Iterative Solutions of Linear Systems

Algorithms 1

Sachin Shanbhag

Department of Scientific Computing
Florida State University,
Tallahassee, FL 32306.

Why Iterative Methods?

For a linear system given by:

$$\mathbf{A}_{n\times n}\mathbf{x}_{n\times 1} = \mathbf{b}_{n\times 1}$$

- When $n \sim 10^6$ or greater, then the $\mathcal{O}(n^3)$ methods can become expensive
- When solving PDEs, a 3D grid can easily have $100 \times 100 \times 100 = 10^6$ grid points
- When $\mathbf{A}$ is sparse, iterative methods are particularly useful.*
- Start with an initial guess $\mathbf{x}^{(0)}$, and refine it until $||\mathbf{b} - \mathbf{A}\mathbf{x}||$ and/or $||\mathbf{x}^k - \mathbf{x}^{k-1}||$ are sufficiently small.

---

*How many computational scientists does it take to screw in a light-bulb? Answer: 0.9997, after 3 iterations.

Jacobi Method

Partition the matrix $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, where, $\mathbf{L}$ and $\mathbf{U}$ are strictly lower and upper-triangular, and $\mathbf{D}$ is a diagonal matrix.

$$\mathbf{D} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Jacobi Method

We have:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ (\mathbf{L}+\mathbf{D}+\mathbf{U})\mathbf{x} &= \mathbf{b} \\ \mathbf{D}\mathbf{x} &= \mathbf{b}-(\mathbf{L}+\mathbf{U})\mathbf{x} \\ \mathbf{x} &= \mathbf{D}^{-1}(\mathbf{b}-(\mathbf{L}+\mathbf{U})\mathbf{x}). \end{aligned}$$

Hence, consider the iterative update:

$$\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}\left(\mathbf{b}-(\mathbf{L}+\mathbf{U})\mathbf{x}^{(k)}\right) \tag{1}$$

A diagonal matrix is easy to "invert", as long as diagonal entries are non-zero (may have to permute the matrix)

Jacobi Method

$$\mathbf{D} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \implies \mathbf{D}^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix}$$

Eqn 1 is equivalent to the scalar form:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k)} \right) \tag{2}$$

Can be easily parallelized, because elements of $\mathbf{x}^{(k+1)}$ do not depend on each other.

Needs duplicate storage for $\mathbf{x}$, ($\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$)

Exercise: Show using eqn. 2 that each iteration is $\mathcal{O}(n^2)$.

Gauss-Seidel

There are other ways to write a fixed-point iteration

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ (\mathbf{L}+\mathbf{D}+\mathbf{U})\mathbf{x} &= \mathbf{b} \\ (\mathbf{L}+\mathbf{D})\mathbf{x} &= \mathbf{b}-\mathbf{U}\mathbf{x} \end{aligned}$$

GS considers the iterative update (forward substitution):

$$\mathbf{x}^{(k+1)} = (\mathbf{L}+\mathbf{D})^{-1}\left(\mathbf{b}-\mathbf{U}\mathbf{x}^{(k)}\right) \tag{3}$$

It's scalar form corresponds to:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \tag{4}$$

Gauss-Seidel

- Needs non-zero diagonal elements like the Jacobi method
- Solve a lower triangular system by forward substitution ($\mathcal{O}(n^2)$), instead of taking the inverse
- Compare equations 2 and 4. The only difference is that we use new values $(k+1)$ as they become available.
- Need to store only one copy of $\mathbf{x}$, since values can be overwritten
- In practice, it converges about twice as fast as Jacobi method.
- Let us look at an example, and then consider the convergence properties more carefully.

Example

Solve:

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$$

The true solution is $\mathbf{x} = \begin{bmatrix} 2 & -2 \end{bmatrix}^T$. Let's try the Jacobi method with $\mathbf{x}^{(0)} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$

Jacobi Solution:

i	x_1	x_2	$\|\|\mathbf{x}^{(i)} - \mathbf{x}\|\|_2$
0	1.000000	1.000000	3.162300
1	0.000000	-1.666667	2.027588
2	1.777778	-1.333333	0.702728
3	1.555556	-1.925926	0.450575
5	1.901235	-1.983539	0.100128
10	1.999458	-1.998374	0.001714
20	2.000000	-1.999999	0.000001

Example

Gauss-Seidel Solution:

i	x_1	x_2	$\|\|\mathbf{x}^{(i)} - \mathbf{x}\|\|_2$
0	1.000000	1.000000	3.162300
1	0.000000	-1.333333	2.108185
2	1.555556	-1.851852	0.468486
3	1.901235	-1.967078	0.104108
5	1.995123	-1.998374	0.005141
10	1.999997	-1.999999	0.000003
11	1.999999	-2.000000	0.000001

Notice that we converged about twice as fast.

Convergence

- The Jacobi method does not always converge, but if the matrix is strictly diagonally dominant it is guaranteed to converge.

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

- Sometimes, it will converge even if $\mathbf{A}$ is not strictly diagonally dominant.
- Conditions for convergence of Gauss-Seidel are weaker than that of Jacobi method
- Convergence is guaranteed for symmetric positive-definite matrices and strictly diagonally dominant matrices. The method may converge even when these conditions are not met.

Successive Relaxation

This method can be understood as a generalization of Gauss-Seidel.

The first intermediate in successive relaxation (SR) is a Gauss-Seidel (GS) step:

$$\hat{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \qquad (5)$$

for $i = 1, 2, ..., n$.

This is followed by the update:

$$x_i^{(k+1)} = x_i^{(k)} + \omega(\hat{x}_i^{(k+1)} - x_i^{(k)}) \qquad (6)$$

If $\omega = 1$ this is equivalent to Gauss-Seidel method

Successive Relaxation

With some rearrangement, it can be shown that eqns 5 and 6 can be combined into matrix form as:

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{x}^{(k+1)} = \omega\mathbf{b} - [\omega\mathbf{U} + (\omega - 1)\mathbf{D}]\mathbf{x}^{(k)} \qquad (7)$$

$0 < \omega < 2$ is a relaxation parameter, chosen to accelerate convergence

When $\omega > 1$, we have over-relaxation, and when $\omega < 1$, we have under-relaxation.

Choosing optimal ω is nontrivial, but convergence can be accelerated by an order of magnitude over Gauss-Seidel.

While the matrix form is useful to see SR fitting the form $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{c}$, in actual calculations, eqns 5 and 6 are used.

Epilogue

- There are other iterative so-called "Krylov subspace" methods that are popular including conjugate-gradient (CG), generalized minimal residual method (GMRES), biconjugate gradient method (BiCG) etc.
- *Preconditioning* of the matrix $\mathbf{A}$ can accelerate convergence.
- Direct methods do not need initial guesses, but cannot take advantage if one is available.
- Iterative methods usually require less work, if convergence is rapid (eg. strong diagonal dominance). Many matrix equations that arise from PDEs satisfy this criteria.
- Iterative methods usually require less storage.