Iterative Solvers
Gauss-Seidel
Ch. 11

Gauss-Seidel

• What is Gauss-Seidel?
  – Alternative to direct methods (like Gauss-Elim.)
  – Iterative approach
  – similar to the idea of successive substitution for root finding

• Why do we want to use it?
  – Works well for large numbers of equations
  – Error is controlled by the number of iterations (Round off error is not usually as big a concern as Gauss-Elim)
  – Handles sparse matrices and large matrices well because it doesn’t have to store all of the zero’s.

Gauss-Seidel

• Approach, Consider
  \[
  \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
  \end{bmatrix}
  \begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3
  \end{bmatrix}
  =
  \begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3
  \end{bmatrix}
  \]

• Solve equation (1) for \( x_1 \), (2) for \( x_2 \) and (3) for \( x_3 \)

  \[
  x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}
  \]

  \[
  x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}}
  \]

  \[
  x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}
  \]
Gauss-Seidel - Method

- Start iteration process by guessing $x^2$ and $x^3$, and always using the most recent values of $x$’s:
  
  \[
  x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}} \\
  x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}} \\
  x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}
  \]

- Repeat using the new $x$’s

- Check for convergence:
  
  \[\epsilon_{ij} = \left| \frac{x^k - x^{k-1}}{x^k} \right| \times 100\%
  \]

  For all $i$’s where $k$ = current iteration, $k-1$ = previous iterations

Gauss-Seidel – Convergence criteria

- Diagonal Dominance – the diagonal element of a row should be greater than the sum of all other row elements:
  
  \[|x_{ii}| > \sum_{j \neq i} |x_{ij}|\]

- Is this matrix diagonally dominant?

- Sufficient but not necessary (i.e., if the condition is satisfied, convergence is guaranteed; if the condition is NOT satisfied, convergence still may occur)

Gauss-Seidel – Relaxation for iterative methods

- Motivation – speed up convergence – assuming we know the direction of the solution:
  
  \[\frac{x^{k+1}}{x^k} = \frac{x^{k+1} - x^k}{x^k} + \lambda (x^k - x^k) = Ax^k + (1 - \lambda)x^k\]

- Linear Extrapolation:
  
  \[x^{k+1} = x^k + \lambda (x^k - x^k)\]
Gauss-Seidel – Relaxation for iterative methods

• Motivation – speed up convergence – assuming we know the direction of the solution

\[ x^{k+1} = \frac{1}{\lambda} (k + 1) \text{ extrapolated solution} \]

\[ k \]

\[ k \]

\[ \lambda \]

\[ \text{New solution (present value)} \]

\[ \text{iterations (} k \text{)} \]

• Linear Extrapolation

\[ x_k = \frac{x^{k+1} - x^k}{\lambda} \]

\[ x^{k+1} = x^k + \lambda (x^{k+1} - x^k) \]

\[ \lambda \text{ relaxation coefficient} \]

\[ x^{k+1} = \lambda x^* + (1 - \lambda) x^k \]

Gauss-Seidel – Relaxation for iterative methods

• What is \( x^* \) ?

\[ x_k^* = \frac{b_i - a_{i1} x_{k+1}^* - a_{i2} x_i}{a_{ii}} \]

• Relaxation Coefficient – \( \lambda \)
  - Typical Values: \( 0 < \lambda < 2 \)
  - No relaxation: \( \lambda = 1 \)
  - Underrelaxation: \( 0 < \lambda < 1 \)
    - Provides a weighted average of current & previous results
    - Used to make non-convergent systems converge
    - Helps speed up convergence by damping oscillations
  - Overrelaxation: \( 1 < \lambda < 2 \)
    - Extra emphasis placed on present value
    - Assumes the solution is proceeding to the desired result, just too slowly
    - Will speed up convergence of a system that is already convergent
    - Selection of \( \lambda \) is problem specific and can require trial and error

Gauss-Seidel – Pseudocode

• 1st Loop: Row Normalization (divide all terms in the equation by diagonal term)

• 2nd Loop: Rearrange Equations:
  - Note \( a_{ii} = 1 \) after normalization

• Error checking flag – set to 1 at the beginning of each loop
  - Change to zero if any \( x_i > \epsilon_i \)
  - No need to check any \( x_i \)'s after that – saves computations

Show Matlab Gauss-Seidel Example
Other Iterative Solvers and GS variants

- Jacobi method – GS always uses the newest value of the variable x, Jacobi uses old values throughout the entire iteration
- Iterative Solvers are regularly used to solve Poisson’s equation in 2 and 3D using finite difference/element/volume discretizations:

\[
\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = f(x, y, z)
\]

- Red Black Gauss Seidel
- Multigrid Methods

Engineering Application

- Determine the force in each member of the truss and the reaction forces