

# Applied Engineering Problem Solving

## Lesson #6: Solution of Linear & Nonlinear Equations

Prof. John R. White  
Chemical and Nuclear Engineering  
UMass-Lowell, Lowell MA

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Lesson #6 Goals

What are the real values of  $\underline{x}$  such that  $\underline{f}(\underline{x}) = \underline{0}$ ?

General system of equations (linear or nonlinear):

$$f_1(x_1, x_2, \dots, x_n) = 0$$

$$f_2(x_1, x_2, \dots, x_n) = 0$$

$\vdots$

$$f_n(x_1, x_2, \dots, x_n) = 0$$

$$\underline{f}(\underline{x}) = \underline{0}$$

linear or nonlinear

Gilat:

Chapters 8 and 9

Chapra:

Chapters 8 - 12

Lesson # 6 Lecture Notes  
and Illustrative Examples

If the equations are linear, then we have

$$f_1(x_1, x_2, \dots, x_n) = a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n - b_1 = 0$$

$$f_2(x_1, x_2, \dots, x_n) = a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n - b_2 = 0$$

$\vdots$

$\vdots$

$$f_n(x_1, x_2, \dots, x_n) = a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n - b_n = 0$$

$$\underline{Ax} = \underline{b}$$

linear

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

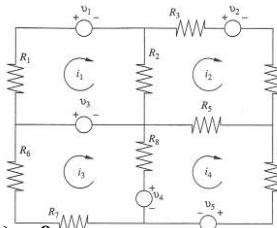
(Nov. 2017)

## Motivation Problems



### Problem 1: Resistive Networks

**Kirchhoff's voltage law states:** the algebraic sum of the voltage drops around a closed loop must be zero



**Loop 1:**  $R_1 i_1 + v_1 + R_2 (i_1 - i_2) - v_3 = 0$

**Loop 2:**  $R_2 (i_2 - i_1) + R_3 i_2 + v_2 + R_4 i_2 + R_5 (i_2 - i_4) = 0$

**Loop 3:**  $R_6 i_3 + v_3 + R_8 (i_3 - i_4) + v_4 + R_7 i_3 = 0$

**Loop 4:**  $-v_4 + R_8 (i_4 - i_3) + R_5 (i_4 - i_2) + R_9 i_4 + v_5 = 0$

**Ohm's Law**  
 $RI = V$

This can be written as  $Ax = b$ , where

$$\begin{bmatrix} R_1 + R_2 & -R_2 & 0 & 0 \\ -R_2 & (R_2 + R_3 + R_4 + R_5) & 0 & -R_5 \\ 0 & 0 & R_6 + R_7 + R_8 & -R_8 \\ 0 & -R_5 & -R_8 & R_5 + R_8 + R_9 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} v_3 - v_1 \\ -v_2 \\ -v_3 - v_4 \\ v_4 - v_5 \end{bmatrix}$$

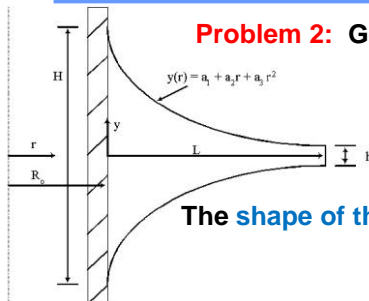
CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 2: Geometry of a Cylindrical Parabolic Fin



The goal is to determine the fin surface area versus length,  $L$ .

The shape of the fin is given by

$$y(r) = a_1 + a_2 r + a_3 r^2$$

with the constraints

$$y(R_0) = \frac{H}{2}, \quad y(R_0 + L) = \frac{h}{2}, \quad \text{and} \quad \left. \frac{dy}{dr} \right|_{r=R_0+L} = 0$$

These three constraints give three equations for the unknown coefficients,  $a_1$ ,  $a_2$ , and  $a_3$  -- which vary as a function of the fin's length,  $L$ .

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

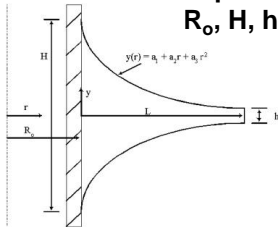
(Nov. 2017)

## Motivation Problems



### Problem 2: Geometry of a Cylindrical Parabolic Fin (cont.)

In particular, for a given set of geometry parameters,  $R_o$ ,  $H$ ,  $h$ , and  $L$ , **applying the above constraints** gives



$$\frac{H}{2} = a_1 + a_2 R_o + a_3 R_o^2$$

$$\frac{h}{2} = a_1 + a_2 (R_o + L) + a_3 (R_o + L)^2$$

$$0 = a_2 + 2a_3 (R_o + L)$$

These three equations can be written in standard  **$Ax = b$**  form:

**Solve for  $a_1, a_2$ , and  $a_3$**

$$A = \begin{bmatrix} 1 & R_o & R_o^2 \\ 1 & R_o + L & (R_o + L)^2 \\ 0 & 1 & 2(R_o + L) \end{bmatrix} \quad x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} H/2 \\ h/2 \\ 0 \end{bmatrix}$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 2: Geometry of a Cylindrical Parabolic Fin (cont.)

For this problem, knowing  $y(r)$  for each  $L$  is not the final result -- **we are interested in the fin's surface area vs.  $L$**  (that is, **heat loss from the fin is related to its heat transfer surface area,  $A$** ).

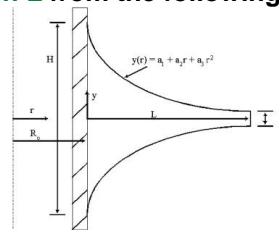
From the sketch, we can **compute  $A$  for each  $L$**  from the following expression:

$A = 2 \times \text{top surface area} + \text{tip area}$

**$ds$  is the differential length along the surface at position  $r$**

$$A = 2 \int 2\pi r ds + 2\pi (R_o + L)h$$

From basic calculus:  $ds = \sqrt{1 + \left(\frac{dy}{dr}\right)^2} dr$



Upon substitution we have:

$$A = 4\pi \int_{R_o}^{R_o+L} r \sqrt{1 + \left(\frac{dy}{dr}\right)^2} dr + 2\pi (R_o + L)h$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 2: Geometry of a Cylindrical Parabolic Fin (cont.)

#### Solution Algorithm:

Set up basic problem parameters ( $R_o$ ,  $H$ ,  $h$ , and range of  $L$  values)

Loop over number of  $L$  values

1. Set up coefficient matrices as defined above
2. Solve system of equations to find the  $a_1$ ,  $a_2$ , and  $a_3$  coefficients for the  $y(r)$  expression
3. Use *quadl* or *integral* to evaluate the fin's surface area as defined above

Plot and tabulate the key results [i.e.  $y(r)$  and  $A$  for several  $L$  values]

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 3: Nonlinear Fin Heat Transfer via the FD Method

The goal here is to determine the temperature distribution,  $T(x)$

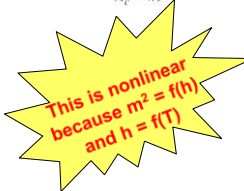
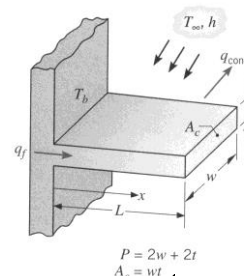
This problem is similar to the fin HT example in Lesson 4, but now there is a **temperature dependent heat transfer coefficient,  $h(T)$** ,

$$h = h(T) = c_1 + c_2 T$$

The **governing continuous ODE** for this problem is still given by

$$\frac{d^2 T}{dx^2} - m^2 (T - T_\infty) = 0 \quad \text{with} \quad m^2 = \frac{hP}{kA_c}$$

but the  $m^2 T$  term now represents a **nonlinear element** since  $m^2$  is a function of  $T$ .



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 3: Nonlinear Fin Heat Transfer via the FD Method (cont.)

For the **Linear Model**, the FD equations are:

node 1  $-(2 + m^2 \Delta x^2) T_1 + T_2 = -m^2 \Delta x^2 T_\infty - T_b$

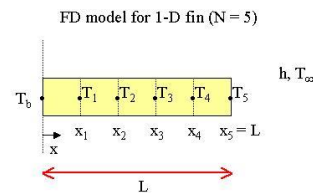
node 2:N-1  $T_{i-1} - (2 + m^2 \Delta x^2) T_i + T_{i+1} = -m^2 \Delta x^2 T_\infty$

node N  $T_{N-2} - \left(2m^2 \Delta x^2 + 1 + \frac{2h\Delta x}{k}\right) T_N = -\left(\frac{2h\Delta x}{k} + 2m^2 \Delta x^2\right) T_\infty$

need to solve  
 $AT = b$

For the case where  $N = 5$ , this gives a **linear matrix equation** with the following structure:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{32} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & 0 & a_{55} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems



### Problem 3: Nonlinear Fin Heat Transfer via the FD Method (cont.)

However, for the **Nonlinear Model**, the FD equations have **temperature (and space) dependent coefficients** since

$$m^2_i = \frac{h_i P}{k A_c} = \frac{P}{k A_c} (c_1 + c_2 T_i)$$

For the case where  $N = 5$ , this gives a **nonlinear matrix equation** with the following structure:

$$\begin{bmatrix} a_{11}(T_1) & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22}(T_2) & a_{32} & 0 & 0 \\ 0 & a_{32} & a_{33}(T_3) & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44}(T_4) & a_{45} \\ 0 & 0 & a_{53} & 0 & a_{55}(T_5) \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} b_1(T_1) \\ b_2(T_2) \\ b_3(T_3) \\ b_4(T_4) \\ b_5(T_5) \end{bmatrix}$$

need to solve  
 $A(T)T = b(T)$   
or  
 $F(T) = 0$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Motivation Problems Summary



These **motivation problems** suggest that we need to develop methods for the solution of **both linear and nonlinear systems of equations**.

**Linear:**  $Ax = b$

**Nonlinear:**  $A(x)x = b(x)$  or  $f(x) = 0$

This lesson will focus on the **Solution of Linear Equations** (both via **hand manipulations for small systems** and via **computer implementation for large systems**)...

Recall that we have already discussed basic **Linear Algebra** terminology & techniques, and performed some hand calculations back in Lesson 2...

And we will also briefly introduce some methods for the **Solution of Nonlinear Equations**, since this class of equations is so important for solving real engineering problems...

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Computer Solution Methods



**Two general schemes** for solving linear systems on the computer:

**Direct Elimination Methods** and **Iterative Methods**

### Direct Methods

All direct methods are based on the standard Gauss Elimination technique, which **systematically applies row operations to transform the original system of equations into a form that is easier to solve**.

We will discuss:

**Gauss Elimination scheme with partial pivoting**

**Basics of the LU Decomposition method** (functionally equivalent to the Gauss Elimination method, but **it provides some additional flexibility for computer implementation**).

Some variation of the **LU decomposition method is often the preferred direct solution method** for low to medium sized systems.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

# Computer Solution Methods (cont.)



Two general schemes for solving linear systems on the computer:

*Direct Elimination Methods* and *Iterative Methods*

## Iterative Methods

For large systems, iterative methods are almost always used.

This switch is required from accuracy considerations (related to round-off errors), from memory limitations for physical storage of the equation constants, from considerations for treating nonlinear problems, and from overall efficiency concerns.

Most iterative methods build upon the base Gauss Seidel method, usually with some acceleration scheme to help convergence.

Thus, our focus as part of Lesson #6 is on the basic Gauss Seidel scheme and on the use of Successive Relaxation (SR) to help accelerate convergence.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

# Gauss Elimination



Direct elimination methods formally implement the three standard row operations:

*normalization by a constant*

*row interchange*

*addition of a constant times one row to another*

The goal is to convert the original fully coupled system into a sequentially coupled system (often called row echelon form) that can be easily solved via back substitution:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix} \Rightarrow \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix}$$

not easy to solve      fully coupled      sequentially coupled      easy to solve

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

# Formal Gauss Elimination Algorithm



With reference to a system of  $n$  equations and  $n$  unknowns,  $Ax = b$ , the *Forward Elimination Step* (with partial pivoting) becomes:

- Step 0 -- Create an augmented matrix,  $\tilde{A} = [A \quad b]$
- Step 1 -- Determine the coefficient in the  $i^{\text{th}}$  column with the largest absolute value and interchange rows such that this element is the pivot element ( $i = 1, 2, 3, \dots, n-1$ )
- Step 2 -- Normalize the pivot equation (i.e. divide by the  $i, i$  element)
- Step 3 -- Multiply normalized eqn.  $i$  by the  $j, i$  element of eqn.  $j$
- Step 4 -- Subtract the resultant equation in Step 3 from eqn.  $j$
- repeat Steps 3 and 4 for  $j = i+1$  to  $n$
- go to Step 1 for next  $i = i+1$  to  $n-1$

**forward  
elimination step**

and the *Back Substitution Step* is given by:

- Step 5 --  $x_n = b'_n / a'_{nn}$
- Step 6 --  $x_i = \left( b'_i - \sum_{j=i+1}^n a'_{ij} x_j \right) / a'_{ii}$
- repeat for  $i = n-1, n-2, \dots, 1$

**backward  
substitution step**

where the primes indicate that the coefficients at this stage are different from the original coefficients.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

# Gauss Elimination via an Example



$$\begin{bmatrix} 3 & -2 & 0 \\ -1 & 3 & -2 \\ 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -7 \\ 9 \\ -5 \end{bmatrix}$$

Step 0 -- form augmented matrix

$$\begin{bmatrix} 3 & -2 & 0 & -7 \\ -1 & 3 & -2 & 9 \\ 0 & -1 & 3 & -5 \end{bmatrix}$$

Step 1 --  $i = 1$  perform partial pivoting

$$\begin{bmatrix} 3 & -2 & 0 & -7 \\ -1 & 3 & -2 & 9 \\ 0 & -1 & 3 & -5 \end{bmatrix} \quad (\text{no change})$$

Step 2 --  $i = 1$  normalize pivot equation

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ -1 & 3 & -2 & 9 \\ 0 & -1 & 3 & -5 \end{bmatrix}$$

Step 3+4 -- multiply eqn.  $i = 1$  by -1 and subtract from eqn.  $j = i+1 = 2$

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ 0 & 7/3 & -2 & 20/3 \\ 0 & -1 & 3 & -5 \end{bmatrix}$$

Step 3+4 -- multiply eqn.  $i = 1$  by 0 and subtract from eqn.  $j = i+2 = 3$

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ 0 & 7/3 & -2 & 20/3 \\ 0 & -1 & 3 & -5 \end{bmatrix} \quad (\text{no change})$$

Step 1 --  $i = 2$  perform partial pivoting

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ 0 & 7/3 & -2 & 20/3 \\ 0 & -1 & 3 & -5 \end{bmatrix} \quad (\text{no change})$$

Step 2 --  $i = 2$  normalize pivot equation

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ 0 & 1 & -6/7 & 20/7 \\ 0 & -1 & 3 & -5 \end{bmatrix}$$

Step 3+4 -- multiply eqn.  $i = 2$  by -1 and subtract from eqn.  $j = i+1 = 3$

$$\begin{bmatrix} 1 & -2/3 & 0 & -7/3 \\ 0 & 1 & -6/7 & 20/7 \\ 0 & 0 & 15/7 & -15/7 \end{bmatrix}$$

Stop elimination phase since  $i = 2 = n-1 = 2$

Step 5 -- evaluate the last element of the vector

$$x_3 = \frac{-15/7}{15/7} = -1$$

Step 6 -- evaluate all other elements (in reverse order)

$$x_2 = \frac{20}{7} + \frac{6}{7}x_3 = \frac{20-6}{7} = 2$$

$$x_1 = -\frac{7}{3} + \frac{2}{3}x_2 = \frac{-7+4}{3} = -1$$

$$\underline{x} = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)



## What about Multiple b vectors?



One **disadvantage** of the **Gauss Elimination** (GE) algorithm is that the **b vector is manipulated along with the A matrix**.

What if we wanted to **solve several systems** with the **same A** but **different b vectors**? For example,

$$\begin{aligned} Ax_1 &= b_1, & Ax_2 &= b_2, & Ax_3 &= b_3, & \text{etc.} \\ A[x_1 \ x_2 \ x_3 \ \dots] &= [b_1 \ b_2 \ b_3 \ \dots] & \text{or simply} & & \mathbf{AX} = \mathbf{B} \end{aligned}$$

discuss calc  
of  $A^{-1}$

We can simply use the standard **GE algorithm multiple times**, but this would be **quite inefficient**, since the **A matrix** would be transformed several times!

Certainly it would be better if we could devise a **method to modify the A and B matrices separately** -- and this is the **advantage** of the **LU Decomposition Method**...

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## LU Decomposition



To develop the basic **LU Decomposition method**, let's break the coefficient matrix into a **product of two matrices**,

$$\mathbf{A} = \mathbf{LU}$$

where **L** is a **lower triangular matrix** and **U** is an **upper triangular matrix**.

Now, the original system of equations,  $\mathbf{Ax} = \mathbf{b}$ , becomes

$$\mathbf{LUx} = \mathbf{b}$$

This expression can be **broken into two problems**,

$$\mathbf{Ly} = \mathbf{b} \quad \text{and} \quad \mathbf{Ux} = \mathbf{y}$$

$$\begin{bmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

use forward substitution

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

use back substitution

both are easy  
to solve

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## LU Decomposition (cont.)



So how do we find the two matrices, L and U?

This is referred to as the **Decomposition Step** and there are a variety of algorithms available (note that this can be performed without knowledge of the b vector).

For example, **Doolittle Decomposition** (for a 4x4 system) would be written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

notice the ones along the diagonal of L

and, because of the specific structure of the matrices, a **systematic set of formulae** for the components of L and U results.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## LU Decomposition (cont.)



A **few steps** in a simple scheme (i.e. **no partial pivoting**) are as follows:

row 1 of L into column 1 of U:  $u_{11} = a_{11}$

row 1 of L into column 2 of U:  $u_{12} = a_{12}$

or  $u_{1j} = a_{1j}$  for  $j = 1, 2, \dots, n$

row 2 of L into column 1 of U:  $\ell_{21}u_{11} = a_{21}$  or  $\ell_{21} = \frac{a_{21}}{u_{11}} = \frac{a_{21}}{a_{11}}$

row 2 of L into column 2 of U:  $\ell_{21}u_{12} + u_{22} = a_{22}$  or  $u_{22} = a_{22} - \ell_{21}u_{12}$   
 $= a_{22} - \frac{a_{21}}{a_{11}}a_{12}$

etc...

partial pivoting is required in any realistic code

This can be developed into an **efficient computational scheme** for the elements of the L and U matrices (with **only one unknown per equation**).

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

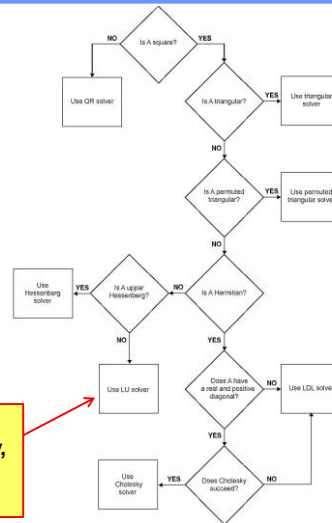
# Matlab's Backslash Operator



```
>> help \
\ Backslash or left matrix
divide.

A\B is the matrix division of A
into B, which is roughly the same
as INV(A)*B , except it is
computed in a different way. If
A is an N-by-N matrix and B is a
column vector with N components,
or a matrix with several such
columns, then X = A\B is the
solution to the equation A*X = B.
```

Unless the A matrix has some special form that can be solved more efficiently,  $x = A \backslash b$  uses an **LU Decomposition** scheme to solve the problem.



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

# Iterative Methods



For large systems, iterative methods are almost always used!!!

A one point iterative formulation can always be written as

$$\begin{aligned} Ax &= b \\ (A_1 - A_2)x &= b \\ A_1x &= A_2x + b \\ x &= A_1^{-1}A_2x + A_1^{-1}b \end{aligned} \quad \rightarrow \quad x^{p+1} = Bx^p + c$$

where **B** is the iteration matrix, **c** is a constant vector, and **p** is an iteration counter.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Convergence of Iterative Methods



Recall that the **disadvantage of one-point iteration schemes is that they are not guaranteed to converge** -- so the convergence properties of a particular scheme is of considerable interest.

Convergence is guaranteed if the **largest eigenvalue of the iteration matrix is less than unity**, where

$$\rho = \text{spectral radius} = |\lambda_{\max}|$$

but  $\rho$  is not usually available

converges for  $\rho < 1$   
diverges for  $\rho > 1$

Since the **spectral radius is not generally available**, another test, **although not as informative**, is often applied.

## Diagonal Dominance



A matrix is said to be **diagonally dominant** if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{for all } i$$

### Rule of Thumb

Most systems derived from physical balance equations are “**nearly diagonally dominant**” and these systems “**usually converge**” ...

This says that “**the magnitude of the diagonal element is greater than the absolute sum of all the other elements in a row**” -- and this must be true for every row.

**Diagonal dominance of the original coefficient matrix is a sufficient (but not necessary) condition for convergence**

- if the system is diagonally dominant it **will converge**
- if the system is not diagonally dominant it **may or may not converge**

## Gauss Seidel Iterative Method



The most common 1-point iteration scheme in use for linear systems is the **Gauss Seidel Method**.

To develop this method, we start with

$$Ax = b$$

and break the original matrix into **three specific components**,

or 
$$A = L + D + U$$

where the 3 matrices on the right hand side, are **strictly lower triangular**, **diagonal**, and **strictly upper triangular matrices**.

For example, for a generic 3x3 system, we have

$$L = \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \quad D = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \quad U = \begin{bmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Gauss Seidel Iterative Method (cont.)



Upon substitution, we have

$$(L + D)x + Ux = b$$

or

$$(L + D)x = b - Ux$$

We now pre-multiply by  $(L + D)^{-1}$  and note that the **solution vector appears on both sides of the equation** -- so we can write the resultant equation in an **iterative form**, with **p as the iteration counter**, as

$$x^{p+1} = -(L + D)^{-1}Ux^p + (L + D)^{-1}b$$

where, clearly, this is in **standard iterative form**

$$x^{p+1} = Bx^p + c$$

with  $B = -(L + D)^{-1}U$  and  $c = (L + D)^{-1}b$



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Gauss Seidel Iterative Method (cont.)



This formal structure is **useful for studying the convergence rate of model problems**, but it is **not useful as a program algorithm**, since **finding the inverse matrix is computationally intensive!!!**

**For actual implementation on the computer**, one writes these equations differently, **never having to formally take the inverse as indicated above**.

In practice, instead of pre-multiplying by  $(L + D)^{-1}$ , we write the equation in iterative form as

$$(L + D)x^{p+1} = b - Ux^p$$

and manipulate this to give

$$Dx^{p+1} = b - Lx^{p+1} - Ux^p$$

or

$$x^{p+1} = D^{-1}(b - Lx^{p+1} - Ux^p)$$

**This is a practical Gauss Seidel algorithm**

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Gauss Seidel Iterative Method (cont.)



This specific form is **somewhat odd at first glance**, since  $x^{p+1}$  **appears on both sides of the equation**.

This formulation is justified because of the **special form of the strictly lower triangular matrix, L**, which can be easily seen if the matrix equations are written explicitly, as follows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{p+1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & 0 \\ 0 & \frac{1}{a_{22}} & 0 \\ 0 & 0 & \frac{1}{a_{33}} \end{bmatrix} \left( \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{p+1} - \begin{bmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^p \right)$$

or as individual equations:

$$x_1^{p+1} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^p - a_{13}x_3^p) \quad x_2^{p+1} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{p+1} - a_{23}x_3^p)$$

**If taken in sequence, all the terms on the right hand side are known...**

$$x_3^{p+1} = \frac{1}{a_{33}}(b_3 - a_{31}x_1^{p+1} - a_{32}x_2^{p+1})$$

**Thus, this is indeed a practical Gauss Seidel algorithm...**

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Successive Relaxation (SR)



Now, since the convergence properties of a system are so important, **to improve the rate of convergence**, one might consider using a weighted average of the results of the two most recent estimates to obtain the next best guess of the solution.

If the solution is converging, this might help extrapolate to the real solution more quickly.

If the solution is diverging, this might help it to converge.

This idea is the basis of the **successive relaxation (SR) method**.

In particular, let  $\alpha$  be some weight factor with a value between 0 and 2 -- this is called the **relaxation factor**.

## Successive Relaxation (SR) (cont.)



Now, let's compute the next value of  $x^{p+1}$  to use in the Gauss Seidel method as a **linear combination of the current value,  $x^{p+1}$ , and the previous solution,  $x^p$** , as follows:

$$x^{p+1}_{\text{new}} = \alpha x^{p+1} + (1 - \alpha) x^p \quad \text{with} \quad 0 < \alpha < 2$$

If  $\alpha$  is unity, we simply get the standard Gauss Seidel method.

When  $\alpha > 1$ , the system is said to be **over-relaxed**, and the system is **under-relaxed** when  $\alpha < 1$ .

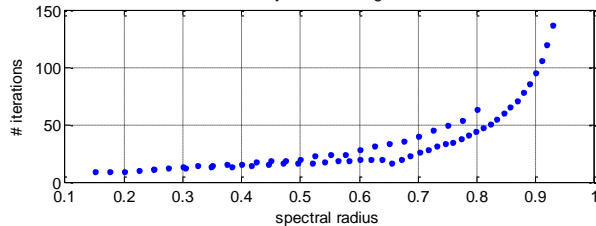
The choice of  $\alpha$  affects the iteration matrix,  $B$ , and the spectral radius,  $\rho$ , and the goal here is to reduce  $\rho$  as much as possible.

An illustrative example is described in detail in the formal Lecture Notes and the results are summarized on the next slide -- **which illustrates nicely how the choice of  $\alpha$  affects the convergence rate of the overall iterative scheme...**

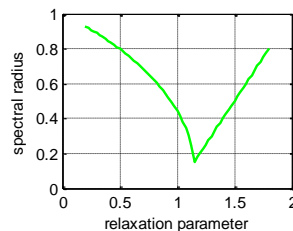
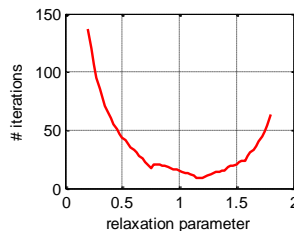
## Successive Relaxation (SR) (cont.)



SRDemo: Results for Analysis of Convergence Rates for SR Method



These results are for a simple 3x3 example system solved via Gauss Seidel with Successive Relaxation



For this situation, the iteration matrix is

$$B = (\alpha L + D)^{-1} ((1 - \alpha)D - \alpha U)$$

Knowing this, one can calculate  $\rho$  vs.  $\alpha$  and the # iterations to converge vs.  $\alpha$  -- and generate the plots shown here...

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## The Linear Motivation Problems & Illustrative Examples



**Problem 1: Resistive Networks**

see  
[resistive\\_networks0.m](#)

**Problem 2: Geometry of a Cylindrical Parabolic Fin**

see  
[parabolic\\_fin1.m](#)

**Illustrative Example: On the Convergence of Iterative Methods**

This example gives further insight into the subject of **diagonal dominance** and how this affects the **convergence of iterative schemes** (specifically the **Gauss Seidel** method).

see  
[conv\\_demo1.pdf](#)

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)



## One More Linear Illustrative Example



### Reaction Stoichiometry

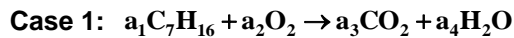
Gives example of two chemical reactions and how to set up and determine the relative mole fractions of the reactants and products.

see  
[reaction\\_eqns.pdf](#)

In a chemical reaction, the number of atoms of each element must be conserved

This conservation law leads to a system of linear equations for the stoichiometric coefficients in the reaction balance equations.

Gives a  
Coupled System  
 $Aa = b$



These examples come directly from your CHEN.2010 Material Balances class

Solution is obtained in Matlab via  $a = A \backslash b$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Techniques for Nonlinear Systems



... Linearized Iteration Method ...

(discuss this via hand illustration)

also discuss Motivation Problem #3

The Nonlinear Fin Heat Transfer Problem

See the formal  
Lecture Notes  
([linear\\_nonlinear\\_eqns.pdf](#))  
for the details...

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Newton's Method (from the Taylor Series)



The one-point linearized iteration scheme outlined above is **not appropriate for many problems** because of the **arbitrary nature for choosing the iteration function**,  $A(x^k) x^{k+1} = b(x^k)$ .

As we have seen before when discussing a single nonlinear equation, the **most common one-point iteration algorithm** used in practice is **Newton's method**.

You should recall that, **for a single equation**, the iteration formula for this method is easily derived using a **truncated Taylor series expansion**,

$$f_{k+1} = f_k + f'_k (x_{k+1} - x_k) + O(\Delta x^2)$$

**Dropping the error term, solving this expression for  $x_{k+1}$ , and setting  $f_{k+1} = 0$ , gives**

$$x_{k+1} - x_k = \frac{f_{k+1} - f_k}{f'_k} \quad \text{or} \quad x_{k+1} = x_k - (f'_k)^{-1} f_k$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Newton's Method (cont.)



Now, of interest here, is finding the solution vector  $\underline{x}$  to a system of **coupled nonlinear equations** written in the form  $\underline{f}(\underline{x}) = \underline{0}$  -- that is, **what is  $\underline{x}$  such that  $\underline{f}(\underline{x}) = \underline{0}$ ?**

This is essentially the same problem as described previously except we now have  $n$  nonlinear equations and  $n$  unknowns -- thus we need to **write the Taylor series for the case of  $n$  independent variables**.

In particular, we can write the **Taylor series expansion for each function**, as

$$f_1(\underline{x}_{i+1}) = f_1(\underline{x}_i) + \left. \frac{\partial f_1(\underline{x})}{\partial x_1} \right|_{\underline{x}_i} (x_{1,i+1} - x_{1,i}) + \left. \frac{\partial f_1(\underline{x})}{\partial x_2} \right|_{\underline{x}_i} (x_{2,i+1} - x_{2,i}) + \dots + \left. \frac{\partial f_1(\underline{x})}{\partial x_n} \right|_{\underline{x}_i} (x_{n,i+1} - x_{n,i}) + O(h^2)$$



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Newton's Method (cont.)



$$f_2(\underline{x}_{i+1}) = f_2(\underline{x}_i) + \left. \frac{\partial f_2(\underline{x})}{\partial x_1} \right|_{\underline{x}_i} (x_{1,i+1} - x_{1,i}) + \left. \frac{\partial f_2(\underline{x})}{\partial x_2} \right|_{\underline{x}_i} (x_{2,i+1} - x_{2,i}) + \dots + \left. \frac{\partial f_2(\underline{x})}{\partial x_n} \right|_{\underline{x}_i} (x_{n,i+1} - x_{n,i}) + O(h^2)$$

and this is for  $f_2(\underline{x}_{i+1})$

etc. for  $n$  equations...

Using summation notation to treat the  $n$  first-derivative terms for the  $k^{\text{th}}$  function,  $f_k(\underline{x})$ , we can generalize the above expressions, as follows:

$$f_k(\underline{x}_{i+1}) = f_k(\underline{x}_i) + \sum_{\ell=1}^n \left. \frac{\partial f_k(\underline{x})}{\partial x_{\ell}} \right|_{\underline{x}_i} (x_{\ell,i+1} - x_{\ell,i})$$

This is for the general case for  $f_k(\underline{x}_{i+1})$

where we have truncated the 2<sup>nd</sup> and higher order terms.

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Newton's Method (cont.)



Note that the second term on the right hand side of this last expression looks like a **matrix times a vector** -- recall that  $\underline{w} = \underline{A} \underline{z}$  is written in discrete form as

$$w_k = \sum_{\ell} a_{k\ell} z_{\ell}$$

Therefore, defining the **Jacobian matrix**,  $\underline{J}(\underline{x})$ , as

$$\underline{J}(\underline{x}) = \left[ \frac{\partial f_k(\underline{x})}{\partial x_{\ell}} \right] \quad \text{for } \ell = 1, 2, \dots, n$$

and the **increment vector** on the  $i^{\text{th}}$  step,  $\underline{h}_i$ , as

$$\underline{h}_i = \underline{x}_{i+1} - \underline{x}_i = [x_{\ell,i+1} - x_{\ell,i}]$$

the boxed equation from the previous slide becomes

$$\underline{f}(\underline{x}_{i+1}) = \underline{f}(\underline{x}_i) + \underline{J}(\underline{x})|_{\underline{x}_i} \underline{h}_i$$

CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)

## Newton's Method (cont.)



This matrix equation is of the **same form as for a single nonlinear function  $f(x)$** .

As before, we **solve this matrix expression for  $\underline{x}_{i+1}$**  and set  **$f(\underline{x}_{i+1}) = 0$** , since this represents the next estimate of the vector  $\underline{x}$  such that  **$f(\underline{x}) = 0$** .

Doing this gives

$$\underline{J}(\underline{x})|_{\underline{x}_i} \underline{h}_i = -\underline{f}(\underline{x}_i)$$

or

$$\underline{x}_{i+1} = \underline{x}_i + \underline{h}_i \quad \text{where} \quad \underline{h}_i = -\underline{J}(\underline{x})|_{\underline{x}_i}^{-1} \underline{f}(\underline{x}_i) = -\underline{J}(\underline{x})|_{\underline{x}_i} \backslash \underline{f}(\underline{x}_i)$$

This is Newton's Method for the case of multiple equations...

where the last equality uses **Matlab's backslash operator** to actually solve for the increment in the  $\underline{x}$  vector on step  $i$ ...

## More Examples (Nonlinear Cases)



### Problem 3: The Nonlinear Fin Heat Transfer Problem

We have already solved this problem using the **Linearized Iteration** method.

see  
[rect1d\\_fin\\_3.m](#)

### Basic Demo: Example from the Lecture Notes

This example involves a simple 3x3 system solved via **Linearized Iteration**, the **Newton method**, and using Matlab's built-in ***fsolve*** command.

see  
[nldemo1\\_lesson6.m](#)  
[nldemo2\\_lesson6.m](#)  
[nldemo3\\_lesson6.m](#)

## More Examples (Nonlinear Cases)



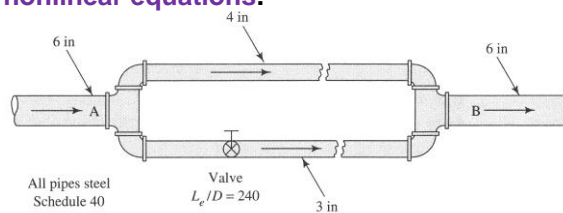
### Illustrative Example: A Two-Pipe Parallel Flow System

This example involves the simple parallel flow system shown in the sketch given below.

Clearly, the  $\Delta P = P_A - P_B$  must be the same in both the upper and lower paths.

However, since **balancing the friction loss** in each line involves terms **containing  $Q^2$** , the problem gives a **system of nonlinear equations**.

see  
[parallel\\_flow1.pdf](#)  
(Linearized Iteration Method)  
[parallel\\_flow2.pdf](#)  
(using Matlab's fsolve function)



CHEN.3170 Applied Engineering Problem Solving  
Lesson 6: Solution of Linear & Nonlinear Equations

(Nov. 2017)