Successive Overrelaxation

Tung and Tung
Menlo School

May 2, 2021

Gaussian elimination

- Consider the matrix equation $A\vec{x} = \vec{b}$, where
  - A is an $n \times n$ matrix
  - $\vec{x}$ and $\vec{b}$ are n-component column vectors
- To solve for $\vec{x}$, we can use Gaussian elimination to write $[A|\vec{b}]$ in row echelon form.
- But this process is computationally intensive.
  - $n(n+1)/2$ divisions
  - $(2n^3 + 3n^2 - 5n)/6$ multiplications
  - $(2n^3 + 3n^2 - 5n)/6$ subtractions
  - $2n^3/3$ operations
  - $O(n^3)$ arithmetic complexity
- Gaussian elimination can be used on a computer for solving systems with thousands of equations and unknowns. But it can't be used when there are millions of equations and unknowns, due to computing power limitations.
- Instead, these systems are often solved using **iterative methods**.

Iterative methods

- An **iterative method** uses an initial value to generate a sequence of improving approximate solutions $\{\vec{x}^k\}$ for k nonnegative. Each approximation is derived from the previous ones.
- An iterative method converges if the spectral radius ρ of the iteration matrix is less than 1.
  - The spectral radius of a square matrix is the largest absolute value of its eigenvalues.

Stationary iterative methods

- Split the matrix A into $M + N$, where M is easily invertible.
- Write $M\vec{x}^{k+1} = -N\vec{x}^{k} + \vec{b}$.
- Solving for $\vec{x}^{k+1}$ yields $\vec{x}^{k+1} = -M^{-1}N\vec{x}^{k} + M^{-1}\vec{b}$.
- In many algorithms, A is split into terms involving lower and upper triangular matrices because these are easy to invert.
  - A lower triangular matrix can be inverted via forward substitution.
  - An upper triangular matrix can be inverted via back substitution.
- The approximate solutions don't always converge to the true solution.
- The three methods we will discuss are all stationary iterative methods.

The Jacobi method

- To run the algorithm, first guess some $\vec{x}_0$. The better the guess, the faster the convergence.
- Write $A = D + L + U$, where D is diagonal, L is strictly lower triangular, and U is strictly upper triangular.
- To find the $k+1$st approximation to $\vec{x}$, compute $\vec{x}^{k+1} = D^{-1}\left(\vec{b} - (L+U)\vec{x}^k\right)$.
- Indexing the elements of A, $\vec{x}$, and $\vec{b}$, we can index the components of $\vec{x}^{k+1}$:

$$\vec{x}_i^{k+1} = \frac{1}{a_{ii}}\left(\vec{b}_i - \sum_{j \neq i} a_{ij}\vec{x}_j^k\right), \quad i = 1, 2, \ldots, n.$$

- Jacobi converges too slowly to be used in practice, so better methods are needed.

The Gauss-Seidel method

- Compared to Jacobi, this method has a faster rate of convergence.
- To run the algorithm, first guess some $\vec{x_0}$. The better the guess, the faster the convergence.
- Write $A = L_* + U$, where L_* is lower triangular and U is strictly upper triangular (diagonal terms of U are 0).
- To find the $k+1$st approximation to $\vec{x}$, compute $\vec{x}^{k+1} = L_*^{-1}(\vec{b} - U\vec{x}^k)$
- Indexing the elements of A, $\vec{x}$, and $\vec{b}$, we can index the components of $\vec{x}^{k+1}$:

$$\vec{x}_i^{k+1} = \frac{1}{a_{ii}} \left(\vec{b}_i - \sum_{j=1}^{i-1} a_{ij} \vec{x}_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} \vec{x}_j^k \right), \quad i = 1, 2, \ldots, n.$$

Successive overrelaxation (SOR)

- SOR is a generalization of Gauss-Seidel that speeds up rate of convergence.
- SOR takes a weighted average between the previous iterate and the Gauss-Seidel iterate successively for each component.
- To run the algorithm, first guess some $\vec{x}_0$. The better the guess, the faster the convergence. In addition, select a relaxation factor $\omega \in (0, 2)$. A method for choosing the optimal ω is discussed later.
- If $\omega = 1$, SOR is the same as Gauss-Seidel.
- Write $A = D + L + U$, where D is diagonal, L is strictly lower triangular, and U is strictly upper triangular.
- To find the $k+1$st approximation to $\vec{x}$, compute $\vec{x}^{k+1} = (D + \omega L)^{-1}\left(\omega\vec{b} - [\omega U + (\omega - 1)D]\vec{x}^k\right)$.
- Indexing the elements of A, $\vec{x}$, and $\vec{b}$, we can index the components of $\vec{x}^{k+1}$:

$$\vec{x}_i^{k+1} = (1-\omega)\vec{x}_i^k + \frac{\omega}{a_{ii}}\left(\vec{b}_i - \sum_{j<i} a_{ij}\vec{x}_j^{k+1} - \sum_{j>i} a_{ij}\vec{x}_j^k\right), \quad i = 1, 2, \ldots, n.$$

The relaxation factor and convergence rate

- The convergence rate for SOR can be analytically derived when these assumptions are satisfied:
  - Jacobi iteration matrix $C_{\text{Jac}} := I - D^{-1}A$ has only real eigenvalues
  - The Jacobi method is convergent: $\mu := \rho(C_{\text{Jac}}) < 1$
  - A unique solution exists: $\det(A) \neq 0$
- Convergence rate:

$$\rho(C_\omega) = \begin{cases} \frac{1}{4}\left(\omega\mu + \sqrt{\omega^2\mu^2 - 4(\omega-1)}\right)^2, & 0 < \omega \leq \omega_{\text{opt}} \\ \omega - 1, & \omega_{\text{opt}} < \omega < 2 \end{cases}$$

- where the optimal relaxation parameter is given by

$$\omega_{\text{opt}} := 1 + \left(\frac{\mu}{1 + \sqrt{1-\mu^2}}\right)^2$$

Sources

- https://en.wikipedia.org/wiki/Gaussian_elimination
- https://en.wikipedia.org/wiki/Iterative_method
- https://en.wikipedia.org/wiki/Jacobi_method
- https://en.wikipedia.org/wiki/Gauss-Seidel_method
- https://en.wikipedia.org/wiki/Successive_over-relaxation
- https://mathworld.wolfram.com/SuccessiveOverrelaxationMethod.html
- Strauss, *Partial Differential Equations: An Introduction*, 2nd ed.