LECTURE 19

ITERATIVE SOLUTIONS TO LINEAR ALGEBRAIC EQUATIONS

- As finer discretizations are being applied with Finite Difference and Finite Element codes:
  - Matrices are becoming increasingly larger
  - Density of matrices is becoming increasingly smaller
- ***Banded storage direct solution algorithms no longer remain attractive as solvers for very large systems of simultaneous equations***

Example

- For a typical Finite Difference or Finite Element code, the resulting algebraic equations have between 5 and 10 nonzero entries per matrix row (i.e. per algebraic equation associated with each node)

$$\mathbf{A} = \begin{bmatrix}
\gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\sigma & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 & 0 & 0 & 0 \\
\sigma & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 & 0 & 0 \\
\sigma & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 & 0 \\
\alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 & 0 \\
0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 & 0 \\
0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon & 0 \\
0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & 0 & \varepsilon \\
0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & 0 & \tau \\
0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & 0 & \tau \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta & \tau \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma & \delta \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \beta & \gamma
\end{bmatrix}$$

- Banded compact matrix density
  - ***Storage*** required for ***banded compact storage mode*** equals NM where N = size of the matrix, and M = full bandwidth
  - ***Total nonzero entries in the matrix*** assuming (a typical estimate of) 5 non-zero entries per matrix row = $5N$
  - Banded compact matrix density = the ratio of actual nonzero entries to entries stored in banded compact mode

$$\text{Banded compact matrix density} = \frac{Actual\ nonzero\ entries}{Banded\ storage} = \frac{5N}{NM} = \frac{5}{M}$$

N	M	Compact Matrix Density
100	20	0.25
10,000	200	0.025
10^6	2,000	0.0025
25×10^6	10,000	0.0005

- Thus with the increasing size of problems/applications and the decreasing matrix densities, iterative methods are becoming increasingly popular/better alternatives!

(Point) Jacobi Method - An Iterative Method

- Let's consider the following set of algebraic equations

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

- Guess a set of values for $\boldsymbol{X} \rightarrow \boldsymbol{X}^{[0]}$
- Now solve each equation for unknowns which correspond to the diagonal terms in $\mathbf{A}$, using guessed values for all other unknowns:

$$x_1^{[1]} = \frac{b_1 - a_{12}x_2^{[0]} - a_{13}x_3^{[0]}}{a_{11}}$$

$$x_2^{[1]} = \frac{b_2 - a_{21}x_1^{[0]} - a_{23}x_3^{[0]}}{a_{22}}$$

$$x_3^{[1]} = \frac{b_3 - a_{31}x_1^{[0]} - a_{32}x_2^{[0]}}{a_{33}}$$

- Arrive at a second estimate $\rightarrow \boldsymbol{X}^{[1]}$
- Continue procedure until you reach convergence (by comparing results of 2 consecutive iterations)
- This method is referred to as the ***(Point) Jacobi Method***
- The ***(Point) Jacobi Method*** is formally described in vector notation as follows:
  - Define **A** as

$$\mathbf{A} = \mathbf{D} - \mathbf{C}$$

    - Such that all diagonal elements of **A** are put into **D**
    - Such that all off-diagonal elements of **A** are put into – **C**
  - The scheme is now defined as:

$$\mathbf{D}\boldsymbol{X}^{[k+1]} = \mathbf{C}\boldsymbol{X}^{[k]} + \boldsymbol{B} \qquad k \geq 0 \quad \Rightarrow$$

$$\boldsymbol{X}^{[k+1]} = \mathbf{D}^{-1}\mathbf{C}\boldsymbol{X}^{[k]} + \mathbf{D}^{-1}\boldsymbol{B} \qquad k \geq 0$$

    - Recall that inversion of a diagonal matrix (to find $\mathbf{D}^{-1}$) is obtained simply by taking the reciprocal of each diagonal term

- The ***(Point) Jacobi Method*** method can be described in index notation as:

$$x_i^{[k+1]} = -\sum_{i=1,\, i\neq i}^{N} \frac{(a_{ij})}{(a_{ii})} x_j^{[k]} + \frac{b_i}{a_{ii}} \qquad 1 \le i \le N, \quad k \ge 0$$

- Advantage of iterative methods:
  - Each cycle $O(N^2)$ operations for full storage mode
  - Therefore roundoff error only accrues during $O(N^2)$ operations! This is much better than direct methods in which $O(N^3)$ operations accrue much more error!
  - Since each cycle only produces an approximation for the next cycle, any error in a guess will be handled by the next cycle
  - We can consider roundoff error to accrue only during the ***last*** iteration
  - Algorithm can be readily implemented to operate only on non-zero entries in the matrix reducing both storage and computations dramatically when matrix density is low

- Total number of operations for full storage mode

$$O(N^2K) \text{ where } K = \text{number of cycles required for convergence}$$

  - Note that you don't a priori know the number of cycles, K, required to achieve a certain degree of convergence and therefore accuracy

- Total number of operations for sparse non-zero entry only storage mode

$$O(N\alpha K) \text{ where } \alpha = \text{number of non zero entries per equation}$$

$$K = \text{number of cycles required for convergence}$$

  - The operation count dramatically reduces for sparse storage modes and is only a function of the number of non-zero entries and the number of cycles. Note that α is not related to the size of the problem, N, but to the local grid structure and algorithm

- Iterative methods are ideally suited for
  - Very large matrices since they reduce the roundoff problem
  - Sparse but not banded matrices since they can reduce computational effort by not operating on zeroes
  - Very large sparse banded matrices due to efficiency

Example

- Solve by point Jacobi method:

$$\begin{cases} 5x + y = 10 \\ 2x + 3y = 4 \end{cases} \Rightarrow$$

$$\begin{cases} 5x^{[k+1]} = 10 - y^{[k]} \\ 3y^{[k+1]} = 4 - 2x^{[k]} \end{cases} \Rightarrow$$

$$\begin{cases} x^{[k+1]} = 2 - \frac{1}{5} y^{[k]} \\ y^{[k+1]} = \frac{4}{3} - \frac{2}{3} x^{[k]} \end{cases}$$

- Start with solution guess $x^{[0]} = -1$, $y^{[0]} = -1$ and start iterating on the solution

k	$x^{[k]}$	$y^{[k]}$
0	-1	-1
1	2.20000	2.00000
2	1.60000	-0.13333
3	2.026666	0.26666
4	1.94666	-0.01777
5	2.00355	0.03555
:	:	:

- This is a converging process $\rightarrow$ keep on going until the desired level of accuracy is achieved

k	$x^{[k]}$	$y^{[k]}$
:	:	:
	2.00000	0.00000

Iterative convergence

- Is the $(k+1)^{th}$ solution better than the k^{th} solution?
  - Iterative process can be convergent/divergent
- A ***necessary*** conditions for convergence is that the set be ***diagonal***.
  - This requires that one of the coefficients in each of the equations be greater than all others and that this "strong coefficient" be contained in a different position in each equation.
  - We can re-arrange all strong elements onto diagonal positions by switching columns $\rightarrow$ this now makes the matrix ***diagonal.***
- A ***sufficient*** condition to ensure convergence is that the matrix is ***diagonally dominant***

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{N} |a_{ij}|, \qquad i = 1, N$$

- There are less stringent conditions for convergence

- A poor first guess will prolong the iterative process but will not make it diverge if the matrix is such that convergence is assured.
  - Therefore better guesses will speed up the iterative process

Criteria for ascertaining convergence

- ***Absolute*** convergence criteria

$$\left| x_i^{[k+1]} - x_i^{[k]} \right| \leq \varepsilon \quad \text{for} \quad i = 1, N$$

  - Where $\varepsilon \equiv$ a user specified tolerance or accuracy
  - The absolute convergence criteria is best used if you have a good idea of the magnitude of the x_i's

- ***Relative*** convergence criteria

$$\left| \frac{x_i^{[k+1]} - x_i^{[k]}}{x_i^{[k]}} \right| \leq \varepsilon$$

  - This criteria is best used if the magnitude of the x_i's are not known.
  - There are also problems with this criteria if $x_i \cong 0$

(Point) Gauss Seidel Method

- This method is very similar to the ***Jacobi*** method except that ***Gauss Seidel*** uses the most recently computed values for $\boldsymbol{X}$ in its computations.
- Using all updated values of $\boldsymbol{X}$ increases the convergence rate (twice as fast as ***Jacobi***)
- Consider the system

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

- Solve for the unknowns associated with diagonal terms as follows

$$x_1^{[k+1]} = \frac{b_1 - a_{12}x_2^{[k]} - a_{13}x_3^{[k]}}{a_{11}}$$

$$x_2^{[k+1]} = \frac{b_2 - a_{21}x_1^{[k+1]} - a_{23}x_3^{[k]}}{a_{22}}$$

$$x_3^{[k+1]} = \frac{b_3 - a_{31}x_1^{[k+1]} - a_{32}x_2^{[k+1]}}{a_{33}}$$

- The ***Gauss Seidel*** method is formally described in vector form as
  - Define **A** as

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$$

    - Put diagonal elements of **A** into **D**
    - Put negative of elements of **A** below the diagonal into **L**
    - Put negative of elements of **A** above the diagonal into **U**
  - Scheme is then defined as:

$$\mathbf{D}\boldsymbol{X}^{[k+1]} = \mathbf{L}\boldsymbol{X}^{[k+1]} + \mathbf{U}\boldsymbol{X}^{[k]} + \mathbf{B} \ , \qquad k \geq 0 \qquad \Rightarrow$$

$$\boldsymbol{X}^{[k+1]} = \mathbf{D}^{-1}\mathbf{L}\boldsymbol{X}^{[k+1]} + \mathbf{D}^{-1}\mathbf{U}\boldsymbol{X}^{[k]} + \mathbf{D}^{-1}\boldsymbol{B} \ , \qquad k \geq 0$$

- The ***Gauss Seidel*** method is formally described using index notation as

$$x_i^{[k+1]} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{[k+1]} - \sum_{j=i+1}^{N} \frac{a_{ij}}{a_{ii}} x_j^{[k]} + \frac{b_i}{a_{ii}}, \qquad 1 \leq i \leq N, \qquad k \geq 0$$

Point Relaxation Methods (Successive/Systematic (Over) Relaxation - SOR)

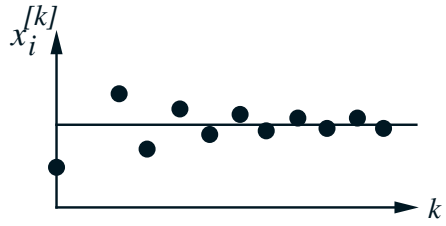
- The ***SOR*** approach improves the calculated values at the $k+1^{th}$ iteration using ***Gauss-Seidel*** by calculating a weighted average of the k^{th} and $k+1^{th}$ iterations and using this for the next iteration

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

  - Where $x_i^{[k+1]*}$ is the value obtained from the current ***Gauss-Seidel*** iteration
  - λ is the relaxation factor which must be specified

- Ranges of λ values
  - λ ranges between $0 < \lambda < 2$
    - $\lambda = 1 \quad \rightarrow$ ***Gauss-Seidel***
    - $0 < \lambda < 1 \quad \rightarrow$ ***Under-relaxation***
    - $1 < \lambda < 2 \quad \rightarrow$ ***Over-relaxation***

- ***Under-relaxation*** $\rightarrow$ $0 < \lambda < 1$
  - The current value is a weighted average of current ***Gauss-Seidel*** value and the value from the previous iteration
  - Typically used to make a non-convergent process converge
  - Can also be useful in speeding up convergence when the solutions oscillate about the converged solution



- ***Over-relaxation*** $\rightarrow$ $1 < \lambda < 2$
  - The current value is extrapolated beyond the ***Gauss-Seidel*** value
  - Typically used to accelerate an already convergent process
  - For $\lambda > 2$, the process diverges

- For a diagonally dominant matrix, ***SOR*** will always converge for $0 < \lambda < 2$

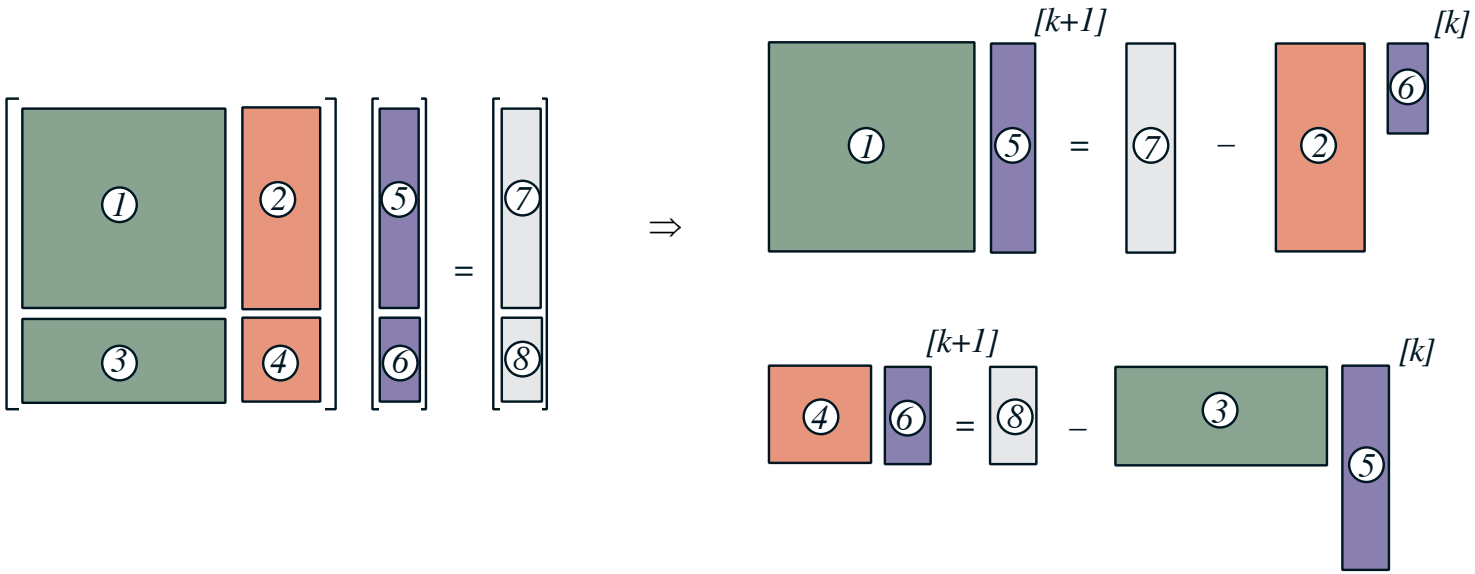
- Selection of an optimal λ value is quite complex
  - Depends on the characteristics of the matrix
  - Certain "classes" of problems will have optimal ranges
  - Trial and error is very useful
  - We can apply different values of λ for different blocks within a matrix which exhibit significantly different characteristics (different blocks in matrix may be associated with different p.d.e.'s in a coupled system)

Application of Gauss-Seidel to Non-Linear Equations

- ***Gauss-Seidel (with relaxation)*** is a very popular method to solve for systems of nonlinear equations
- Notes:
  - Multiple solutions exist for nonlinear equations
  - There ***must*** be linear components included in the equations such that a diagonal is formed
- No general theory on iterative convergence is available for nonlinear equations

Block Iterative Methods

- Instead of operating on a point by point basis, we solve simultaneously for entire groups of unknowns using direct methods
- Partition the coefficient matrix into blocks. All elements in the block are then solved in one step using a direct method



Direct/Iterative Methods

- Can correct errors due to roundoff in direct solutions by applying an iterative solution after the direct solution has been implemented.