Iterative methods for solving linear systems

Linear Algebra, Michaelmas Term 2018
Jonathan Whiteley

Solving sparse linear systems

Suppose we want to solve the matrix equation

$$A\mathbf{u} = \mathbf{b}$$

where A is a given non-singular $N \times N$ matrix, and $\mathbf{b}$ is a vector of length N

Many linear systems that must be solved in practice have very large N, but only a small number of non-zeros per row

These matrices are known as sparse matrices

Sparse matrices are seen when modelling particles joined by springs, modelling tissue development, and in molecular dynamics simulations

If N is large then it may not be possible to store all entries of a sparse matrix A

It may, however, be possible to store only the non-zero entries and their location

This leads to a problem when solving the linear system — although A is sparse, the LU-decomposition of A isn't sparse

How can linear systems such as these be solved?

Iterative methods

Suppose we want to solve the linear system

$$A\mathbf{u} = \mathbf{b}$$

for a given matrix A, and given vector $\mathbf{b}$

When using iterative methods to estimate $\mathbf{u}$ we start with an initial guess $\mathbf{u}_0$

We then iterate — i.e. we generate (hopefully better) approximations $\mathbf{u}_1, \mathbf{u}_2, \ldots$

We stop when $\|A\mathbf{u}_n - \mathbf{b}\| < \epsilon$ for some sufficiently small ϵ

We will see that this approach allows us to solve large, sparse systems of equations

Some linear systems that can be solved easily

Before deriving some iterative methods for solving linear systems we will identify some classes of linear systems that can be solved easily without calculating the inverse of the matrix or the LU factorisation

These classes of equations will be used in iterative methods

Diagonal systems of equations

Suppose D is a diagonal matrix, of size $N \times N$, with all diagonal entries non-zero

We can then write D as

$$D = \begin{pmatrix} D_{11} & 0 & 0 & \dots & 0 \\ 0 & D_{22} & 0 & \dots & 0 \\ 0 & 0 & D_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & D_{NN} \end{pmatrix}$$

Suppose we want to solve the linear system

$$D\mathbf{u} = \mathbf{b}$$

We may write this linear system as

$$\begin{aligned} D_{11}u_1 &= b_1 \\ D_{22}u_2 &= b_2 \\ D_{33}u_3 &= b_3 \\ \vdots \quad &\vdots \\ D_{NN}u_N &= b_N \end{aligned}$$

As we have assumed that all diagonal entries are non-zero, the solution of this equation is

$$\begin{aligned} u_1 &= \frac{b_1}{D_{11}} \\ u_2 &= \frac{b_2}{D_{22}} \\ u_3 &= \frac{b_3}{D_{33}} \\ \vdots &\vdots \vdots \\ u_N &= \frac{b_N}{D_{NN}} \end{aligned}$$

We note that this linear system is very simple to solve — it requires only N divisions

Aside: since $\det(D) = D_{11}D_{22}D_{33}\dots D_{NN}$, we see that the matrix would have been singular if any entry on the diagonal was zero

Solving triangular systems of equations

Suppose A is a lower triangular matrix, so that

$$A = \begin{pmatrix} A_{11} & 0 & 0 & \dots & 0 \\ A_{21} & A_{22} & 0 & \dots & 0 \\ A_{31} & A_{32} & A_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & A_{N3} & \dots & A_{NN} \end{pmatrix}$$

Noting that $\det(A) = A_{11}A_{22}A_{33}\dots A_{NN}$ we see that if A is non-singular then the diagonal entries of A are non-zero

The linear system $A\mathbf{u} = \mathbf{b}$ may then be written

$$\begin{pmatrix} A_{11} & 0 & 0 & \dots & 0 \\ A_{21} & A_{22} & 0 & \dots & 0 \\ A_{31} & A_{32} & A_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & A_{N3} & \dots & A_{NN} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{pmatrix}$$

This may be solved by writing

$$\begin{aligned} u_1 &= \frac{1}{A_{11}} b_1 \\ u_2 &= \frac{1}{A_{22}} (b_2 - A_{21}u_1) \\ u_3 &= \frac{1}{A_{33}} (b_3 - A_{31}u_1 - A_{32}u_2) \\ \vdots\ &\ \vdots \\ u_N &= \frac{1}{A_{NN}} (b_N - A_{N1}u_1 - A_{N2}u_2 - \dots - A_{N,N-1}u_{N-1}) \end{aligned}$$

Note that both diagonal systems of equations and lower-triangular systems of equations may be easily solved without needing to perform any LU factorisation of the matrices

Similar remarks apply to upper-triangular matrices

We now think about the number of operations required to solve a lower triangular system

Calculating u_1 requires 1 division

Calculating u_2 requires 1 division, 1 subtraction and 1 multiplication

Calculating u_3 requires 1 division, 2 subtractions and 2 multiplications

Calculating u_n requires 1 division, $n-1$ subtractions and $n-1$ multiplications

Calculating u_N requires 1 division, $N-1$ subtractions and $N-1$ multiplications

Recall that

$$\sum_{n=1}^{N-1} n = \frac{(N-1)N}{2}$$

We then see that solving a lower triangular system requires N divisions, $\frac{1}{2}N(N-1)$ subtractions and $\frac{1}{2}N(N-1)$ multiplications

Iterative methods

Suppose we want to solve the linear system

$$A\mathbf{u} = \mathbf{b}$$

for a given matrix A, and given vector $\mathbf{b}$

When using iterative methods to estimate $\mathbf{u}$ we start with an initial guess $\mathbf{u}_0$

We then iterate — i.e. we generate (hopefully better) approximations $\mathbf{u}_1, \mathbf{u}_2, \ldots$

We stop when $\|A\mathbf{u}_n - \mathbf{b}\| < \epsilon$ for some sufficiently small ϵ

When using many iterative methods it is convenient to decompose the matrix A into the sum of the entries below the diagonal, the entries on the diagonal, and the entries above the diagonal

We will write

$$A = D - L - U$$

where

- $-L$ contains the entries of A strictly below the diagonal
- D contains the entries of A on the diagonal
- $-U$ contains the entries of A strictly above the diagonal

Suppose

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Then

$$L = \begin{pmatrix} 0 & 0 & 0 \\ -4 & 0 & 0 \\ -7 & -8 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -2 & -3 \\ 0 & 0 & -6 \\ 0 & 0 & 0 \end{pmatrix}$$

The Jacobi method

Our linear system is

$$A\mathbf{u} = \mathbf{b}$$

where

$$A = D - L - U$$

Equivalently,

$$D\mathbf{u} = (L + U)\mathbf{u} + \mathbf{b}$$

We can then define an iterative scheme — known as the Jacobi method — by

$$\begin{aligned} \mathbf{u}_0 &= \text{initial guess} \\ D\mathbf{u}_n &= (L+U)\mathbf{u}_{n-1} + \mathbf{b}, \qquad n = 1, 2, \ldots \end{aligned}$$

The linear system that must be solved on each step is a diagonal linear system

We explained earlier that diagonal linear systems are computationally cheap to solve provided all diagonal entries are non-zero

An example of Jacobi's method

Suppose we apply Jacobi's method to the linear system

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

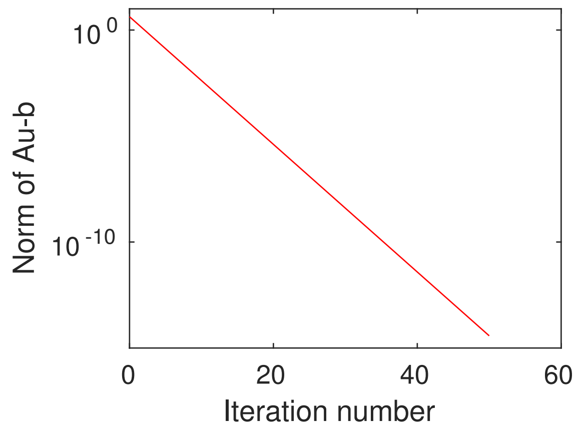
This linear system has solution $\mathbf{u} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

Using the notation from earlier,

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -0.5 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0 \end{pmatrix}$$

and we start with the initial guess $\mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

We now plot the quantity $\|\mathbf{b} - A\mathbf{u}_n\|$ —known as the magnitude of the residual — as a function of iteration number



We see that the Jacobi method works well in this case

Another example of Jacobi's method

Suppose we apply Jacobi's method to the linear system

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

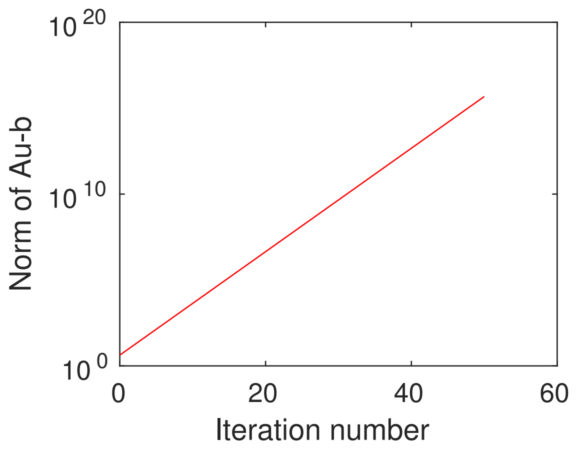
This linear system has solution $\mathbf{u} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Using the notation from earlier,

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix}$$

and we start with the initial guess $\mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

We now plot the quantity $\|\mathbf{b} - A\mathbf{u}_n\|$ as a function of iteration number



This time the residual diverges, and the Jacobi method fails

The number of operations required when using Jacobi's method

We set $M = L + U$, and $\mathbf{v} = M\mathbf{u}_{n-1} + b$

The Jacobi method then becomes $D\mathbf{u}_n = \mathbf{v}$

Each v_i is then given by

$$v_i = b_i + \sum_{j=1}^{N} M_{ij} (u_i)_{n-1}$$

which requires N additions and N multiplications

As $\mathbf{v}$ has N entries, calculating $\mathbf{v}$ therefore requires N^2 additions and N^2 multiplications

Solving $D\mathbf{u}_n = \mathbf{v}$ then requires a further N divisions

Each iteration of Jacobi's method therefore requires N^2 additions, N^2 multiplications, and N divisions

The Gauss-Seidel method

Our linear system is

$$A\mathbf{u} = \mathbf{b}$$

where

$$A = D - L - U$$

Equivalently,

$$(D - L)\mathbf{u} = U\mathbf{u} + \mathbf{b}$$

We can then define an iterative scheme — known as the Gauss-Seidel method — by

$$\begin{aligned} \mathbf{u}_0 &= \text{initial guess} \\ (D - L)\mathbf{u}_n &= U\mathbf{u}_{n-1} + \mathbf{b}, \qquad n = 1, 2, \ldots \end{aligned}$$

The linear system that must be solved on each step is a lower triangular linear system

We explained earlier that lower triangular linear systems are computationally cheap to solve provided all diagonal entries are non-zero

An example of the Gauss-Seidel method

We now apply the Gauss-Seidel method to the linear system

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

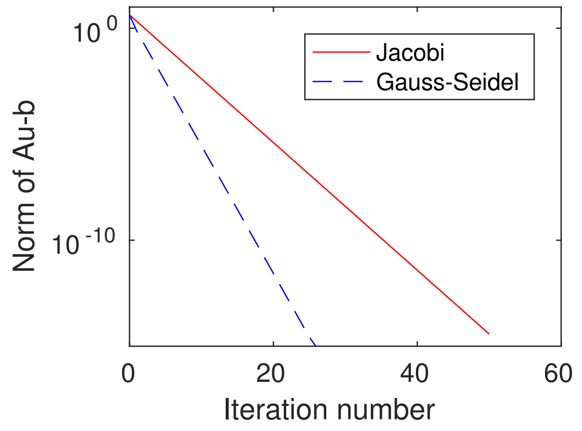
This linear system has solution $\mathbf{u} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

Using the notation from earlier,

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -0.5 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0 \end{pmatrix}$$

and we start with the initial guess $\mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

We now plot the quantity $\|\mathbf{b} - A\mathbf{u}_n\|$ as a function of iteration number, and compare with Jacobi's method



We see that the Gauss-Seidel method works well in this case, and converges faster than Jacobi's method

Another example of the Gauss-Seidel method

Suppose we apply the Gauss-Seidel method to the linear system

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

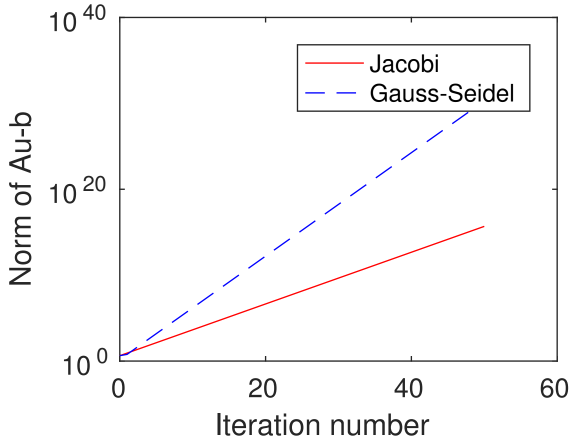
This linear system has solution $\mathbf{u} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Using the notation from earlier,

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix}$$

and we start with the initial guess $\mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

We now plot the quantity $\|\mathbf{b} - A\mathbf{u}_n\|$ as a function of iteration number, and compare with Jacobi's method



This time both the Gauss-Seidel and the Jacobi methods fail

The successive over relaxation (SOR) method

Recall that our linear system is

$$A\mathbf{u} = \mathbf{b}$$

where

$$A = D - L - U$$

We may write the Gauss-Seidel scheme as

$$\begin{aligned} \mathbf{u}_0 &= \text{initial guess} \\ D\mathbf{u}_n &= U\mathbf{u}_{n-1} + L\mathbf{u}_n + \mathbf{b}, \qquad n = 1, 2, \ldots \end{aligned}$$

Suppose we approximate $D\mathbf{u}_n$ by

$$\begin{aligned} D\mathbf{u}_n &= (1-\omega) \times \{\text{approximation to } D\mathbf{u} \text{ at iteration } n-1\} \; + \\ &\qquad \omega \times \{\text{Gauss-Seidel approximation to } D\mathbf{u} \text{ at iteration } n\} \\ &= (1-\omega)D\mathbf{u}_{n-1} + \omega\left(U\mathbf{u}_{n-1} + L\mathbf{u}_n + \mathbf{b}\right) \end{aligned}$$

where ω is a parameter chosen by the user

This is known as the successive over relaxation or SOR for short

Note that when $\omega = 1$ the SOR method is identical to the Gauss-Seidel method

The SOR iteration method may be written

$$\mathbf{u}_0 = \text{initial guess}$$
$$(D-\omega L)\mathbf{u}_n = ((1-\omega)D+\omega U)\,\mathbf{u}_{n-1}+\omega\mathbf{b}, \qquad n=1,2,\ldots$$

Note that this requires only the solution of a lower-triangular linear system

An example of SOR

We now apply the SOR method to the linear system

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

This linear system has solution $\mathbf{u} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

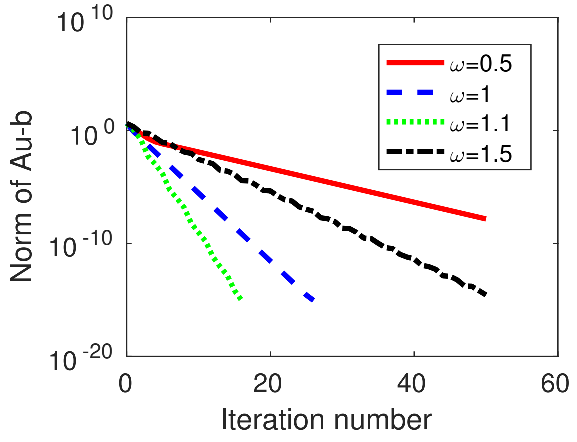
Using the notation from earlier,

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -0.5 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0 \end{pmatrix}$$

and we start with the initial guess $\mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

We are free to choose the parameter ω

Below we show convergence for different values of ω



The choice of ω can clearly have a significant effect on the convergence of SOR

The convergence rate is not a monotonic function of ω

Is there a systematic way of choosing ω?

Error analysis for iterative methods

The iterative step of the Jacobi method is

$$D\mathbf{u}_n = (L+U)\mathbf{u}_{n-1} + \mathbf{b}, \qquad n = 1, 2, \ldots$$

This may be written as

$$\mathbf{u}_n = D^{-1}(L+U)\mathbf{u}_{n-1} + D^{-1}\mathbf{b}, \qquad n = 1, 2, \ldots$$

We will write this as

$$\mathbf{u}_n = G\mathbf{u}_{n-1} + \mathbf{c}, \qquad n = 1, 2, \ldots$$

where

$$G = D^{-1}(L+U), \qquad \mathbf{c} = D^{-1}\mathbf{b}$$

Remembering that the true solution $\mathbf{u}$ satisfies

$$A\mathbf{u} = \mathbf{b}$$

and that

$$A = D - L - U$$

we may write

$$D\mathbf{u} = (D+L)\mathbf{u} + \mathbf{b}$$

or, using G and $\mathbf{c}$ defined on the previous slide,

$$\mathbf{u} = G\mathbf{u} + \mathbf{c}$$

We now know that the iterative solution satisfies

$$\mathbf{u}_n = G\mathbf{u}_{n-1} + \mathbf{c}, \qquad n = 1, 2, \ldots$$

and that the true solution satisfies

$$\mathbf{u} = G\mathbf{u} + \mathbf{c}$$

Subtracting the second equation from the first equation gives

$$\mathbf{u}_n - \mathbf{u} = G(\mathbf{u}_{n-1} - \mathbf{u}), \qquad n = 1, 2, \ldots$$

If we define the error between the true solution and the iterative solution by

$$\mathbf{e}_n = \mathbf{u}_n - \mathbf{u}$$

we see that

$$\mathbf{e}_n = G\mathbf{e}_{n-1}, \qquad n = 1, 2, \ldots$$

We then have

$$\begin{aligned}\mathbf{e}_n &= G\mathbf{e}_{n-1}\\ &= G(G\mathbf{e}_{n-2})\\ &= G^2\mathbf{e}_{n-2}\\ &= G^3\mathbf{e}_{n-3}\\ &\vdots\\ &= G^n\mathbf{e}_0\end{aligned}$$

Note the dependence on powers of a matrix G

We can use the material from the lectures on eigenvalues and eigenvectors to reason about the error at step n as n increases

If the matrix G is diagonalisable, there exists a diagonal matrix D and a non-singular matrix S such that

$$G = S^{-1}DS$$

where the entries on the diagonal of D are the eigenvalues of G

We then have

$$G^n = S^{-1}D^nS$$

We then see that

$$\mathbf{e}_n = S^{-1}D^nS\mathbf{e}_0$$

The eigenvalues of G are clearly crucial for the convergence of Jacobi's method

Clearly, convergence of Jacobi's method requires that all eigenvalues of G have modulus less than 1

If this condition is satisfied then the largest entry in D^n will be of magnitude $|\lambda_{\max}|^n$, where $\lambda_{\max}$ is the eigenvalue with largest modulus

It is then clearly advantageous for $|\lambda_{\max}|$ to be as small as possible

This is true even if G is not diagonalisable and the eigenvalues are complex

In our first example using the Jacobi method we had

$$D = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \qquad L = \begin{pmatrix}0 & 0\\ -0.5 & 0\end{pmatrix}, \qquad U = \begin{pmatrix}0 & -0.5\\ 0 & 0\end{pmatrix}$$

We then have

$$G = D^{-1}(L+U) = \begin{pmatrix}0 & -0.5\\ -0.5 & 0\end{pmatrix}$$

The eigenvalues of G are ± 0.5, and so the Jacobi method will converge, as we observed

In our second example using the Jacobi method we had

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix}$$

We then have

$$G = D^{-1}(L+U) = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$$

The eigenvalues of G are ± 2, and so the Jacobi method will diverge, as we observed

The Gauss-Seidel method

We may also write the Gauss-Seidel method in the form used above for the Jacobi method

We write

$$\mathbf{u}_0 = \text{initial guess}$$
$$\mathbf{u}_n = G\mathbf{u}_{n-1} + \mathbf{c}, \qquad n = 1, 2, \ldots$$

where

$$G = (D-L)^{-1}U, \qquad \mathbf{c} = (D-L)^{-1}\mathbf{b}$$

The convergence of the Gauss-Seidel method then depends on the eigenvalues of the matrix G defined above

In our first example using the Gauss-Seidel method we had

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -0.5 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0 \end{pmatrix}$$

We then have

$$G = (D-L)^{-1}U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0.25 \end{pmatrix}$$

The eigenvalues of G are 0 and 0.25, and so the Gauss-Seidel method will converge, as we observed

As the eigenvalue with largest modulus is smaller than that seen for Jacobi's method, Gauss-Seidel will converge faster in this case, as observed

In our second example using the Gauss-Seidel method we had

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix}$$

We then have

$$G = (D-L)^{-1}U = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$$

The eigenvalues of G are 0 and 4, and so the Jacobi method will diverge, as we observed

The SOR method

We may also write the SOR method in the form used above

We write

$$\mathbf{u}_0 = \text{initial guess}$$
$$\mathbf{u}_n = G\mathbf{u}_{n-1} + \mathbf{c}, \qquad n = 1, 2, \ldots$$

where

$$G = (D - \omega L)^{-1}\left((1-\omega)D + \omega U\right), \qquad \mathbf{c} = (D - \omega L)^{-1}\omega\mathbf{b}$$

We now see that we can accelerate convergence of the SOR method by choosing ω so that the largest eigenvalue of G is as small as possible

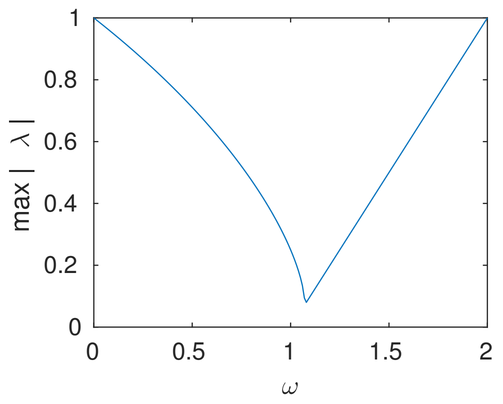
In example using the SOR method we had

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & 0 \\ -0.5 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -0.5 \\ 0 & 0 \end{pmatrix}$$

We then have

$$\begin{aligned} G &= (D - \omega L)^{-1}\left((1-\omega)D + \omega U\right) \\ &= \begin{pmatrix} 1 & 0 \\ 0.5\omega & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1-\omega & -0.5\omega \\ 0\omega & 1-\omega \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ -0.5\omega & 1 \end{pmatrix} \begin{pmatrix} 1-\omega & -0.5\omega \\ 0\omega & 1-\omega \end{pmatrix} \end{aligned}$$

The eigenvalues of G depend on the choice of ω



The rate of convergence is fastest when the maximum eigenvalue has smallest modulus

This agrees with our earlier results

The method of steepest descent

A symmetric matrix $A^\top = A$ is said to be positive definite if $\mathbf{x}^\top A\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$.

A symmetric matrix A is positive definite if and only if it has positive eigenvalues

Proof:

As A is symmetric, we have $A = P^\top DP$, where $P^\top P = \mathcal{I}$, and D is a diagonal matrix with the eigenvalues of A on the diagonal

We then have

$$\begin{aligned}
&\mathbf{x}^\top A\mathbf{x} > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}\\
\iff\ &\mathbf{x}^\top P^\top D P\mathbf{x} > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}\\
\iff\ &(P\mathbf{x})^\top D(P\mathbf{x}) > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}\\
\iff\ &\mathbf{y}^\top D\mathbf{y} > 0 \text{ for all } P^\top\mathbf{y} \neq \mathbf{0}\\
\iff\ &\mathbf{y}^\top D\mathbf{y} > 0 \text{ for all } \mathbf{y} \neq \mathbf{0}\\
\iff\ &\sum_{n=1}^{N} D_n y_n^2 > 0 \text{ for all } \mathbf{y} \neq \mathbf{0}\\
\iff\ &\text{entries of diagonal matrix } D \text{ are positive}\\
\iff\ &\text{eigenvalues of } A \text{ are positive}
\end{aligned}$$

Let A be a real, symmetric, positive definite matrix of size $N \times N$

Suppose we want to solve $A\mathbf{u} = \mathbf{b}$

Define $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top\mathbf{b}$

Then the solution of the linear system, $\mathbf{u}$, is then equal to the unique minimum of $F(\mathbf{x})$

Proof:

Suppose $\mathbf{x} \neq \mathbf{u}$. We then have

$$\begin{aligned}
F(\mathbf{x}) &= F(\mathbf{u}+\mathbf{y}) \qquad \text{for some } \mathbf{y} \neq \mathbf{0}\\
&= \frac{1}{2}(\mathbf{u}+\mathbf{y})^\top A(\mathbf{u}+\mathbf{y}) - (\mathbf{u}+\mathbf{y})^\top\mathbf{b}\\
&= F(\mathbf{u}) + \frac{1}{2}\mathbf{y}^\top A\mathbf{u} + \frac{1}{2}\mathbf{u}^\top A\mathbf{y} + \frac{1}{2}\mathbf{y}^\top A\mathbf{y} - \mathbf{y}^\top\mathbf{b}\\
&= F(\mathbf{u}) + \frac{1}{2}\mathbf{y}^\top\mathbf{b} + \frac{1}{2}\mathbf{b}^\top\mathbf{y} + \frac{1}{2}\mathbf{y}^\top A\mathbf{y} - \mathbf{y}^\top\mathbf{b}\\
&= F(\mathbf{u}) + \frac{1}{2}\mathbf{y}^\top A\mathbf{y}\\
&> F(\mathbf{u})
\end{aligned}$$

and so $F(\mathbf{u})$ must be the minimum of $F(\mathbf{x})$.

Further, if $F(\mathbf{u}) = F(\mathbf{x})$ then

$$\frac{1}{2}\mathbf{y}^\top A\mathbf{y} = \mathbf{0}$$

As A is positive definite this implies that $\mathbf{y} = \mathbf{0}$, and so $\mathbf{x} = \mathbf{u}$

We will use this observation to motivate the method of steepest descent

We have shown that solving $A\mathbf{u} = \mathbf{b}$ is equivalent to minimising $F(\mathbf{x})$

We may write

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} x_i A_{ij} x_j - \sum_{i=1}^{N} x_i b_i$$

We then have

$$\frac{\partial F}{x_k} = \sum_{i=1}^{N} A_{ki} x_i - b_k$$

and so the gradient is in the direction of $A\mathbf{x} - \mathbf{b}$

Don't worry if you don't understand this slide — it will be clearer after the Continuous Mathematics course next term

Suppose $\mathbf{u}_n$ is an approximation to the solution of $A\mathbf{u} = \mathbf{b}$

We know that we want to minimise $F(\mathbf{x})$

We will move in the opposite direction to the gradient (as we want to decrease $F(\mathbf{x})$), given by

$$\mathbf{r}_n = \mathbf{b} - A\mathbf{u}_n$$

We then seek α such that

$$F(\mathbf{u}_n + \alpha\mathbf{r}_n)$$

is minimised

$$\begin{aligned}
f(\alpha) &= F(\mathbf{u}_n + \alpha\mathbf{r}_n) \\
&= \frac{1}{2}(\mathbf{u}_n + \alpha\mathbf{r}_n)^\top A(\mathbf{u}_n + \alpha\mathbf{r}_n) - (\mathbf{u}_n + \alpha\mathbf{r}_n)^\top\mathbf{b} \\
&= F(\mathbf{u}_n) + \frac{1}{2}\alpha^2\mathbf{r}_n^\top A\mathbf{r}_n + \alpha\mathbf{r}_n^\top(A\mathbf{u}_n - \mathbf{b}) \\
&= F(\mathbf{u}_n) + \frac{1}{2}\alpha^2\mathbf{r}_n^\top A\mathbf{r}_n - \alpha\mathbf{r}_n^\top\mathbf{r}_n
\end{aligned}$$

At a minimum, $f'(\alpha) = 0$, and so we choose

$$\alpha = \frac{\mathbf{r}_n^\top\mathbf{r}_n}{\mathbf{r}_n^\top A\mathbf{r}_n}$$

Note that $f''(\alpha) = \mathbf{r}_n^\top A\mathbf{r}_n > 0$, and so this point is a minimum, as required

The steepest descent method uses the observation that if $\mathbf{u}_{n-1}$ is an approximation to the solution, then $\mathbf{u}_{n-1} + \alpha\mathbf{r}_{n-1}$ is a better approximation

We now define the method of steepest descent by the following algorithm

- Choose an initial guess $\mathbf{u}_0$
- For $n = 1, 2, \ldots$
  - $\mathbf{r}_{n-1} = \mathbf{b} - A\mathbf{u}_{n-1}$
  - $\mathbf{u}_n = \mathbf{u}_{n-1} + \frac{\mathbf{r}_{n-1}^\top\mathbf{r}_{n-1}}{\mathbf{r}_{n-1}^\top A\mathbf{r}_{n-1}}\mathbf{r}_{n-1}$

We terminate the algorithm, as for other iterative methods, when $\|A\mathbf{u}_n - \mathbf{b}\| < \epsilon$

Note that the steepest descent algorithm requires us to divide by $\mathbf{r}_{n-1}^\top A\mathbf{r}_{n-1}$

If $\mathbf{r}_{n-1} = \mathbf{0}$ this will be zero

This doesn't actually cause problems, as if $\mathbf{r}_{n-1} = \mathbf{0}$ then $A\mathbf{u}_{n-1} = \mathbf{b}$, and so $\mathbf{u}_{n-1}$ is the solution of the linear system

An example of the steepest descent method

We now apply the steepest descent method to the linear system

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

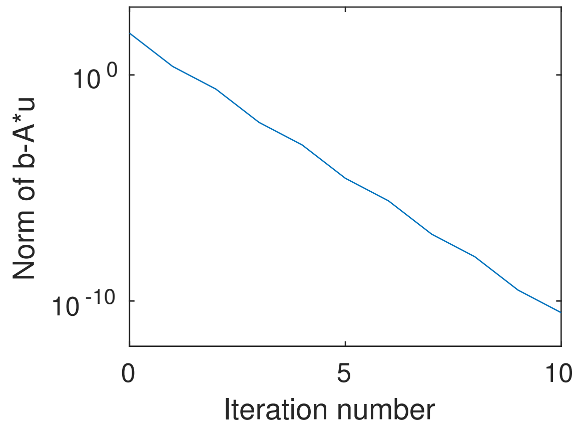
This linear system has solution $\mathbf{u} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

A is symmetric

The eigenvalues of A are positive

As A is symmetric and positive definite, we may use the steepest descent method

We begin with initial guess $\mathbf{u}_0 = \begin{pmatrix} 40 \\ 30 \end{pmatrix}$


Norm of b-A*u
10^{0}
10^{-10}
Iteration number