities, we have:

$$\frac{1}{\|A\| \, \|A^{-1}\|} \frac{\|r\|}{\|b\|} \leq \frac{\|\varepsilon\|}{\|\hat{x}\|} \leq \|A\| \, \|A^{-1}\| \frac{\|r\|}{\|b\|}$$

or

$$\frac{1}{\kappa(A)} \frac{\|r\|}{\|b\|} \leq \frac{\|\hat{x} - \bar{x}\|}{\|\hat{x}\|} \leq \kappa(A) \frac{\|r\|}{\|b\|}$$

where $\kappa(A) = \|A\| \, \|A^{-1}\|$ is called the *condition number* of A.

What we have is an upper and lower bound for the relative error in the solution. From the norm property 5 $\|A\| \, \|A^{-1}\| \geq \|AA^{-1}\| = \|I\| = 1$—the identity matrix I always has norm 1—so $\kappa(A) \geq 1$. If $\kappa(A) = 1$ then we can measure the relative error exactly:

$$\frac{\|\hat{x} - \bar{x}\|}{\|\hat{x}\|} = \frac{\|r\|}{\|b\|}$$

Also, if $\kappa(A)$ is small then a small residual $\Rightarrow$ a small error; however, for large $\kappa(A)$—an ill-conditioned matrix—it is *possible* for the residual to be small and yet for the error to be large.

In the example

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix} \quad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}$$

and so

$$A^{-1} = 10^8 \begin{pmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{pmatrix}$$

Thus the norm of A $\|A\| = \max(2.1617, 0.3602) = 2.1617$ and the norm of A^{-1} $\|A^{-1}\| = \max(1.0089 \times 10^8, 1.513 \times 10^8) = 1.513 \times 10^8$. So, $\kappa(A) \approx 3.27 \times 10^8$; so the matrix is ill-conditioned.

Since ill-conditioning is a property of the given system of equations, it may not be possible to cope with it in the particular precision used. In the example, double precision could be used; but, for $\kappa(A) = 10^{20}$ say, the problem is essentially insoluble.

Since it may be important to know the order of magnitude of the condition number of a matrix A, and since the definition $\kappa(A) = \|A\| \, \|A^{-1}\|$ refers to the inverse of A^{-1} which is too expensive to calculate, we can estimate $\kappa(A)$ as follows:

Take k different vectors y_i $i = 1, \ldots, k$ e.g., $y_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ $y_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ etc. and solve $Aw_i = y_i$ $i = 1, \ldots, k$.

Since A has been factorised as $A = LU$ this only involves kn^2 operations; so, it is economical to do this for $k << n$.

We can calculate $\|w_i\|$ and $\|y_i\|$ for all i, and since $w_i = A^{-1} y_i$ for all i, we have:

$$\begin{aligned} \|w_i\| &\leq \|A^{-1}\| \, \|y_i\| \\ \Rightarrow \|A^{-1}\| &\geq \|w_i\| / \|y_i\| \end{aligned}$$

and so, we use the approximation:

$$\|A^{-1}\| \approx \max_{1 \leq i \leq k} \left(\frac{\|w_i\|}{\|y_i\|} \right)$$

and use this to estimate $\kappa(A)$. Of course, where applicable, the Cholesky decomposition $A = LL^T$ may be used in the same way.

5. Iterative methods

Iterative or *indirect* methods—as opposed to *direct methods* like Gauss elimination and its variations—produce a sequence $x^{(k)}$ of approximations to the solution of $Ax = b$, where $\lim_{k \to \infty} x^{(k)} = \hat{x}$, and $\hat{x}$ is the true solution. The simplest such method is the *Jacobi method*.

5.1. Jacobi method. The system of equations $Ax = b$ can be written

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} x_j &= b_i \; i = 1, \ldots, n \\ \Rightarrow a_{ii} x_i &= -\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j + b_i \; i = 1, \ldots, n \\ \Rightarrow x_i &= \frac{1}{a_{ii}} \Big[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \Big] \; i = 1, \ldots, n \end{aligned}$$

From this equation for the components of x we obtain the iterative method:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\big[b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^{(k)}\big] \; i = 1, \ldots, n$$

This is the Jacobi method, where each element of the new vector $x^{(k+1)}$ is obtained from all the other elements of the old vector $x^{(k)}$.

This is how the elements of the successive approximation vectors are calculated, but to analyse the method it is helpful to write it in vector form:

$$D = \begin{pmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & \ddots & & \\ & & & a_{n-1\,n-1} & \\ & & & & a_{nn} \end{pmatrix}$$

is the diagonal part of A.

So, in vector form the Jacobi method is:

$$x^{(k+1)} = D^{-1}b + (I - D^{-1}A)x^{(k)}$$

REMARK 5.1. that in the calculation of $x_i^{(k+1)}$, even though the new elements $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ are available, they are not used. This also means that both the old and the new vectors must be stored until all the elements of the new vector $x^{(k+1)}$ have been calculated.

5.2. Gauss-Seidel method. If these new elements replace their old counterparts as soon as they are calculated, we have the Gauss-Seidel method:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\big[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\big] \; i = 1, \ldots, n$$

If we write $L = \begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{pmatrix}$ the lower triangular part of A and $U = A - L$, so that $A = L + U$—this should not be confused with the LU-decomposition of A—we can write the Gauss-Seidel method as

$$\begin{aligned} \sum_{j=1}^{i} a_{ij}x_j^{(k+1)} &= b_i - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \; i = 1, \ldots, n \\ \Rightarrow \; Lx^{(k+1)} &= b - Ux^{(k)} \\ \Rightarrow \; x^{(k+1)} &= L^{-1}b - L^{-1}Ux^{(k)} \\ &= L^{-1}b - L^{-1}(A - L)x^{(k)} \\ &= L^{-1}b + (I - L^{-1}A)x^{(k)} \end{aligned}$$

5.3. General iterative methods. A general iterative method for the solution of $Ax = b$ is:

$$x^{(k+1)} = Gx^{(k)} + Hb$$

where the *iteration matrix* G satisfies $G = I - HA$, and H is some approximate inverse of A, i.e.,

$$\begin{aligned} x^{(k+1)} &= Hb + (I - HA)x^{(k)} \\ &= x^{(k)} + H(b - Ax^{(k)}) \\ &= x^{(k)} + H\mathbb{R}^{(k)} \end{aligned}$$

For the Jacobi method $H = D^{-1}$ and for Gauss-Seidel $H = L^{-1}$.

DEFINITION 5.2. The *spectral radius* of a matrix M denoted $\rho(M) = \max\limits_{1 \le i \le n} |\lambda_i|$ where $\{\lambda_i\}$ is the set of eigenvalues —the *spectrum* of M.

REMARK 5.3. $\rho(M)$ has the special property that for any natural matrix norm $\|M\|$ on M, $\rho(M) \le \|M\|$.

Example. $M = \begin{pmatrix} 1 & 4 \\ 3 & 2 \end{pmatrix}$. The characteristic equation is:

$$\begin{aligned} \begin{vmatrix} 1-\lambda & 4 \\ 3 & 2-\lambda \end{vmatrix} &= 0 \\ \Rightarrow (1-\lambda)(2-\lambda) - 12 &= 0 \\ \Rightarrow \lambda^2 - 3\lambda - 10 &= 0 \\ \Rightarrow (\lambda - 5)(\lambda + 2) &= 0 \Rightarrow \lambda_1 = 5, \lambda_2 = -2 \Rightarrow \rho(M) = 5 \end{aligned}$$

The maximum row-sum norm $\|M\| = \max(5, 5) = 5 = \rho(M)$.

5.4. Convergence criterion.

THEOREM 5.4. *For a general iterative method $x^{(k+1)} = Gx^{(k)} + Hb$, convergence is guaranteed for any initial guess $x^{(0)}$ to the solution $\hat{x}$ of $Ax = b$ if and only if $\rho(G) < 1$.*

PROOF. This is proved only for the special case where G has a full set of n linearly independent eigenvectors $\{\omega_i\}_{i=1,\ldots,n}$:
The solution $\hat{x}$ satisfies $\hat{x} = G\hat{x} + Hb$—since $A\hat{x} = b \Rightarrow HA\hat{x} = Hb \Rightarrow \hat{x} = (I - HA)\hat{x} + Hb$. Define the error vector $\varepsilon^{(k)} = x^{(k)} - \hat{x}$. Then:

$$\varepsilon^{(k+1)} = G\varepsilon^{(k)} = G^2\varepsilon^{(k-1)} = G^3\varepsilon^{(k-2)} = \ldots = G^{k+1}\varepsilon^{(0)}$$

where $\varepsilon^{(0)} = x^{(0)} - \hat{x}$. So:

$$\varepsilon^{(k)} = G^k\varepsilon^{(0)} \Rightarrow \|\varepsilon^{(k)}\| \le \|G^k\|\,\|\varepsilon^{(0)}\| \le \|G\|^k\|\varepsilon^{(0)}\|$$

So, if $\|G\| < 1$ then $\lim_{k\to\infty} \|\varepsilon^{(k)}\| = 0$ for any $\varepsilon^{(0)}$, i.e., $x^{(k)} \to \hat{x}$ as $k \to \infty$ for any $x^{(0)}$.
We can now use the fact that $\{\omega_i\}_{i=1,\ldots,n}$ are linearly independent to write $\varepsilon^{(0)}$ as $\varepsilon^{(0)} = \sum_{i=1}^n \mu_i\omega_i$ for some constants μ_i. So:

$$\varepsilon^{(k)} = G^k\varepsilon^{(0)} = \sum_{i=1}^n \mu_i G^k\omega_i = \sum_{i=1}^n \mu_i\lambda_i^k\omega_i$$

where λ_i is the eigenvalue corresponding to ω_i (i.e., $G\omega_i = \lambda_i\omega_i$ for all $i = 1, \ldots, n$). Therefore, $\varepsilon^{(k)} \to 0$ as $k \to \infty$ if and only if $|\lambda_i| < 1$ for all i; i.e., if and only if $\rho(G) < 1$. □

Example.

$$\begin{aligned} x_1 + x_3 &= 2 \\ x_2 + x_3 &= 3 \\ x_2 + 2x_3 &= 5 \end{aligned}$$

In this case $A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ and we examine the convergence of the Jacobi method, where:

$$\begin{aligned} G &= I - D^{-1}A \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & \frac{1}{2} & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & -\frac{1}{2} & 0 \end{pmatrix} \end{aligned}$$

The characteristic equation

$$|\lambda I - G| = 0 \Rightarrow \begin{vmatrix} \lambda & 0 & 1 \\ 0 & \lambda & 1 \\ 0 & \frac{1}{2} & \lambda \end{vmatrix} = 0$$

$$\Rightarrow \lambda(\lambda^2 - \frac{1}{2}) = 0 \Rightarrow \lambda = 0, \frac{1}{\sqrt{2}} \text{ or } -\frac{1}{\sqrt{2}}$$

In each case, $|\lambda| < 1$; so, the Jacobi method works in this case for any initial guess.

5.5. Special cases. There are two types of matrix A for which convergence can be guaranteed without having to examine the iteration matrix:

- If the matrix A is *positive definite*—i.e., if for all vectors x $x^T Ax \geq 0$ and $x^T Ax = 0 \iff x = 0$ (or, equivalently if all eigenvalues of A have positive real part)—then both the Jacobi and Gauss-Seidel methods converge for any initial $x^{(0)}$. There are certain situations such as the solution of partial differential equations where the matrices involved are known to be of this type; but, in general, it is difficult to determine whether or not a matrix is positive definite.
- Convergence is also guaranteed for both methods if the matrix A is *strictly diagonally dominant* i.e., if

  $$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \; i = 1, \ldots, n$$

  This property can be determined by observation and is thus more easily used.

5.6. The SOR method. The Gauss-Seidel method can be generalised as follows:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right] \; i = 1, \ldots, n$$

where ω is a parameter satisfying $0 < \omega < 2$. There are three distinct cases:

$\omega = 1$**:** This is just the Gauss-Seidel case.

$\omega < 1$**:** This is called *Successive Under-relaxation*.

$\omega > 1$**:** This is called *Successive Over-Relaxation* or simply *SOR*.

SOR is more commonly used than under-relaxation as it is an attempt to speed up the convergence of the iteration method.

For certain types of problems, usually in the solution of partial differential equations, an optimal value of ω is known or can be determined which will ensure the fastest possible convergence.

Example.

$$\begin{aligned} 4x_1 + x_3 + x_4 &= 1 \\ 4x_2 + x_4 &= 2 \\ x_1 + 4x_3 &= 3 \\ x_1 + x_2 + 4x_4 &= 4 \end{aligned}$$

The exact solution—to 6 decimal places— is: $x_1 = -.196172$, $x_2 = .253589$, $x_3 = .799043$, $x_4 = .985646$.

Jacobi method:

$$\begin{aligned} x_1^{(k+1)} &= (1 - x_3^{(k)} - x_4^{(k)})/4 \\ x_2^{(k+1)} &= (2 - x_4^{(k)})/4 \\ x_3^{(k+1)} &= (3 - x_1^{(k)})/4 \\ x_4^{(k+1)} &= (4 - x_1^{(k)} - x_2^{(k)})/4 \end{aligned}$$

SOR:

$$\begin{aligned} x_1^{(k+1)} &= (1-\omega)x_1^{(k)} + \omega(1 - x_3^{(k)} - x_4^{(k)})/4 \\ x_2^{(k+1)} &= (1-\omega)x_2^{(k)} + \omega(2 - x_4^{(k)})/4 \\ x_3^{(k+1)} &= (1-\omega)x_3^{(k)} + \omega(3 - x_1^{(k+1)})/4 \\ x_4^{(k+1)} &= (1-\omega)x_4^{(k)} + \omega(4 - x_1^{(k+1)} - x_2^{(k+1)})/4 \end{aligned}$$

Gauss-Seidel is obtained from SOR with $\omega = 1$. Using as initial guess $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_4^{(0)} = 0$, and carrying out 5 iterations of each method, the following results are obtained:

k	0	1	2	3	4	5
$x_1^{(k)}$	0	.25	-.1875	-.125000	-.195313	-.184570
$x_2^{(k)}$	0	.50	.2500	.296875	.253906	.260742
$x_3^{(k)}$	0	.75	.6875	.796875	.781250	.798828
$x_4^{(k)}$	0	1.00	.8125	.984375	.957031	.985352

TABLE 1. Jacobi method

k	0	1	2	3	4	5
$x_1^{(k)}$	0	.2500	-.125000	-.184570	-.194275	-.195862
$x_2^{(k)}$	0	.5000	.296875	.260742	.254761	.253780
$x_3^{(k)}$	0	.6875	.781250	.796143	.798569	.798965
$x_4^{(k)}$	0	.8125	.957031	.980957	.984879	.985520

TABLE 2. Gauss Seidel

k	0	1	2	3	4	5
$x_1^{(k)}$	0	.262500	-.160617	-.194335	-.196121	-.196172
$x_2^{(k)}$	0	.525000	.250000	.277389	.254620	.253624
$x_3^{(k)}$	0	.718594	.793732	.798826	.799040	.799043
$x_4^{(k)}$	0	.843281	.977183	.985316	.985640	.985646

TABLE 3. **SOR** ($\omega = 1.05$)