Relaxation Methods for Iterative Solution
to Linear Systems of Equations

Gerald Recktenwald
Portland State University
Mechanical Engineering Department
gerry@me.pdx.edu

Primary Topics

- Basic Concepts
- Stationary Methods a.k.a. Relaxation methods
  - Gauss Seidel
  - Jacobi
  - SOR
Direct and Iterative Methods

Solve

\[ Ax = b \]

where \( A \) is \( n \times n \) matrix, \( x \) and \( b \) are \( n \) element column vectors.

Gaussian elimination with backward substitution is a **Direct Method**

- A Direct Method obtains the solution in a finite number of steps
- Solution is *equivalent* to
  \[ x = A^{-1}b \]

**Iterative Methods** obtain a sequence of approximations to the solution

\[ x^k \quad k = 0, 1, 2, \ldots \]

such that \( (b - Ax^k) \to 0 \) as \( k \to \infty \).

Residual

The residual

\[ r = b - Ax \]

is zero when \( x \) is the solution to \( Ax = b \).

For an iterative method the residual at iteration \( k \) is

\[ r^k = b - Ax^k \quad k = 0, 1, 2, \ldots \]

If an iterative methods converges, then \( r^k \to 0 \) as \( k \to \infty \).
Relaxation Methods (1)

Overview
- Simple to program
- Converges slowly for large systems of equations \((\text{large } n)\)
- Not a useful \textit{stand alone} solution method
- Key ingredient to \textit{multigrid} methods

Examples
- Jacobi
- Gauss-Seidel
- SOR

Relaxation Solution to Linear Systems

Relaxation Methods (2)

Basic Idea
- Update elements of the solution vector one element at a time
- "Solve" a nodal equation by assuming other nodal values are known

Apply the finite volume method to get

\[-a_S \phi_S - a_W \phi_W + a_P \phi_P - a_E \phi_E - a_N \phi_N = b\]

If \(k\) is the iteration counter, we "solve" for \(\phi_P^{k+1}\)

\[\phi_P^{k+1} = \frac{1}{a_P} \left[ b + a_S \phi_S^k + a_W \phi_W^k + a_E \phi_E^k + a_N \phi_N^k \right]\]

Values of \(\phi_W, \phi_E, \phi_N\) and \(\phi_S\) are updated by applying the iterative formula to the cells centered at those neighbor nodes.
**Matrix Notation (1)**

On a *structured mesh* the *interior* nodes can be numbered sequentially with a mapping like

\[ i = I - 1 + (J - 2)n_x \]

where \( n_x \) is the number of cells in the \( x \) direction and \( x_i \) is the value of the \( i \)th dependent field variable which is located at the cells with indices \( I \) and \( J \).

The compass point notation

\[-a_S \phi_S - a_W \phi_W + a_P \phi_P - a_E \phi_E - a_N \phi_N = b\]

becomes

\[-a_{S,i} x_{i-n_x} - a_{W,i} x_{i-1} + a_{P,i} x_i - a_{E,i} x_{i+1} - a_{W,i} x_{i+n_x} = b_i\]

**Matrix Notation (2)**

The *natural ordering* of nodes on a two-dimensional structured mesh yields a coefficient matrix with five diagonals. Coefficients on the diagonals correspond to \( a_P \) coefficient of \( \phi_P \leftrightarrow x_i \) and the four neighbor coefficients \( a_S, a_W, a_E, \) and \( a_N \).

![Matrix Diagram](image)

\[
A = \begin{pmatrix}
  -a_x & -a_y & a_x & -a_y & \\
  -a_x & -a_y & a_x & -a_y & \\
  a_x & -a_y & -a_x & a_y & \\
  -a_x & a_y & -a_x & a_y & \\
  a_x & -a_y & a_x & -a_y & 
\end{pmatrix}
\]
Matrix Notation\(^{(3)}\)

In an unstructured mesh, the coefficient matrix is also sparse, but the non-zero elements of the coefficient matrix do not lie along diagonals.

Unstructured Mesh

![Unstructured Mesh](image)

Non-zeros in coefficient matrix

![Non-zeros in coefficient matrix](image)

To simplify the presentation, we’ll use a structured mesh.

Relaxation Methods

Use relaxation methods to solve

\[ Ax = b \]

Relaxation methods do not work for all \( A \), but they do work for standard finite-difference and finite-volume discretizations of the Poisson equation.

- Jacobi
- Gauss-Seidel
- SOR – Successive Over-relaxation
Jacobi Iteration (1)

A discrete model of the Poisson equation yields a system of equations that can be written \( Ax = b \). The equation for the \( i \)th nodal value is

\[
\sum_{j=1}^{n} a_{ij} x_j = b_i
\]

where \( n \) is the total number of nodes in the domain.

Extract the diagonal term from the sum

\[
a_{ii} x_i + \sum_{\substack{j=1\atop j \neq i}}^{n} a_{ij} x_j = b_i
\]

The preceding equation will not be satisfied until, in the limit as the number of iterations increased to \( \infty \).

Jacobi Iteration (2)

“Solve” for the next guess \( x_i \).

\[
x_i = \frac{1}{a_{ii}} \left( b - \sum_{\substack{j=1\atop j \neq i}}^{n} a_{ij} x_j \right)
\]

In terms of the compass-point notation for the two-dimensional finite-volume method, the Equation (\( \ast \)) is

\[
\phi_P^{(k+1)} = \frac{1}{a_P} \left[ b + a_S \phi_S^{(k)} + a_W \phi_W^{(k)} + a_E \phi_E^{(k)} + a_N \phi_N^{(k)} \right]
\]

Equation (\( \ast \)) is not really the solution for \( x_i \) until all the other \( x_j \) are known. In other words, the value of \( x_i \) from Equation (\( \ast \)) will be correct only when the system of equations \( Ax = b \) is solved.

If matrix \( A \) has favorable properties, the value of \( x_i \) from Equation (\( \ast \)) will be closer to the solution than the previous guess at \( x_i \).
Jacobi Iteration (3)

Let k be the iteration counter. The Jacobi iteration formula is

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

This formula implies that two copies of the x vector are maintained in memory: one for iteration k and one for iteration k + 1.

```matlab
for k=1:maxit
    % -- For nodes away from the boundaries
    for i=(nx+1):n-nx
        xnew(i) = (b(i) + as(i)*x(i-nx) + aw(i)*x(i-1) ... 
            + an(i)*x(i+nx) + ae(i)*x(i+1)) / ap(i);
    end
    x = xnew;
end
```

Jacobi Iteration (4)

The Jacobi iterative formula can be written as a matrix equation. Let \( D = \text{diag}(a_{11}, \ldots, a_{nn}) \) and \( B = D - A \) then

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

is equivalent to

\[ x^{(k+1)} = D^{-1} \left( b + B x^{(k)} \right) \tag{1} \]

where \( x^{(k+1)} \) and \( x^{(k)} \) are the solution vectors at iteration \( k + 1 \) and \( k \), respectively.
**Jacobi Iteration** \((5)\)

The matrix form of the iteration can be rewritten

\[ \mathbf{x}^{(k+1)} = H\mathbf{x}^{(k)} + \mathbf{d} \quad (2) \]

where

\[ H = D^{-1}B \]
\[ \mathbf{d} = D^{-1}\mathbf{b} \]

The matrix formulation is primarily useful in the analysis of iterative methods and in toy implementations in MATLAB. The eigenvalues of \(H\) determine the convergence rate of the iterations.

**Stopping Criteria** \((1)\)

Recall that the goal is to solve

\[ A\mathbf{x} = \mathbf{b} \]

where \(A\) is \(n \times n\) matrix, \(\mathbf{x}\) and \(\mathbf{b}\) are \(n\) element column vectors.

*Iterative Methods* obtain a sequence of approximations to the solution

\[ \mathbf{x}^k \quad k = 0, 1, 2, \ldots \]

such that \(\mathbf{b} - A\mathbf{x}^k \to 0\) as \(k \to \infty\).

The residual

\[ r = \mathbf{b} - A\mathbf{x} \]

is zero when \(\mathbf{x}\) is the solution to \(A\mathbf{x} = \mathbf{b}\).
Stopping Criteria \( (2) \)

For an iterative method the residual at iteration \( k \) is

\[ r^k = b - Ax^k \quad k = 0, 1, 2, \ldots \]

If an iterative methods converges, then \( r^k \to 0 \) as \( k \to \infty \).

Since \( r \) is a vector, what does it mean that \( r^k \to 0 \)?

Use a convenient vector norm to test whether \( \| r \| \) is small enough.

---

Jacobi Iteration \( (3) \)

Solution to a toy problem

\[ \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \]

on \( 0 \leq x \leq L, 0 \leq y \leq W \)

\( T = 0 \) on three boundaries

uniform \( q \) on fourth boundary

Solution is obtained by MATLAB code \texttt{demoJacobi}
**Jacobi Iteration** (4)

Convergence of Jacobi iterations slow as mesh is refined.

![Graph showing convergence of Jacobi iterations.](image)

**Gauss-Seidel Iteration** (1)

In the Jacobi method \( x^{(k+1)} \) is obtained from the frozen values of \( x^{(k)} \).

The Gauss-Seidel method uses a similar formula for updating \( x_{i}^{(k+1)} \), but the new value of \( x_{i}^{(k+1)} \) is used as soon as it is available.
Gauss-Seidel Iteration \((2)\)

Replace nodal values as sweep progresses through the domain

for \(k=1:\text{maxit}\)

\%
% -- For nodes away from the boundaries

for \(i=(n+1):n-nx\)

\(x(i) = \frac{(b(i) + a(i) \cdot x(i-nx) + a(i) \cdot x(i-1) + a(i) \cdot x(i+nx) + a(i) \cdot x(i+1))}{a(i)};\)

end

diag

Gauss-Seidel Iteration \((3)\)

Consider the sweep through the nodal values in order of increasing \(i\):

\[
x_1^{(k+1)} = \frac{1}{a_{11}} \left( b_1 - \sum_{j=2}^{n} a_{1j} x_j^{(k)} \right)
\]

\[
x_2^{(k+1)} = \frac{1}{a_{22}} \left( b_2 - a_{21} x_1^{(k+1)} - \sum_{j=3}^{n} a_{1j} x_j^{(k)} \right)
\]

\[
x_3^{(k+1)} = \frac{1}{a_{33}} \left( b_3 - a_{31} x_1^{(k+1)} - a_{32} x_2^{(k+1)} - \sum_{j=4}^{n} a_{1j} x_j^{(k)} \right)
\]

\[
\vdots
\]

\[
x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{1j} x_j^{(k)} \right)
\]
Gauss-Seidel Iteration \((4)\)

\[
A x = b
\]

Again, let

\[
D = \text{diag}(a_{11}, \ldots, a_{nn})
\]

and define

\[
L = \text{lower triangular part of } A
\]

\[
U = \text{upper triangular part of } A
\]

Then Gauss-Seidel iterations correspond to the splitting of \(A\)

\[
A = D - L - U
\]

Note that \(A \neq LU\), i.e., \(L\) and \(U\) are not the usual factors of \(A\).

Gauss-Seidel Iteration \((5)\)

Equation (3) can be rewritten as

\[
a_{ii} x_i^{(k+1)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_i^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}
\]

or

\[
a_{ii} x_i^{(k+1)} - \sum_{j=1}^{i-1} a_{ij} x_i^{(k+1)} = b_i - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}
\]

or

\[
D x^{(k+1)} - L x^{(k+1)} = b + U x^{(k)}
\]

\[
\implies (D - L) x^{(k+1)} = U x^{(k)} + b
\]

or

\[
x^{(k+1)} = (D - L)^{-1} (U x^{(k)} + b)
\]

\[
(4)
\]
Gauss-Seidel Iteration (6)

Note: The order in which the nodes are processed in the Gauss-Seidel iterations will affect the path toward the solution. In other words, for two different orderings of the nodes, the intermediate iterates of $x$ will be different.

The Jacobi iteration does not depend on the order in which the nodes are numbered.

Much of the classical theory of relaxation methods assumes that the nodes and numbered in natural order.

SOR: Successive Over-Relaxation (1)

Let $\tilde{x}_i^{(k+1)}$ be the updated value of $x_i$ from the Gauss-Seidel formula, viz.

$$\tilde{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

Instead of taking $\tilde{x}_i^{(k+1)}$ as the value of $x_i$ at the $k + 1$ step, use

$$x_i^{(k+1)} = \omega \tilde{x}_i^{(k+1)} + (1 - \omega) x_i^{(k)}$$

where $\omega$ is a scalar weighting factor. The basic step in Equation (5) is called relaxation.

$\omega = 1$ for Gauss-Seidel
$\omega < 1$ causes the iterations to more slowly move toward the solution. This is called under-relaxation.
$\omega > 1$ causes the iterations to accelerate, i.e. to be more aggressive. This is called over-relaxation.
Relaxation as an update

We can rewrite Equation (6) as

\[ x_i^{(k+1)} = x_i^{(k)} + \omega \left( x_i^{(k+1)} - x_i^{(k)} \right) \] (7)

or

\[ x_i^{(k+1)} = x_i^{(k)} + \omega \Delta x_i^{(k+1)} \] (8)

where

\[ \Delta x_i^{(k+1)} = x_i^{(k+1)} - x_i^{(k)} \]

\( \Delta x_i^{(k+1)} \) is the update to \( x_i \) in iteration \( k \).

\( \Delta x_i^{(k+1)} \) is the direction in which the solution is changing.

\( \omega \Delta x_i^{(k+1)} \) is a magnified (\( \omega > 1 \)) or reduced (\( \omega < 1 \)) change to the solution.

SOR: Successive Over-Relaxation (2)

Substitute Equation (6) into Equation (5)

\[ x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right) \] (9)

Multiply through by \( a_{ii} \) and rearrange to get

\[ a_{ii} x_i^{(k+1)} + \omega \sum_{j<i} a_{ij} x_j^{(k+1)} = (1 - \omega) a_{ii} x_i^{k} - \omega \sum_{j>i} a_{ij} x_j^{(k)} + \omega b_i \]

Use the splitting \( A = D - L - U \) and the preceding equation becomes

\[ D x^{(k+1)} - \omega L x^{(k+1)} = (1 - \omega) D x^{(k)} + \omega U x^{(k)} + \omega b \]

\[ (D - \omega L) x^{(k+1)} = \left[ (1 - \omega) D x^{(k)} + \omega U \right] x^{(k)} + \omega b \]

\[ x^{(k+1)} = (D - \omega L)^{-1} \left[ (1 - \omega) D x^{(k)} + \omega U \right] x^{(k)} + \omega (D - \omega L)^{-1} b \]
**SOR: Successive Over-Relaxation** (3)

The update equation is

\[
x^{(k+1)} = (D - \omega L)^{-1} \left[ (1 - \omega) Dx^{(k)} + \omega U \right] x^{(k)} + \omega (D - \omega L)^{-1} b
\]

Thus

\[
x^{(k+1)} = Hx^{(k)} + d
\]

where

\[
H = (D - \omega L)^{-1} \left[ (1 - \omega) Dx^{(k)} + \omega U \right]
\]

\[
d = \omega (D - \omega L)^{-1} b
\]

The speed of convergence depends on the eigenvalues of \( H \).

---

**Compare Relaxation Convergence**

On a coarse 8 × 8 mesh

![Graph comparing relaxation methods](image-url)
Compare Relaxation Convergence

On a coarse $16 \times 16$ mesh