Solving Systems of Linear Algebraic Equations

A systematic approach to stuff you've done before

Read Chapters 9 and 10

Motivation

- We have already encountered a couple situations in which we have needed to solve a system of equations:
  - Calculating the remaining terms of an Eigenvector:

$$\begin{bmatrix} a_{22}-\lambda & a_{23} & a_{24} \\ a_{32} & a_{33}-\lambda & a_{34} \\ a_{42} & a_{43} & a_{44}-\lambda \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \\ v_4 \end{bmatrix} = -\begin{bmatrix} a_{21} \\ a_{31} \\ a_{41} \end{bmatrix}$$

  - Calculating a Householder Matrix for matrix deflation

Motivation

- Multi-dimensional Newton-Raphson

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \end{bmatrix} = -\begin{bmatrix} f_1\left({}^i x_1, {}^i x_2, {}^i x_3, {}^i x_4\right) \\ f_2\left({}^i x_1, {}^i x_2, {}^i x_3, {}^i x_4\right) \\ f_3\left({}^i x_1, {}^i x_2, {}^i x_3, {}^i x_4\right) \\ f_4\left({}^i x_1, {}^i x_2, {}^i x_3, {}^i x_4\right) \end{bmatrix}$$

Motivation

- System of linear equations also encountered in:
  - Electrical engineering
    - current equations in a resistor network
  - Chemical engineering
    - how final properties relate to batch ingredients
  - Finite element analysis
  - Civil engineering
    - truss under load
- System could be solved by inverting matrix
- Matrix inversion is <u>very</u> time consuming!!!

Manual solution

- Consider the following system of linear algebraic equations

$$
\begin{aligned}
4q + 2r + 3s - 4t &= -1 \\
q - 6r + 2s - 2t &= 2 \\
3r - 2s + 3t &= -3 \\
4q + s + 4t &= 4
\end{aligned}
$$

- You could try to randomly combine equations hoping to eliminate variables
  – Impossible to implement on a computer

Systematic manual solution

- For a systematic approach, you may decide to solve the 1st equation for q :

$$q = -0.5r - 0.75s + 1t - 0.25$$

  – and substitute the result in the 2nd equation:

$$\left(-0.5r - 0.75s + 1t - 0.25\right) - 6r + 2s - 2t = 2$$

$$-6.5r + 1.25s - 1t = 2.25$$

  – and into the 4th equation:

$$4\left(-0.5r - 0.75s + 1t - 0.25\right) + s + 4t = 4$$

$$-2r - 2s + 8t = 5$$

Systematic manual solution (cont.)

- Resulting in a 3rd order system:

$$
\begin{aligned}
-6.5r + 1.25s - 1t &= 2.25 \\
3r - 2s + 3t &= -3 \\
-2r - 2s + 8t &= 5
\end{aligned}
$$

- The same method could now be used to
  – eliminate r from the new Equations 3 and 4, then
  – eliminate s from the newer Equation 4:

  – leaving one equation with one unknown t

Systematic and numeric solution

- While the approach described here is certainly systematic, symbolic manipulations are difficult (or impossible) to implement on a computer

- However, I contend that
  – solving Equation 1 for q,
  – and substituting the result into Equation 2

  – is exactly the same as

Systematic and numeric (cont.)

- taking Equation 1
- multiplying it by the appropriate value
- and subtracting the result from Equation 2

$$\begin{array}{r} q - 6r + 2s - 2t = 2 \\ -\left(1/4\right) * \left(4q + 2r + 3s - 4t = -1\right) \\ \hline -6.5r + 1.25s - 1t = 2.25 \end{array}$$

- The results are certainly the same, and the processes are, in fact, equivalent

Systematic process

- Generic form of a 4th order system of linear eq's

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & a_{24}x_4 & = & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & + & a_{34}x_4 & = & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & + & a_{44}x_4 & = & b_4 \end{array}$$

- q, r, s, t have been replaced by x_1, x_2, x_3, …
- On the coefficient subscripts (a_{ij}), i is the equation number, j matches the variable number

Systematic process (cont.)

- Use Equation 1 to eliminate a_{21}, a_{31}, and a_{41}

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 & = & b_1 \\ 0x_1 & + & a'_{22}x_2 & + & a'_{23}x_3 & + & a'_{24}x_4 & = & b'_2 \\ 0x_1 & + & a'_{32}x_2 & + & a'_{33}x_3 & + & a'_{34}x_4 & = & b'_3 \\ 0x_1 & + & a'_{42}x_2 & + & a'_{43}x_3 & + & a'_{44}x_4 & = & b'_4 \end{array}$$

  - where a single prime indicates one modification away from the original
  - e.g. $a'_{22} = a_{22} - a_{12}\left(\frac{a_{21}}{a_{11}}\right)$, $b'_3 = b_3 - b_1\left(\frac{a_{31}}{a_{11}}\right)$

Systematic process (cont.)

- Now use equation 2' to eliminate a'_{32} and a'_{42}

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 & = & b_1 \\ 0x_1 & + & a'_{22}x_2 & + & a'_{23}x_3 & + & a'_{24}x_4 & = & b'_2 \\ 0x_1 & + & 0x_2 & + & a''_{33}x_3 & + & a''_{34}x_4 & = & b''_3 \\ 0x_1 & + & 0x_2 & + & a''_{43}x_3 & + & a''_{44}x_4 & = & b''_4 \end{array}$$

  - where: $a''_{34} = a'_{34} - a'_{24}\left(\frac{a'_{32}}{a'_{22}}\right)$
  - and $b''_4 = b'_4 - b'_2\left(\frac{a'_{42}}{a'_{22}}\right)$

Systematic process (cont.)

- And finally use equation 3'' to eliminate a''_{43}

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 & = & b_1 \\ 0x_1 & + & a'_{22}x_2 & + & a'_{23}x_3 & + & a'_{24}x_4 & = & b'_2 \\ 0x_1 & + & 0x_2 & + & a''_{33}x_3 & + & a''_{34}x_4 & = & b''_3 \\ 0x_1 & + & 0x_2 & + & 0x_3 & + & a'''_{44}x_4 & = & b'''_4 \end{array}$$

  – where: $a'''_{44} = a''_{44} - a''_{34}\left(\dfrac{a'_{43}}{a'_{33}}\right)$

  – and $b'''_4 = b''_4 - b''_3\left(\dfrac{a''_{43}}{a''_{33}}\right)$

Gauss Elimination

Chapra & Canale
Chapter 9

Systematic and numeric

- But we want a process that is numeric only
  – there were clearly still symbols (variables) in those equations
  – all the a_{ij} terms are actually numbers
- Let's consider the system of equations in its matrix format:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{Bmatrix}$$

Systematic and numeric

- Expressed generically as: $[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}$
- Now we can perform our mathematical operations on rows of the matrix $[\mathbf{A}]$
  – e.g. to eliminate a_{31}, the math looks like this:

$$\begin{array}{r} [a_{31} \quad a_{32} \quad a_{33} \quad a_{34}] \\ -\left(a_{31}/a_{11} \right)[a_{11} \quad a_{12} \quad a_{13} \quad a_{14}] \\ \hline [0 \quad a'_{32} \quad a'_{33} \quad a'_{34}] \end{array}$$

Systematic and numeric

- Furthermore, we notice that the same operations are applied to each b term as are applied the a terms in the same row

$$a''_{34} = a'_{34} - a'_{24}\left(\frac{a'_{32}}{a'_{22}}\right) \qquad b''_3 = b'_3 - b'_2\left(\frac{a'_{32}}{a'_{22}}\right)$$

  – we can think of b_i as $a_{i,N+1}$ (N is the system order)
  – rename b_3 as a_{35} (in the case of a 4th order system)

$$a''_{34} = a'_{34} - a'_{24}\left(\frac{a'_{32}}{a'_{22}}\right) \qquad a''_{35} = a'_{35} - a'_{25}\left(\frac{a'_{32}}{a'_{22}}\right)$$

Gauss Elimination

- Sum total of these observations lead to the Gauss Elimination process
  – Given an Nth order system: $[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}$
  – Concatenate $\{\mathbf{b}\}$ onto the right side of $[\mathbf{A}]$
    - Result is call the augmented A matrix $[\mathbf{A*}]$
    - e.g. for a 3rd order system

$$\left[\mathbf{A}^*\right] = \left[\mathbf{A}|\mathbf{b}\right]_{N\times(N+1)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{bmatrix}$$

Gauss Elimination

– Perform row operations on $[\mathbf{A*}]$ to eliminate all terms below the diagonal in the 1st column
– e.g., to eliminate a_{31}
  - Row3′ = Row3 – ((a_{31}/a_{11}) * Row1)
– Bottom N-1 rows of the resulting $[\mathbf{A*}]'$ matrix represent a self-contained N-1 order system

– Perform row operations on $[\mathbf{A*}]'$ to eliminate all terms below the diagonal in the 2nd column

Gauss Elimination (cont.)

– Move rightward through the columns, continuing the process as you go
  - Last column operated on is the N-1 column, where N is the order of the original system
  - Result is an upper triangular augmented matrix

– e.g. for a 4th order system:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a'_{22} & a'_{23} & a'_{24} & b'_2 \\ 0 & 0 & a''_{33} & a''_{34} & b''_3 \\ 0 & 0 & 0 & a'''_{44} & b'''_4 \end{bmatrix}$$

Back propagation

- Last row represents a 1st order system:

$$a'''_{44} x_4 = b'''_4$$

- Back propagation process:
  - Solve the last row (equation) for x_N
  - With x_N , solve the next to last row for x_{N-1}
  - Continue upward through the rows until all variables are solved

Example: Gauss Elimination

- Solve the following system of equations:

$$\begin{aligned} x_1 + 2x_2 - x_3 + 2x_4 &= -3 \\ -2x_1 - 2x_2 + 4x_3 - 2x_4 &= 6 \\ 4x_1 + 4x_2 + 2x_3 - x_4 &= 3 \\ -x_1 + x_2 - 4x_3 + 2x_4 &= -3 \end{aligned}$$

- Create the augmented $[\mathbf{A^*}]$ matrix:

$$\begin{bmatrix} 1 & 2 & -1 & 2 & -3 \\ -2 & -2 & 4 & -2 & 6 \\ 4 & 4 & 2 & -1 & 3 \\ -1 & 1 & -4 & 2 & -3 \end{bmatrix}$$

Example (cont.)

- Use the first row to:
  - eliminate a_{21}
    - $row2' = row2 - (a_{21}/a_{11})row1$

$$\begin{array}{rrrrrr} & -2 & -2 & 4 & -2 & 6 \\ -\left(\frac{-2}{1}\right) & 1 & 2 & -1 & 2 & -3 \\ \hline & 0 & 2 & 2 & 2 & 0 \end{array}$$

  - eliminate a_{31}
    - $row3' = row3 - (a_{31}/a_{11})row1$

$$\begin{array}{rrrrrr} & 4 & 4 & 2 & -1 & 3 \\ -\left(\frac{4}{1}\right) & 1 & 2 & -1 & 2 & -3 \\ \hline & 0 & -4 & 6 & -9 & 15 \end{array}$$

  - and eliminate a_{41}
    - $row4' = row4 - (a_{41}/a_{11})row1$

$$\begin{array}{rrrrrr} & -1 & 1 & -4 & 2 & -3 \\ -\left(\frac{-1}{1}\right) & 1 & 2 & -1 & 2 & -3 \\ \hline & 0 & 3 & -5 & 4 & -6 \end{array}$$

- the new $[\mathbf{A^*}]$ after one full set of row operations

$$[\mathbf{A^*}]' = \begin{bmatrix} 1 & 2 & -1 & 2 & -3 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & -4 & 6 & -9 & 15 \\ 0 & 3 & -5 & 4 & -6 \end{bmatrix}$$

Example (cont.)

- Use the 2nd row of $[\mathbf{A^*}]'$ to:
  - eliminate a'_{32}
    - $row3'' = row3' - (a'_{32}/a'_{22})row2'$

$$\begin{array}{rrrrr} & -4 & 6 & -9 & 15 \\ -\left(\frac{-4}{2}\right) & 2 & 2 & 2 & 0 \\ \hline & 0 & 10 & -5 & 15 \end{array}$$

  - and eliminate a'_{42}
    - $row4'' = row4' - (a'_{42}/a'_{22})row2'$

$$\begin{array}{rrrrr} & 3 & -5 & 4 & -6 \\ -\left(\frac{3}{2}\right) & 2 & 2 & 2 & 0 \\ \hline & 0 & -8 & 1 & -6 \end{array}$$

- $[\mathbf{A^*}]$ matrix after two full sets of row operations →

$$[\mathbf{A^*}]'' = \begin{bmatrix} 1 & 2 & -1 & 2 & -3 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 10 & -5 & 15 \\ 0 & 0 & -8 & 1 & -6 \end{bmatrix}$$

T05 25

Example (cont.)

- Use the 3rd row of $[\mathbf{A*}]''$ to eliminate a''_{43}
  - $row4''' = row4'' - (a''_{43}/a''_{33})row3''$

$$\begin{array}{r} -8 \quad 1 \quad -6 \\ -\left(-8/10\right) \; 10 \quad -5 \quad 15 \\ \hline 0 \quad -3 \quad 6 \end{array}$$

- The resulting $[\mathbf{A*}]$ after three sets of row operations

$$[\mathbf{A*}]''' = \begin{bmatrix} 1 & 2 & -1 & 2 & -3 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 10 & -5 & 15 \\ 0 & 0 & 0 & -3 & 6 \end{bmatrix}$$

T05 26

Example: Back propagation

- The 4th row of $[\mathbf{A*}]'''$ represents a single equation with one unknown
  - Solve this equation for the last system variable

$$-3x_4 = 6 \qquad \rightarrow \qquad x_4 = -2$$

- The 3rd row of $[\mathbf{A*}]'''$ represents a single equation with two unknowns

$$10x_3 - 5x_4 = 15$$

  - but we know x_4, and can therefore solve for x_3:

$$10x_3 = 15 + 5(-2) \qquad \rightarrow \qquad x_3 = 0.5$$

T05 27

Example: Back propagation

- The 2nd row of $[\mathbf{A*}]'''$ represents a single equation with three unknowns
  - but we know x_4 and x_3 , and can solve for x_2 :

$$2x_2 + 2x_3 + 2x_4 = 0$$

$$2x_2 = -2(0.5) - 2(-2) \qquad \rightarrow \qquad x_2 = 1.5$$

- Finally, solve the top “equation” for the remaining system variable:

$$x_1 + 2x_2 - x_3 + 2x_4 = -3$$

$$x_1 = -3 - 2(1.5) + 0.5 - 2(-2) \qquad \rightarrow \qquad x_1 = -1.5$$

T05 28

Example: Back propagation (cont.)

- We can check our solution in MATLAB:

```
>> A = [1 2 -1 2; -2 -2 4 -2; 4 4 2 -1;
-1 1 -4 2]
>>x = [-1.5 1.5 0.5 -2]'
>>b = A*x
b =
     -3
      6
      3
     -3
```

Which agrees with our original $[\mathbf{b}]$ vector

Terminology

- Current column = column in which we are zeroing out all values below the diagonal
- Pivot term = diagonal term in current column
- Control row (all of the below are true)
  - row that contains the pivot term
  - row that is being used to zero out other terms
  - “control row” number always equals “current column” number

Terminology

- Target row = row in which we are zeroing out the “current column” term
- Row factor = $a_{\text{target row, current column}}$ ÷ pivot term
- Upper triangular = form of the final **[A*]** matrix

Terminology, graphically

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & b_1 \\ 0 & a'_{22} & a'_{23} & a'_{24} & a'_{25} & b'_2 \\ 0 & 0 & a''_{33} & a''_{34} & a''_{35} & b''_3 \\ 0 & a'_{42} & a'_{43} & a'_{44} & a'_{45} & b'_4 \\ 0 & a'_{52} & a'_{53} & a'_{54} & a'_{55} & b'_5 \end{bmatrix}$$

- Current column = 2 = Control row
- Current pivot term = a'_{22}
- Next target row = Row 4
- Next row factor = (a'_{42} / a'_{22})

Aside: There’s no such thing as a free lunch

- This is one of the recurring concepts we will encounter this semester
  - You know it as “three unknowns, three equations”
- Better to think of as:
  - If we wish to determine N pieces of information, we need N pieces of information
- In our current case, we wish to determine the values of x_1 thru x_N , so we need N <u>unique</u> equations

Possible outcomes

- A unique solution exists, and you find it
  - Whoot
- An infinite set of constrained solutions exist
  - Some equations not independent
  - Final 1st order system looks something like: $0{*}x_5 = 0$
- No solution exists
  - Some equations not compatible
  - Final 1st order system looks something like: $0{*}x_5 = 3$

Ramifications of how numbers are stored in computers

- In Naïve Gauss Elimination we are repeatedly multiplying the control row times the row factor
  - in some cases, the row factor may be several orders of magnitude greater than 1
  - in other cases, the row factor may be several orders of magnitude less than 1
- This is undesirable because of how numbers are stored in the computer

Additional danger

- Furthermore, the row factor calculation involves dividing some term by the pivot term
  - if the pivot term is zero, the result is undefined

Solution: Partial pivoting

- Immediately upon entering a new column, swap rows of the (primed) **[A*]** matrix to obtain the largest absolute-value pivot term
  - Find the largest magnitude term in the current column at or below the control row
  - If the largest magnitude term is not in the control row, swap the control row with the row holding the largest magnitude term

T05 37

Solution: Partial pivoting

- Remaining operations stay the same:
  - Reduce the matrix of equations to upper triangular form using row operations
  - Back-substitute to sequentially find solution vector

T05 38

Pseudo-code for Gauss Elimination

Row operations to produce upper triangular form

```
1. column = 1
2.     row = column + 1
3.             factor = A(row,column)/A(column,column)
4.             target_row = target_row – factor*control_row
5.     if not last row, increment row value, go to line 3
6. if not next-to-next-to-last column, increment column,
   go to 2
```

– Note that lines 2&5 represent a `for` loop, as do lines 1&6

T05 39

Pseudo-code for Gauss Elimination

Back substitution to generate answers

```
x(rowsA) = A(rowsA,rowsA+1)/A(rowsA,rowsA)
for row = rowsA-1 to 1 by -1
   sum = 0
   for column = row+1 to rowsA
        sum = sum + A(row,column) * x(column)
   end
   x(row) = (A(row,rowsA+1) - sum)/A(row,row)
end
```

T05 40

Pseudo-code for Gauss Elimination

Partial pivoting operation

upon entering new current column, find location of largest absolute value on or below the diagonal

if max absolute value not on diagonal

swap (row with max absolute value) and (control row)

(perform row operations as normal)

Other Gauss Elim. Enhancements

- Full Pivoting: upon entering a new column, you could swap both rows and columns to get the largest possible absolute value in the pivot term
  - Swapping columns rearranges the positions of x_1, x_2, etc, so you must keep track of the final position of each column!!
- Scaling: It is an easy and effective approach to add a scaling step, i.e. scale each row by the largest element in that row before doing the pivoting operation. (Chapra & Canale, § 9.4.3)

So, what else can go wrong?

- In certain cases, the solution vector **{x}** values can be highly dependent on the values in the **[A]** matrix
- This makes the results susceptible to not only rounding and chopping (number storage) errors, but also to measurement errors in setting up the problem description
- We call a matrix of this type "ill-conditioned."

Ex: Ill-conditioned matrix

- The MATLAB function `cond(A)` calculates the condition number of the matrix **[A]**
- Condition number = 259
  - 0.5% change in $\mathbf{A}(3,2)$
  - causes 29% change in x_3

$$\mathbf{A} = \begin{bmatrix} 15 & 16 & 19 \\ 15 & 19 & 23 \\ 15 & 20 & 25 \end{bmatrix}$$

- Condition number = 2.34
  - 1.0% change in B(3,1)
  - causes 0.7% change in x_2

$$\mathbf{B} = \begin{bmatrix} -15 & 15 & 19 \\ 18 & 19 & 23 \\ 10 & -10 & 20 \end{bmatrix}$$

Solving multiple simultaneous inputs

- We can think of our linear algebraic equations as representing a physical system, in which:
  - **[A]** represents the system characteristics
  - **{b}** represents external inputs
  - **{x}** represents the system's response to those specific inputs
- For example, **[A]** could be the stiffness of an airplane wing, **{b}** could be the aerodynamic loads on the wing, and **{x}** would be the displacements of the wing

T05 45

Multiple simultaneous inputs (cont)

- What if we want to predict the system response under multiple input conditions, ${}_1\{\mathbf{b}\}$, ${}_2\{\mathbf{b}\}$, etc?
- If we consider the three systems:

$$[\mathbf{A}] * {}_1\{\mathbf{x}\} = {}_1\{\mathbf{b}\}$$

$$[\mathbf{A}] * {}_2\{\mathbf{x}\} = {}_2\{\mathbf{b}\} \qquad [\mathbf{A}] * {}_3\{\mathbf{x}\} = {}_3\{\mathbf{b}\}$$

- We could define three unique augmented $[\mathbf{A*}]$ matrices

$${}_1[\mathbf{A*}] = \left[\mathbf{A} \mid {}_1\{\mathbf{b}\}\right]$$

$${}_2[\mathbf{A*}] = \left[\mathbf{A} \mid {}_2\{\mathbf{b}\}\right] \qquad {}_3[\mathbf{A*}] = \left[\mathbf{A} \mid {}_3\{\mathbf{b}\}\right]$$

T05 46

Multiple simultaneous inputs (cont)

- and perform Gauss Elimination three times
  - that would be tedious and <u>redundant</u>, since the row operations depend only on the values in [A]
- Furthermore, matrix math allows us to combine these three equations into one equation:

$$[\mathbf{A}][\mathbf{X}] = [\mathbf{B}]$$

  - where $[\mathbf{X}] = \left[{}_1\{\mathbf{x}\} \middle| {}_2\{\mathbf{x}\} \middle| {}_3\{\mathbf{x}\}\right]$

$$[\mathbf{B}] = \left[{}_1\{\mathbf{b}\} \middle| {}_2\{\mathbf{b}\} \middle| {}_3\{\mathbf{b}\}\right]$$

T05 47

Multiple simultaneous inputs (cont)

  - I leave it to you to convince yourself that ${}_3\{\mathbf{b}\}$ depends only on $[\mathbf{A}]$ and ${}_3\{\mathbf{x}\}$, etc.

- Similarly, we could create a single augmented $[\mathbf{A*}]$ matrix:

$$\left[\mathbf{A}^*\right] = [\mathbf{A} \mid \mathbf{B}] = \left[\mathbf{A} \mid {}_1\{\mathbf{b}\} \mid {}_2\{\mathbf{b}\} \mid {}_3\{\mathbf{b}\}\right]$$

- and perform Gauss Elimination on this “uber” augmented $[\mathbf{A*}]$ matrix

T05 48

Multiple simultaneous inputs (cont)

- After N-1 sets of row operations, the results of the Gauss Elimination process are:

$$\left[\mathbf{A}^*\right]^{final} = \left[\mathbf{A}^{final} \mid {}_1\{\mathbf{b}\}^{final} \mid {}_2\{\mathbf{b}\}^{final} \mid {}_3\{\mathbf{b}\}^{final}\right]$$

- where the $\mathbf{A}^{final}$ part of $[\mathbf{A*}]^{final}$ is upper triangular in form

T05 49

Multiple simultaneous inputs (cont)

- Then we would:
  - use back propagation on $[\mathbf{A}]^{final}$ and ${}_1\{\mathbf{b}\}^{final}$ to solve for the ${}_1\{\mathbf{x}\}$ response,
  - use back propagation on $[\mathbf{A}]^{final}$ and ${}_2\{\mathbf{b}\}^{final}$ to solve for the ${}_2\{\mathbf{x}\}$ response,
  - etc.

T05 50

Gauss Elimination recap

- We can describe the Gauss-Elimination process thus:

$$\left[\mathbf{A}\,|\,\{\mathbf{b}\}\right] \xrightarrow[\text{Elimination}]{\text{Gauss}} \left[\mathbf{A}^{final}\,|\,\{\mathbf{b}\}^{final}\right]$$

- where $\mathbf{A}^{final}$ is upper triangular, and

$$[\mathbf{A}]^{final} * \{\mathbf{x}\} = \{\mathbf{b}\}^{final}$$

- In <u>Gauss Elimination</u>, the row operations are limited to eliminating terms below the diagonal

T05 51

Gauss-Jordan

- In <u>Gauss Jordan</u>
  - 1) the row operations are extended to eliminate ALL non-diagonal elements
    - process will include row operations such as:

$$row1' \Leftarrow row1 - \left(\frac{a_{12}}{a'_{22}}\right) * row2' \qquad row2''' \Leftarrow row2'' - \left(\frac{a''_{24}}{a'''_{44}}\right) * row4'''$$

    - <u>Note</u>: In Gauss-Jordan, row operations will be performed in the Nth column, unlike in Gauss-Elimination
  - 2) After all row operations are done, each row is scaled by the inverse of the pivot term

T05 52

Gauss-Jordan

- Recap of Gauss-Jordan:
  - ALL non-diagonal terms are eliminated with row operations
  - each row is divided by its diagonal term, making the diagonal term equal to 1
- As a result, the final $[\mathbf{A}]$ matrix looks like this:

$${}^{final}[\mathbf{A}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gauss-Jordan

- Therefore, our equation becomes:

$$^{final}[\mathbf{A}]*\{\mathbf{x}\} = {}^{final}\{\mathbf{b}\}$$
$$[\mathbf{I}]*\{\mathbf{x}\} = {}^{final}\{\mathbf{b}\}$$
$$\{\mathbf{x}\} = {}^{final}\{\mathbf{b}\}$$

- And the Gauss-Jordan method can be expressed as

$$[\mathbf{A} \mid \mathbf{b}] \xrightarrow[\text{Jordan}]{\text{Gauss}} [\mathbf{I} \mid \mathbf{x}]$$

Gauss-Jordan

- Our previous example would look like this:

$$\begin{bmatrix} 1 & 2 & -1 & 2 & -3 \\ -2 & 2 & 4 & -2 & 6 \\ 4 & 4 & 2 & -1 & 3 \\ -1 & 1 & -4 & 2 & -3 \end{bmatrix} \xrightarrow[\text{Jordan}]{\text{Gauss}} \begin{bmatrix} 1 & 0 & 0 & 0 & -1.5 \\ 0 & 1 & 0 & 0 & 1.5 \\ 0 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

Why Gauss-Jordan?

- Gauss-Jordan converts an N^{th} order system into quantity N 1^{st} order systems
  - Each equation represented by the final **[A*]** matrix has only one variable (one non-zero coefficient)
  - and thanks to the scaling, the right hand column of the final **[A*]** matrix is exactly the solution vector **{x}**
- Basically, the Gauss Jordan technique has exchanged the back propagation process for additional row operations

Why Gauss-Jordan?

- This is not computationally advantageous
  - Perhaps it is psychologically advantageous
  - Regardless, it does lead to a convenient way of inverting a matrix

Matrix inversion

- We established that we can solve for multiple inputs at once:

$$[\mathbf{A}][\mathbf{X}]=[\mathbf{B}]$$

  – using Gauss Elimination

- We can also solve for multiple inputs using Gauss-Jordan:

$$[\mathbf{A}\,|\,\mathbf{B}]\xrightarrow[\text{Jordan}]{\text{Gauss}}\left[\mathbf{A}^{final}\,|\,\mathbf{B}^{final}\right]=[\mathbf{I}\,|\,\mathbf{X}]$$

Matrix inversion

- We also know that:

$$[\mathbf{A}][\mathbf{X}]=[\mathbf{B}]$$

- So, if $[\mathbf{B}] = [\mathbf{I}]$, then, by definition, $[\mathbf{X}] = [\mathbf{A}]^{-1}$

- Applied to the Gauss-Jordan process, this condition looks like this:

$$[\mathbf{A}\,|\,\mathbf{I}]\xrightarrow[\text{Jordan}]{\text{Gauss}}\left[\mathbf{I}\,|\,\mathbf{A}^{-1}\right]$$

LU Decomposition

Chapra & Canale
Chapter 10

Gauss computational expense

- In Gauss elimination, the row operations are performed on the augmented **[A*]** matrix *after* the **{b}** vector is determined. The number of required mathematical operations is approximately $(2/3)n^3$.
- Is there a way to do some math before-hand, to reduce the number of operations that are necessary after the **{b}** vector is determined?
- Yes. LU Decomposition is one of those methods.

T05 61

Concept – LU Decomposition

- We start with the same system of linear algebraic equations:

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

- Now, let's decompose the coefficient matrix:

$$[\mathbf{A}] = [\mathbf{L}][\mathbf{U}]$$

  – we can do this off-line, before the $\{\mathbf{b}\}$ vector is determined

T05 62

Decomposition concept (cont.)

- Substitute this identity into our original equation:

$$[\mathbf{L}][\mathbf{U}]\{\mathbf{x}\} = \{\mathbf{b}\}$$

- Now, if we let

$$[\mathbf{U}]\{\mathbf{x}\} = \{\mathbf{d}\}$$

  – then

$$[\mathbf{L}]\{\mathbf{d}\} = \{\mathbf{b}\}$$

T05 63

Decomposition concept (cont.)

- This is helpful if we specify that:
  – $[\mathbf{U}]$ is upper triangular
  – $[\mathbf{L}]$ is lower triangular
    - $[\mathbf{L}]$ also has 1's on the diagonal, which we'll discuss later

$$[\mathbf{U}] = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1,N} \\ 0 & u_{22} & u_{23} & \cdots & u_{2,N} \\ 0 & 0 & u_{33} & \ddots & u_{3,N} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{N,N} \end{bmatrix} \qquad [\mathbf{L}] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{N,1} & l_{N,2} & l_{N,3} & \cdots & 1 \end{bmatrix}$$

T05 64

Decomposition concept (cont.)

- Given the stipulated forms of $[\mathbf{U}]$ and $[\mathbf{L}]$,
  – Once $\{\mathbf{b}\}$ is determined, $\{\mathbf{d}\}$ can be calculated through <u>forward</u> substitution:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{N,1} & l_{N,2} & l_{N,3} & \cdots & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_N \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{Bmatrix}$$

$$d_1 = b_1$$
$$d_2 = b_2 - l_{21} d_1$$
$$\vdots$$
$$d_N = b_N - \sum_{p=1}^{N-1} l_{N,p} d_p$$

Decomposition concept (finished)

– With $\{\mathbf{d}\}$ known, $\{\mathbf{x}\}$ can be calculated through backward substitution:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1,N} \\ 0 & u_{22} & u_{23} & \cdots & u_{2,N} \\ 0 & 0 & u_{33} & \ddots & u_{3,N} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{N,N} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_N \end{Bmatrix}$$

$$x_N = \frac{d_N}{u_{N,N}}$$

$$\vdots$$

$$x_1 = \frac{\left(d_1 - \sum_{p=2}^{N} u_{1,p} x_p\right)}{u_{11}}$$

How to determine [L] and [U]

- Let's look at a 3x3 system.
  - The algorithm can be generalized by induction

$$[\mathbf{L}][\mathbf{U}] = [\mathbf{A}]$$

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Decomposition process (cont.)

- Set individual terms equal to one another:
  - Let's start with the really easy ones:

$$u_{11} = a_{11} \qquad u_{12} = a_{12} \qquad u_{13} = a_{13}$$

  - The rest of the first column is pretty easy also:

$$l_{21}u_{11} = a_{21} \quad \Rightarrow \quad l_{21} = a_{21} / u_{11} = a_{21} / a_{11}$$

$$l_{31}u_{11} = a_{31} \quad \Rightarrow \quad l_{31} = a_{31} / u_{11} = a_{31} / a_{11}$$

Decomposition process (cont.)

- Now the rest of the second column:

$$l_{21}u_{12} + u_{22} = a_{22}$$

$$u_{22} = a_{22} - l_{21}u_{12}$$

$$u_{22} = a_{22} - \frac{a_{21}}{a_{11}} a_{12}$$

$$l_{31}u_{12} + l_{32}u_{22} = a_{32}$$

$$l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}}$$

$$l_{32} = \frac{a_{32} - \frac{a_{31}}{a_{11}} a_{12}}{u_{22}}$$

Decomposition process (cont.)

- And finally the rest of the third column:

$$l_{21}u_{13} + u_{23} = a_{23}$$

$$u_{23} = a_{23} - l_{21}u_{13}$$

$$u_{23} = a_{23} - \frac{a_{21}}{a_{11}}a_{13}$$

$$l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33}$$

$$u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$$

$$u_{33} = a_{33} - \frac{a_{31}}{a_{11}}a_{13} - l_{32}u_{23}$$

Let's take a closer look

- Start with 3x3 system

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- Eliminate a_{21}

$$\begin{array}{r} \begin{pmatrix} a_{21} & a_{22} & a_{23} \end{pmatrix} \\ -\left(a_{21} / a_{11} \right) \begin{pmatrix} a_{11} & a_{12} & a_{13} \end{pmatrix} \\ \hline \begin{pmatrix} 0 & a'_{22} & a'_{23} \end{pmatrix} \end{array}$$

- Eliminate a_{31}

$$\begin{array}{r} \begin{pmatrix} a_{31} & a_{32} & a_{33} \end{pmatrix} \\ -\left(a_{31} / a_{11} \right) \begin{pmatrix} a_{11} & a_{12} & a_{13} \end{pmatrix} \\ \hline \begin{pmatrix} 0 & a'_{32} & a'_{33} \end{pmatrix} \end{array}$$

- But a_{21}/a_{11} is exactly l_{21}
- and a_{31}/a_{11} is exactly l_{31}

Solving for $[\mathbf{L}]$ and $[\mathbf{U}]$

- Furthermore,

$$a'_{22} = a_{22} - \left(a_{21} / a_{11} \right) a_{12}$$

- which is exactly u_{22}
- and

$$a'_{23} = a_{23} - \left(a_{21} / a_{11} \right) a_{13}$$

- which is exactly u_{23}

- Continuing our Gauss Elimination
  - Eliminate a_{21}

$$\begin{array}{r} \begin{pmatrix} a'_{32} & a'_{33} \end{pmatrix} \\ -\left(a'_{32} / a'_{22} \right) \begin{pmatrix} a'_{22} & a'_{23} \end{pmatrix} \\ \hline \begin{pmatrix} 0 & a''_{33} \end{pmatrix} \end{array}$$

Solving for $[\mathbf{L}]$ and $[\mathbf{U}]$ (cont.)

- where $\quad a'_{32} / a'_{22} = \dfrac{a_{32} - \left(a_{31} / a_{11} \right) a_{12}}{a_{22} - \left(a_{21} / a_{11} \right) a_{12}}$
  - which is exactly l_{32}
- and

$$\begin{aligned} a''_{33} &= a'_{33} - \left(a'_{32} / a'_{22} \right) a'_{23} \\ &= a_{33} - \frac{a_{31}}{a_{11}} a_{13} - l_{32}u_{23} \end{aligned}$$

  - which is exactly u_{33}

Generalized results

- The terms in the $[\mathbf{U}]$ matrix are exactly the terms that appear in the final upper triangular $[\mathbf{A}]$ matrix
- The non-diagonal terms in $[\mathbf{L}]$ are the row factors that arise during Gauss Elimination

$$l_{21} = a_{21}/a_{11}$$

$$l_{31} = a_{31}/a_{11} \quad l_{32} = a'_{32}/a'_{22}$$

$$l_{41} = a_{41}/a_{11} \quad l_{42} = a'_{42}/a'_{22} \quad l_{43} = a''_{43}/a''_{33}$$

Pivoting

- Since the LU Decomposition process is so closely related to Gauss Elimination, it suffers all the same potential setbacks
  - including problems arising from number storage and orders of magnitude differences between different row factors
- Can we pivot with LU Decomposition?

Pivoting

- Remember, pivoting is equivalent to re-ordering your equations
  - In Gauss Elimination, $\{\mathbf{b}\}$ was part of the $[\mathbf{A^*}]$ matrix, so it was automatically re-ordered at the same time as $[\mathbf{A}]$
- In LU Decomposition, $\{\mathbf{b}\}$ is separate from $[\mathbf{A}]$
  - Pivot operations must be tracked
  - Same pivot operations must be applied to $\{\mathbf{b}\}$ before forward substitution begins

Notes on LU Decomposition

- LU Decomposition is not unique
  - There are other possible forms of matrix pairs (e.g. Crout Decomposition, which is in your textbook)
  - Other decomposition matrices take advantage of special situations, such a symmetrical **[A]** matrix
  - Choices are generally driven by best computational efficiency
- The matrices in LU Decomposition have the advantage of being easily related to the Gauss elimination routine

Jacobi Method
Gauss-Seidel Method

Chapra & Canale

Chapter 11

Remember

- Previously, when confronted with multiple equations in multiple unknowns:

$$7x_1 + 4x_2 - 2x_3 = 2$$
$$2x_1 - 8x_2 + 3x_3 = -3$$
$$-2x_1 + 2x_2 + 6x_3 = 5$$

- We have thought of it in the following terms:

$$y_1 = f_1(x_1, x_2, x_3) = 7x_1 + 4x_2 - 2x_3 = 2$$
$$y_2 = f_2(x_1, x_2, x_3) = 2x_1 - 8x_2 + 3x_3 = -3$$
$$y_3 = f_3(x_1, x_2, x_3) = -2x_1 + 2x_2 + 6x_3 = 5$$

An iterative approach

- Instead, let's think of the first equation as being an equation for x_1 in terms of x_2 and x_3:

$$x_1 = h_1(x_2, x_3) = \frac{2 - (4x_2 - 2x_3)}{7}$$

- Similarly, we can rearrange equations 2 and 3:

$$x_2 = h_2(x_1, x_3) = \frac{-3 - (2x_1 + 3x_3)}{-8}$$

$$x_3 = h_3(x_1, x_2) = \frac{5 - (-2x_1 + 2x_2)}{6}$$

- Now, make initial guesses ${}^0x_1$, ${}^0x_2$, and ${}^0x_3$

Iterative approach (cont)

- Using h_1 and our initial guesses ${}^0x_2$ and ${}^0x_3$, calculate a new guess for x_1

$${}^1x_1 = h_1({}^0x_2, {}^0x_3) = \frac{2 - (4\,{}^0x_2 - 2\,{}^0x_3)}{7}$$

- Similarly, we can calculate new guesses for x_2 and x_3:

$${}^1x_2 = h_2({}^0x_1, {}^0x_3) = \frac{-3 - (2\,{}^0x_1 + 3\,{}^0x_3)}{-8}$$

$${}^1x_3 = h_3({}^0x_1, {}^0x_2) = \frac{5 - (-2\,{}^0x_1 + 2\,{}^0x_2)}{6}$$

T05 81

Iterative approach (cont)

- Repeat this process, using our new guesses each time, until we (hopefully) get convergence
- Let's see how this works out:

T05 82

Example: Jacobi method

T05 83

Example: Jacobi method (cont)

T05 84

Convergence (cont)

- Our example worked out pretty well, but can we be sure it will always work?
- Let's consider the 1st order error propagation approximation we derived from Taylor Series

$$\Delta g_j = \sum_i \left(\left| \frac{\partial g_j}{\partial x_i} \right|_{\tilde{x}} \cdot \Delta x_i \right)$$

- And let's apply that to our original system:

$$x_1 = g_1(x_2, x_3) = \frac{2 - (4x_2 - 2x_3)}{7}$$

$$x_2 = g_2(x_1, x_3)$$

$$x_3 = g_3(x_1, x_2)$$

Convergence (cont)

- From the previous slide, we can estimate the error in x_1 :

$$\Delta x_1 = \left|\frac{\partial g_1}{\partial x_2}\right| \Delta x_2 + \left|\frac{\partial g_1}{\partial x_3}\right| \Delta x_3$$

- Note: Because we're dealing with a linear system, the partial derivatives are constants
- This equation basically says that the error in x_1 is a weighted sum of the errors in x_2 and x_3
- How do we guarantee that the error keeps getting smaller?

Convergence (cont)

- If the sum of the weighting factors is < 1, i.e.

$$\left|\frac{\partial g_1}{\partial x_2}\right| + \left|\frac{\partial g_1}{\partial x_3}\right| < 1$$

- then the error in x_1 is guaranteed to be less than the maximum of the errors in x_2 and x_3
- We can make a similar statement for the 1^{st} equation of an n^{th} order linear system:

$$\left|\frac{\partial g_1}{\partial x_2}\right| + \left|\frac{\partial g_1}{\partial x_3}\right| + \cdots + \left|\frac{\partial g_1}{\partial x_{N-1}}\right| + \left|\frac{\partial g_1}{\partial x_N}\right| < 1$$

Convergence criterion

- However, similar criteria must be met for <u>each</u> of the other equations in the system
- Therefore, we write the convergence criteria for an N^{th} order system:

$$\text{For } i = 1 \ldots N$$

$$\sum_{j=1\ldots N}^{j\neq i} \left|\frac{\partial g_i}{\partial x_j}\right| < 1$$

- For an N^{th} order linear system, we can recast each of the equations in the format:

$$x_i = \frac{b_i - \sum_{j=1\ldots N}^{j\neq i}\left(a_{ij} x_j\right)}{a_{ii}}$$

- And the convergence criterion becomes:

$$\text{For } i = 1 \ldots N \quad \left|a_{ii}\right| \geq \sum_{j=1\ldots N}^{j\neq i} \left|a_{ij}\right|$$

Jacobi algorithm

$$^{i+1}x_1 = \frac{b_1 - \left(a_{12}\,{}^{i}x_2 + \ldots. + a_{1,N}\,{}^{i}x_N\right)}{a_{11}}$$

$$^{i+1}x_2 = \frac{b_2 - \left(a_{21}\,{}^{i}x_1 + a_{23}\,{}^{i}x_3 + \ldots. + a_{2,N}\,{}^{i}x_N\right)}{a_{22}}$$

$$\vdots$$

$$^{i+1}x_N = \frac{b_N - \left(a_{N,1}\,{}^{i}x_1 + a_{N,2}\,{}^{i}x_2 + \ldots. + a_{N,N-2}\,{}^{i}x_{N-1}\right)}{a_{N,N}}$$

Two similar, but different, methods

- Some of you may have noticed that
  - even after we calculate a new estimate for x_1
  - we use the old estimate of x_1 to calculate new values of x_2 , x_3 , etc.
- The Gauss-Seidel Method incorporates each new estimate as it becomes available, e.g.

$$ {}^{new}x_3 = \frac{b_3 - \left(a_{31}\,{}^{new}x_1 + a_{32}\,{}^{new}x_2 + a_{34}\,{}^{old}x_4 + ... + a_{3n}\,{}^{old}x_N \right)}{a_{33}} $$

- Gauss-Seidel generally converges faster, while Jacobi method may be more stable

Gauss-Seidel algorithm

$$ {}^{i+1}x_1 = \frac{b_1 - \left(a_{12}\,{}^{i}x_2 + ... + a_{1,N}\,{}^{i}x_N \right)}{a_{11}} $$

$$ {}^{i+1}x_2 = \frac{b_2 - \left(a_{21}\,{}^{i+1}x_1 + a_{23}\,{}^{i}x_3 + ... + a_{2,N}\,{}^{i}x_N \right)}{a_{22}} $$

$$ \vdots $$

$$ {}^{i+1}x_N = \frac{b_N - \left(a_{N,1}\,{}^{i+1}x_1 + a_{N,2}\,{}^{i+1}x_2 + ... + a_{N,N-1}\,{}^{i+1}x_{N-1} \right)}{a_{N,N}} $$

When are Jacobi / Gauss-Seidel beneficial?

- Jacobi and Gauss-Seidel are particularly useful when dealing with an $[\mathbf{A}]$ matrix that is characterized by:
  - Dominant diagonal terms
  - Lots of zeroes off the diagonal (called “sparse”)
- Jacobi and Gauss-Seidel can save math operation by ignoring the 0 terms

When are Jacobi / Gauss-Seidel beneficial?

- The type of matrix described above is representative of several engineering situations, including:
  - Multi-body spring-mass systems
  - Finite element modeling

Notes

- When presented with a system of equations, don’t just blindly solve the first equation for x_1, the second for x_2, etc.
  - (unless specifically told to do so, as in the homework)

Notes (cont.)

- Rearrange the equations so as to get the largest possible terms in the diagonal
  - Both methods obviously require that the terms in the denominators be non-zero
  - May be able to satisfy, or at least “get closer to” the convergence criterion