Lecture #01

Finite Difference-based Numerical Methods in Chemical Engineering

Recommended Books:

1) Numerical Methods for Engineers (Gupta)
2) Numerical Methods for Engineering Application (Ferziger)
3) Linear Algebra (Strang)

Objective: Learn a few basic numerical techniques (Finite Difference) to solve simple ODEs/PDEs/algebraic equations derived from a variety of heat and mass transfer related conservation equations.

- Analytical methods yield exact solutions, often in infinite series; use mathematical functions.
- Numerical methods yield approximate solutions; use numbers, computations; results are dependent on the method accuracies and limited by computers machine errors.

Course focuses on

- Algebra (Matrix Operation)
- Methods (Finite Difference)

Requires knowledge of

- Computer Programming (because large # of iterations)
  (there are engineering software, e.g., Matlab, NAG, IMSL, Polymath)
- No numerical analysis in this course!

Examples: (Applications in chemical engineering)

1. Distillation:

Assume: Constant molar flowrates of vapor & liquid.

Binary components: A, B

(more volatile)

Species balance on $i^{th}$ plate:
\[ L_i = x_i L \]
\[ V_i = y_i V \]
\[ y_i = k_i x_i \quad \text{(equilibrium)} \]

Therefore,
\[ V_i = \left( \frac{y_i}{x_i} \right) \left( \frac{V}{L} \right) L_i \]
or
\[ V_i = A_i L_i \]

And,
\[ V_i + L_i = V_{i+1} + L_{i-1} \quad \text{(in = out)} \]

\[
\begin{array}{c}
V_i \\
V_{i+1}
\end{array}
\quad \downarrow l_{i-1}
\quad \downarrow l_i \quad \ldots \quad i^{\text{th}} \text{ plate}
\]

\[ V_i + V_i/A_i - \frac{V_{i-1}}{A_{i-1}} - V_{i+1} = 0 \]

Re-arrange
\[
A_{i-1} V_{i-1} + (1 - A_i) V_i - V_{i+1} = 0
\]

\[ i = 1, N \]

Thus, there are ‘N’ equations and (N+2) variables \((V_0 \cdots V_{N+1})\)

However, \(V_{N+1}\) and \(L_o\) are known (inlet or boundary conditions)

Re-arrange for all trays \(i = 1, N\)

\[
(1 - A_1) V_1 - V_2 = -A_0 V_o (= L_o) \quad \text{(known)}
\]

\[ A_1 V_1 + (1 - A_2) V_2 - V_3 = 0 \quad \text{(1)} \]

\[ : \]

\[ A_{N-1} V_{N-1} + (1 - A_{N+1}) V_N = V_{N+1} \quad \text{(known)} \]

\( \text{(N)} \)

We have a set of algebraic equations (linear/non-linear) represented as

\[
A \vec{X} = \vec{b}
\]

coefficient matrix \quad column vectors or \quad \(N \times 1\) matrix

\[
A = \begin{bmatrix}
(1 - A_1) & -1 & 0 & 0 & 0 & 0 \\
A_1 & (1 - A_2) & -1 & 0 & 0 & 0 \\
0 & A_2 & (1 - A_3) & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & A_{N-1} & (1 - A_{N+1})
\end{bmatrix}_{N \times N}
\]
\[
\tilde{X} = \begin{bmatrix}
V_1 \\
V_2 \\
\vdots \\
V_N \\
\end{bmatrix}, \quad \tilde{b} = \begin{bmatrix}
L_0 \\
\vdots \\
V_{N+1} \\
\end{bmatrix} \quad \text{(Known)}
\]

Alternatively*,
\[
\begin{bmatrix}
(1 - A_1) & -1 & 0 & 0 & 0 & 0 \\
(1 - A_1) & -1 & 0 & 0 & 0 & 0 \\
A_1 & (1 - A_2) & -1 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & A_{N-1} & (1 - A_{N+1})
\end{bmatrix}_{N \times N}
\begin{bmatrix}
V_1 \\
V_2 \\
\vdots \\
V_N \\
V_{N+1}
\end{bmatrix}_{N \times 1}
= \begin{bmatrix}
L_0 \\
\vdots \\
\end{bmatrix}_{N \times 1}
\]

* represented as matrix multiplication

**Note:**
- Most of the finite difference methods lead to the discretization of ODEs/PDEs and a set of algebraic equations.
- You have to solve a set of algebraic equations using Gauss Elimination, G-Jordan etc.

**Example 2:**

**Particle settling**

\[
\begin{align*}
\frac{dv}{dt} &= g - \frac{\rho_f}{\rho_p} g - C_D \left( \frac{1}{2} \rho_f v^2 \right) \frac{A_p}{\frac{\pi}{4} \left( \frac{dp}{4} \right)^2} \\
\end{align*}
\]

**Force balance on the particle:**

\[
\begin{align*}
\frac{dv}{dt} &= g - \frac{\rho_f}{\rho_p} g - C_D \left( \frac{1}{2} \rho_f v^2 \right) \left( \frac{dp}{4} \right)^2 \\
\end{align*}
\]
\[ \frac{dv}{dt} = A - B C_D(v) v^2 \]
\[ = A - B \Phi(v) \]
\[ t = 0, \quad v = 0 \]

Therefore, you have to solve an initial value problem (ODE)

**Solution:**

- linear or non-linear
- homogeneous or non-homogeneous
- or a set of ODEs.

**Methods:** RK-4, Euler Forward, etc.

**Example 3:** Heating or cooling of a moving particle \((\nabla T(r) = 0)\)

1. Energy balance over entire particle
   \[ \frac{hd_p}{k_p} \to 0, \quad k_p >> hd_p \]
   \[ mC_p \frac{dT}{dt} = -hA_s(T - T_f) \]
   \[ (A_s = \pi d_p^2) \]
   \[ (No \: temperature \: gradient \: inside \: the \: particle) \]
   \[ h = h(Nu) \Rightarrow \frac{hd_p}{k_f} = f(R_e,p_r) \]
   \[ or \quad h = h(v_p); \quad Re = \frac{v_p d_p \rho \mu}{\mu_f}, \quad Pr = \left( \frac{C_p \mu}{k} \right)_f \]

2. Time-dependence of velocity (see previous example) and energy balance equation can be written as
\[
\frac{dv_p}{dt} = A - B\Phi(v_p) \\
\frac{dT}{dt} = CD(v_p) - E \\
\] 
\[
t = 0 \quad v_p = 0, \quad T = T_o \text{ Type equation here.}
\]

In this example, two ODEs are to be solved; first \(y_1\) and then \(y_2\):

\[
\frac{dy_1}{dt} = f(y_1) \\
\frac{dy_2}{dt} = \Phi(y_1, y_2) \\
\] 

Two ODEs

**Example 4:** Heating or cooling of a stationary solid \((\nabla T \neq 0)\) in a moving fluid

\[
t = 0 \quad T = T_o > T_f \text{ (Constant)} \\
\] 

\[T(t, r) = ? \]

\[T_o \quad r_p \quad T_f \]

\[h_f \text{(convective heat transfer coefficient)}\]

Assume \(\nabla T(r) = 0\), i.e., there is temperature gradient inside the particle

Energy balance within the particle:

\[
\rho C_p \frac{\partial T}{\partial t} = k \frac{\partial}{\partial r} \left( r^2 \frac{\partial T}{\partial r} \right)
\]

\[
t = 0 \quad T = T_o \quad r_p \geq r \geq 0 \quad \text{(Initial condition)}
\]

\[
= 0^+ \quad r = 0 \quad \frac{\partial T}{\partial r} = 0 \quad \text{(Boundary conditions)}
\]

\[
r = r_p, \quad -k \frac{\partial T}{\partial r} = h(T - T_f)
\]

Now, we have to solve a PDE equation (unsteady - state 1D on space-problem)

Under the steady-state condition with/without a source term, the equation will be reduced to a 2nd order ODE or a boundary value problem.
Example 5: Mass transfer in a tubular liquid flow (steady-state)

\[ C_A = C_{\text{in}} \]

\[ t = 0 \]

\[ \text{tube length} = L \]

For \( R_e < 2100 \)

\[ v_z(r) = 2v_o \left( 1 - \frac{r^2}{R^2} \right) \]

\[ c(t, z, r) = ? \]

Species (A) balance in the CV \((2\pi r \Delta z \Delta r)\)

\[ \frac{\partial C_A}{\partial t} + v_z \nabla C_A = D \nabla^2 C_A + r(= 0) \]

(no reaction)

\[ \frac{\partial C_A}{\partial t} + 2v_o \left( 1 - \frac{r^2}{R^2} \right) \frac{\partial C_A}{\partial z} + v_r \frac{\partial C_A}{\partial r} = D \left( \frac{\partial C_A}{\partial z^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial C_A}{\partial r} \right) \right) \]

\[ t = 0, \quad C_A = 0 \quad \text{everywhere} \]

\[ 0^+ \quad z = 0 \quad C_A = C_{\text{in}} \]

\[ z = L \quad \frac{\partial C_A}{\partial z} = 0 \quad \text{(long tube approximation)} \]

\[ r = 0, \quad \frac{\partial C_A}{\partial r} = 0 \quad \text{(symmetric)} \]

\[ R = \frac{\partial C_A}{\partial r} = 0 \quad \text{(non-reactive wall)} \]

In this example, we have a transient/unsteady-state 2D (space) problem or a PDE to solve.

Under the steady-state condition with/without a source term, the equation will be reduced to an elliptic PDE or 2nd order (in both \( z \) and \( r \)) PDE.
Lecture #02

Linear Algebraic Equations

\[
\begin{align*}
    f_1(X_1, X_2, \ldots, X_n) &= 0 \\
    f_2(X_1, X_2, \ldots, X_n) &= 0 \\
    \vdots & \ \\
    f_n(X_1, X_2, \ldots, X_n) &= 0
\end{align*}
\] system of \( n \) linear algebraic equations

or

\[
\begin{align*}
    a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n &= b_1 \\
    a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_n &= b_2 \\
    \vdots & \ \\
    a_{n1}X_1 + a_{n2}X_2 + \cdots + a_{nn}X_n &= b_n
\end{align*}
\] \( n \) \# of independent equations to solve \( n \) \# of variables

or

\[
AX = b \quad \text{or} \quad \bar{X} = \bar{b}
\]

where \( A = \) coefficient matrix

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}_{n \times n}
\] \( (a_{ij} \text{ is a general element; } i, j = 1 \text{ to } n) \)

\[
X = \text{matrix} = \begin{bmatrix}
    X_1 \\
    X_2 \\
    \vdots \\
    X_n
\end{bmatrix}_{n \times 1} \quad \text{or}
\]

\[
b = \text{matrix} = \begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{bmatrix}_{n \times 1}
\]

\[
\bar{X} = \text{column vector} = \begin{bmatrix}
    X_1 \\
    X_2 \\
    \vdots \\
    X_n
\end{bmatrix}, \quad \bar{b} = \text{column vector} = \begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{bmatrix}
\]
Note: In general, a matrix has $n \times m$ elements

Therefore, $a_{ij}$: $i = 1$ to $n$, $j = 1$ to $m$

If $n = m$, there may be unique solution and the rank of the matrix = $m$

In general, rank $>, =, < m$ (3 cases)

(For more, refer a book on matrix operation)

In this course, we will be interested in solving number of equations with as many variables:

$$\begin{align*}
3x + 2y &= 5 \\
x - y &= 6 \\
3x + 2y + z &= 5 \\
x + y - z &= 2 \\
2x + 2y - 2z &= 4
\end{align*}$$

etc.

Some properties of a matrix and operation:

1. Symmetric matrix, $a_{ij} = a_{ji}$
2. Square matrix, $n = m$
3. Diagonal matrix, $\begin{bmatrix}
\ldots & 0 \\
0 & \ldots \\
\end{bmatrix}$ (only diagonal elements)

Banded matrix (tridiagonal):

$$\begin{bmatrix}
a_{11} & a_{12} & 0 & 0 & \ldots & 0 \\
a_{21} & a_{22} & a_{23} & 0 & \ldots & 0 \\
0 & a_{32} & a_{33} & a_{34} & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & a_{n-1} & a_{nn}
\end{bmatrix}_{n \times n}$$

(First and last rows have only two elements and all other rows have 3 elements, one on the diagonal and the other two on each side of the diagonal element)

Upper/Lower triangular matrix:

(All elements below and above the diagonal are zero, respectively)

$$\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\
a_{22} & a_{23} & \ldots & a_{2n} \\
& & \ddots & \vdots \\
& & & \ddots & \vdots \\
0 & & & & a_{n-1,n} \\
& & & & a_{nn}
\end{bmatrix}_{n \times n}$$

$$\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \ldots & 0 \\
a_{21} & a_{22} & a_{23} & \ldots & \vdots \\
& & \ddots & \ddots & \vdots \\
& & & \ddots & \ddots \\
0 & & & & 0
\end{bmatrix}_{n \times n}$$

(2) Operation:

Summation: $[A]_{m \times n} = [B]_{m \times n} + [C]_{m \times n}$
Multiplication (rules):
\[
[A]_{m \times n} \times [B]_{n \times l} = [C]_{m \times l}
\]

Therefore, multiplication is not permissible for \([B] \times [A]\)
\[
\begin{cases}
  n \times l & m \times n \\
  \text{different} & l \neq m
\end{cases}
\]

Also, these multiplications are permissible
\[
\begin{bmatrix}
  5 & 1 \\
  3 & 2 \\
  0 & 3
\end{bmatrix}_{3 \times 2}
\begin{bmatrix}
  -3 & 0 \\
  1 & 0
\end{bmatrix}_{2 \times 2};
\begin{bmatrix}
  5 & 1 \\
  3 & 2 \\
  0 & 3
\end{bmatrix}_{3 \times 2}
\begin{bmatrix}
  -4 & -3 & 2 \\
  2 & 7 & -3
\end{bmatrix}_{2 \times 3};
\begin{bmatrix}
  1 & 3 \\
  2\end{bmatrix}_{1 \times 3}
\begin{bmatrix}
  0 \\
  5 \\
 -8
\end{bmatrix}_{3 \times 1}
\]

Rule: 
\[
C_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}
\]

Examples:
\[
\begin{bmatrix}
  3 & 1 \\
  8 & 6 \\
  0 & 4
\end{bmatrix}_{3 \times 2}
\begin{bmatrix}
  5 & 6 \\
  7 & 2
\end{bmatrix}_{2 \times 2}
= \begin{bmatrix}
(3 \times 5 + 1 \times 7) & 3 \times 6 + 1 \times 2 \\
(8 \times 5 + 6 \times 7) & (8 \times 6 + 6 \times 2) \\
0 \times 5 + 4 \times 7 & (0 \times 6 + 4 \times 2)
\end{bmatrix}_{3 \times 2}
\]

Inverse of a matrix:
\[
[A]^{-1} \Rightarrow [A]_{m \times n} \times [A]_{n \times m}^{-1} = [I] = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Identity matrix

For \(2 \times 2\) matrix
\[
[A]^{-1} = \frac{1}{|A|} \begin{bmatrix}
a_{22} & -a_{12} \\
-a_{21} & a_{11}
\end{bmatrix}
\text{ where } A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\]

Note: \(|A| = (a_{11} \times a_{22} - a_{12} \times a_{21}) \neq 0\)
else, matrix A is considered to be singular.
For bigger size matrices, there are special methods to determine\([A]^{-1}\), discussed later.

Transpose of a matrix \([A]^T\):
\[
[A] = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}_{m \times n},
[A]^T = \begin{bmatrix}
a_{11} & a_{21} & \cdots & a_{m1} \\
a_{1n} & a_{2n} & \cdots & a_{mn}
\end{bmatrix}_{n \times m}
\]

(rows have been interchanged with columns)
example: \( [A] = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}_{n \times 1} \Rightarrow [A]^T = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}_{1 \times n} \)

Let us get back to solving a set of algebraic equations:

\[
\begin{align*}
[A][X] &= \{b\} \\
\text{or } [A][X] &= \{b\} & b^T &= [b_1 \ b_2 \ \cdots \ b_n], \\
\text{or } A\hat{X} &= \vec{b} & X^T &= [X_1 \ X_2 \ \cdots X_n] \\
\end{align*}
\]

\[
\begin{bmatrix} X \end{bmatrix} = A^{-1}b
\]

Finding solution to the set of linear algebraic equation is nothing but finding inverse of the coefficient matrix, followed by multiplication with \( b \).

Generally, Cremer’s rule works nicely for small matrix:

\[
X_j = \frac{|A_j|}{|A|}
\]

\( A_j \) is determined by replacing \( j^{th} \) column of \( A \) by \( b \).

Therefore, if \( A = \begin{bmatrix} 3 & 5 & 7 \\ 2 & 0 & 8 \\ -1 & 2 & 5 \end{bmatrix} ; b = \begin{bmatrix} 7 \\ 3 \\ 2 \end{bmatrix} \)

\[
X_2 = \begin{bmatrix} 3 & 7 & 7 \\ 2 & 3 & 8 \\ -1 & 2 & 5 \end{bmatrix}/|A|; \ |A| \neq 0
\]

\[
X_3 = \begin{bmatrix} 3 & 5 & 7 \\ 2 & 0 & 3 \\ -1 & 2 & 2 \end{bmatrix}/|A|, \text{etc.}
\]

Let us briefly discuss unique solution, singularity, and rank \( r \) of a \( (m \times n) \) matrix. (Read the textbook for details)

Consider the following set of equations:

\[
\begin{align*}
(1) \quad 2x_1 + 3x_2 &= 11 \\
4x_1 + 6x_2 &= 22 \\
\end{align*}
\]

Note: Two equations are not independent.

and \( |A| = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix}_{2 \times 2} = 0 \)
Therefore, there is no unique solution, coefficient matrix is singular and \( r \) of such matrix = 1 < 2 or \( m(\# \text{ of equations}) < n(\# \text{ of variables}) \)

\[(2) \quad 2x_1 + 3x_2 = 11 \\
\quad \quad \quad x_1 + x_2 = 4 \]

**Note:** Two equations are independent.

\[
|A| = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix}_{2 \times 2} \neq 0; \quad \text{and} \quad m = n, \ r = 2(n) \]

There is a unique solution.

Geometrically, two equations represent two straight lines intersecting at a unique point.

Similarly, 3 independent equations for

3 variables will represent 3 planes on a geometrical space; two planes intersecting

will yield a straight line, and two straight lines intersecting will yield a point \((X_1, X_2, X_3)\), or the unique solution to the equations.

\[(3) \quad 2x_1 + 3x_2 - x_3 = 11 \\
\quad \quad \quad x_1 + x_2 + x_3 = 4 \\
\quad \quad \quad x_1 + 2x_2 + 2x_3 = 7 \]

**Note:** There are three linearly independent equations representing three non-parallel equations. In this case, \( m = n, \ r = 3(n) \).
Lecture #03

Gauss Elimination

It is the most widely used method to solve a set of linear algebraic equations. The method reduces the original matrix to an upper triangular matrix which can also be used to determine the determinant of the matrix. Interestingly, one can also inspect the intermediate steps/solutions to determine singularity, rank, and number of independent equations.

General steps: \[ A\vec{x} = \vec{b} \] where \( A \) is a 3 × 3 matrix

\[ a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \quad \cdots \cdots R_1 \]

\[ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \quad \cdots \cdots R_2 \]

\[ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \quad \cdots \cdots R_3 \]

or

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

\( R_1 \), first row is called pivot row and first non-zero element \( (a_{11}) \) is called pivotal element. Subsequent operations are performed around pivot row and pivotal element.

Step 1:

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
0 & a_{22}' & a_{23}' \\
0 & a_{32}' & a_{33}'
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\
b_2' \\
b_3'
\end{bmatrix}
\]

where

\[
a_{22}' = a_{22} - \left( \frac{a_{21}}{a_{11}} \right) \times a_{12}
\]

\[
a_{23}' = a_{23} - \left( \frac{a_{21}}{a_{11}} \right) \times a_{13}
\]

Briefly, the first equation is divided by \( a_{11} \), multiplied by \( a_{21} \), and subtracted from the second equation to yield the modified equation (2)

\[
a_{11} \neq 0
\]

Similarly,

\[
a_{32}' = a_{32} - \left( \frac{a_{31}}{a_{11}} \right) \times a_{12}
\]

\[
a_{33}' = a_{33} - \left( \frac{a_{31}}{a_{11}} \right) \times a_{13}
\]

briefly, the first equation is divided by \( a_{11} \), multiplied by \( a_{31} \), and subtracted from the 3\text{rd} equation to yield the modified equation (3)

\[
b_3' = b_3 - \left( \frac{a_{31}}{a_{11}} \right) \times b_1
\]

or, in the simplest form \( R_2' = R_2 - \frac{b_1}{a_{11}} R_1 \)
\[ R'_3 = R_3 - R_1 \left( \frac{a_{31}}{a_{11}} \right) \]

**Note:** Multiplication or division of an equation, or subtraction or addition of one independent equation from/to another independent equation does not yield a new independent equation. Therefore, at the end of step 1 we still have three linearly independent equations, although modified, to solve.

Geometrically, two planes (non-parallel) will intersect to yield a straight line. Therefore, step 1 has yielded two straight lines by the intersections of plane 1 (eq 1) with plane 2 (eq 2) and with plane 3 (eq 3).

**Step 2:** Follow the similar procedure. Now \( R'_2 \) (2nd row) becomes pivotal equation and \( a'_{22} \) becomes pivotal element.

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
0 & a'_{22} & a'_{23} \\
0 & 0 & a''_{33}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
b_1 \\
b_2' \\
b_3''
\end{bmatrix}
\]

where,

\[ a''_{33} = a'_{33} - \left( \frac{a'_{32}}{a'_{22}} \right) \times a'_{23} \]

\[ b''_{3} = b'_3 - \left( \frac{a'_{32}}{a'_{22}} \right) \times b'_2 \]

or in the simplest form, \( R''_3 = R'_3 - R'_2 \left( \frac{a'_{32}}{a'_{22}} \right) \)

Geometrically, two straight lines (\( R'_2 \) and \( R'_3 \)) intersect at a point.

**Note:** At the end of step (2) an upper triangular matrix is obtained. The three modified algebraic equations are as follows:

\[
\begin{align*}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 &= b'_2 \\
a''_{33}x_3 &= b''_{33}
\end{align*}
\]

This is a new set of three linearly independent algebraic equations to solve! Here ‘new’ means ‘modified’.
Reverse (back) substitution (to determine $\bar{x}$):

\[
x_3 = \frac{b_3''}{a_{33}''} \quad \text{(from the last row)}
\]

\[
x_2 = \frac{b_2' - a_{23}' x_3}{a_{22}'} \quad \text{(from the 2\textsuperscript{nd} last row)}
\]

\[
x_1 = \frac{b_1 - a_{12} x_2 - a_{13} x_3}{a_{11}} \quad \text{(from the 1\textsuperscript{st} row)}
\]

In general, for the $i$\textsuperscript{th} variable of 'n' equations

\[
x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}}
\]

A large number of equations ($n > 3$) requires a programming code to solve.

**Evaluation of determinant:**

Consider a simple $2 \times 2$ matrix

\[
A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}_{2\times2} \Rightarrow detA = (a_{11} \times a_{22} - a_{21} \times a_{12})
\]

Apply GE method

\[
A' = \begin{bmatrix}
a_{11} & a_{12} \\
0 & a_{22}'
\end{bmatrix}_{2\times2}; \quad a_{22}' = (a_{22} - \left(\frac{a_{21}}{a_{11}}\right) \times a_{12})
\]

The determinant of the modified matrix $A' = (a_{11} \times a_{22} - a_{21} \times a_{12}) = \text{same as det}(A)$

"Value of determinant is not changed by the forward elimination step of GE". This must be true because forward step only modifies the equations.

Take $3 \times 3$ matrix:

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}_{3\times3} \xrightarrow{\text{GE}} \begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
0 & a_{22}' & a_{23}' \\
0 & 0 & a_{33}''
\end{bmatrix}
\]

\[
[A] \xrightarrow{\text{GE}} [A']
\]

\[
det[A] = det[A'] = a_{11} a_{22}' a_{33}'' \quad \text{(can be checked)}
\]

Therefore,

\[
det[A] \Rightarrow \text{product of the diagonal elements of its upper triangular matrix}
\]
Note that swapping of rows of a matrix does not alter values of determinant. However, sign changes: 
\((-1)^k\), where \(k\) is the number of times rows are swapped.

Re-visit

\[
\text{det}[A]_{3\times3} = a_{11}a_{22}'a_{33}'' \quad (\text{product of the diagonal elements of UTM of } A)
\]

**Note:** 1. For a matrix to be non-singular \(\text{det}[A] \neq 0\) or \(a_{11}a_{22}'a_{33}'' \neq 0\)

Alternatively, the corresponding three equations are linearly independent; rank of

\[
[A] = \#\text{of variables} = \#\text{of eqns} = 3.
\]

Geometrically, three planes intersect at a unique point in space, or none of them is \(\parallel\) to the other.

Alternatively,

If \(a_{33}'' = 0\), \(\text{det}[A] = 0 \Rightarrow\) the matrix is singular; rank of the matrix = 2 (< 3); only first two equations are linearly independent; the problem is under-determined and one more equation is required to solve\((x_1, x_2, x_3)\). Algebraically, 3rd plane is \(\parallel\) to one of the other two planes.

**Pivoting and ill-conditioning:**

\[
A\bar{X} = \bar{b} \Rightarrow \bar{X} = A^{-1}\bar{b} ; \quad A^{-1} = \frac{Adj[A]}{|A|}
\]

Therefore, for \(\bar{X}\) to be solved or \(A\) to be invertible, \(|A| \neq 0\). Yet if \(|A| \rightarrow 0\) (or it is a small number), a small change in \(|A|\) will lead to a large change in \(A^{-1}\).

\[\text{Note: } Y = \frac{1}{X}, \quad \frac{dy}{dx} = -\frac{1}{x^2}\]

In other words, a small change in \(X\) will lead to large change in \(Y\).

In such case, the problem is said to be ill-conditioned. To invert such a matrix, large machine precision may be required, and one has to be careful with round-off errors.

Recall, determinant of a matrix is the product of the diagonal elements of UTM. Therefore, pivotal elements should not only be zero, but also large, or the matrix should be “diagonally strong”.


Example: \(0.0001x_1 + x_2 = 1 \quad (A)\)

\[x_1 + x_2 = 2\]

Working with 4th decimal, apply GE with 0.0001 as the pivotal element to obtain
\[X^T = \begin{bmatrix} 0.0000 & 1.0000 \end{bmatrix} \quad \text{which is wrong. Now, swap the equations as}\]
\[x_1 + x_2 = 2\]
\[0.0001x_1 + x_2 = 1\]

Apply GE to obtain \(x^T = \begin{bmatrix} 1.0000 & 1.0000 \end{bmatrix}\) which is 4th decimal accurate.

This is called pivoting. Swap the row, so that the pivotal element is relatively larger. Also, scaling (or multiplying equation (A) by 10000) can be done before applying GE.

Example: Solve by GE:

\[
\begin{bmatrix}
1 & -1 & 2 \\
2 & -2 & 3 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
-20 \\
-2
\end{bmatrix}
\]

Step 1:
\[
\begin{bmatrix}
1 & -1 & 2 \\
0 & 0 & -1 \\
0 & 2 & -1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
-4 \\
+6
\end{bmatrix}
\]

Row pivoting is required (or swap 2nd equation with 3rd so that pivotal elemental is non-zero).

\[
\begin{bmatrix}
1 & -1 & 2 \\
0 & 2 & -1 \\
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
+6 \\
-4
\end{bmatrix}
\]

We have now UTM. Reverse substitution will yield
\[x^T = \begin{bmatrix} -11 & 5 & 4 \end{bmatrix}\]

(a) \(\text{det}[A] = -1 \times 2 \times -1 = 2 \neq 0\) (matrix is singular, rank = 3, all 3 equations are linearly independent). Also, note that we swapped one time. So, there is the multiplication with -1.

Column pivoting (generally it should be avoided)

\[
\begin{bmatrix}
1 & 2 & -1 \\
0 & -1 & 0 \\
0 & -1 & 2
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
-4 \\
+6
\end{bmatrix}
\]
Column 2 is replaced with column 3 to make the pivotal element ≠ 0. However, note that the variables \( x_2 \) and \( x_3 \) are also swapped in the column vector \( \bar{x} \), else you would be solving a different set of equations!

\[
\begin{align*}
\begin{cases}
  x_1 + 2x_2 - x_3 = -8 \\
  -x_2 = -4 \\
  -x_2 + 2x_3 = +6
\end{cases}
\text{(left) original equation} / \begin{cases}
  x_1 + 2x_3 - x_2 = -8 \\
  -x_3 = -4 \\
  2x_2 - x_3 = +6
\end{cases}
\text{(right) swapped equation:}
\end{align*}
\]

Proceed as follows:

\[
\begin{bmatrix}
1 & -2 & -1 \\
0 & -1 & 0 \\
0 & 0 & 2
\end{bmatrix}
\begin{bmatrix}
 x_1 \\
 x_3 \\
 x_2
\end{bmatrix}
= 
\begin{bmatrix}
 -8 \\
 -4 \\
 10
\end{bmatrix}
\]

Reverse substitution will yield \( x^T = [-11, \ 5, \ 4] \)
Lecture #04

Gauss-Jordan

1. Procedure is similar to that of GE

2. Normalize the pivotal element

3. Eliminate the unknown from all rows, below and above the pivot row

4. No back substitution is required as required in GE.

\[
A\bar{X} = \bar{b}
\]

\[
\begin{bmatrix}
 a_{11} & a_{12} & a_{13} \\
 a_{21} & a_{22} & a_{23} \\
 a_{31} & a_{32} & a_{33}
\end{bmatrix}_{3 \times 3}
\begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3
\end{bmatrix} =
\begin{bmatrix}
 b_1 \\
 b_2 \\
 b_3
\end{bmatrix}
\]

Step 1:
\[
a_{1j}' = \frac{a_{1j}}{a_{11}}, \quad j = 1,3 \text{ (normalize } a_{11} \text{ (pivotal element) to 1.0)}
\]

\[
b_1' = b_1 / a_{11}
\]

\[
\begin{aligned}
a_{2j}' &= a_{2j} - a_{1j}' \times a_{21} \\
a_{3j}' &= a_{3j} - a_{1j}' \times a_{31}
\end{aligned}
\]\n
\[
\begin{aligned}
b_2' &= b_2 - b_1' \times a_{21} \\
b_3' &= b_3 - b_1' \times a_{31}
\end{aligned}
\]

same as the GE method; making the first column below the pivotal row to be 0.

\[
\begin{bmatrix}
 1 & a_{12}' & a_{13}' \\
 0 & a_{22}' & a_{23}' \\
 0 & a_{32}' & a_{33}'
\end{bmatrix}_{3 \times 3}
\begin{bmatrix}
 \bar{x}_1 \\
 \bar{x}_2 \\
 \bar{x}_3
\end{bmatrix} =
\begin{bmatrix}
 b_1' \\
 b_2' \\
 b_3'
\end{bmatrix}
\]

Step 2:
\[
a_{2j}'' = \frac{a_{2j}'}{a_{22}'} \quad j = 2,3 \text{ (normalize pivotal element of 2nd row to 1.0)}
\]

\[
b_2'' = b_2' / a_{22}'
\]

\[
\begin{aligned}
a_{1j}'' &= a_{1j}' - a_{2j}'' \times a_{12}' \\
b_1'' &= b_1' - b_2'' \times a_{12}'
\end{aligned}
\]

subtract the modified pivotal row from the row above. \textbf{Note this is the extra step in GE.}

\[
\begin{aligned}
a_{3j}'' &= a_{3j}' - a_{2j}'' \times a_{32}' \\
b_3'' &= b_3' - b_2'' \times a_{32}'
\end{aligned}
\]

subtract the modified pivotal row from the row below. \textbf{This is the same step as in GE.}
\[
\begin{bmatrix}
1 & 0 & a_{13}''
0 & 1 & a_{23}''
0 & 0 & a_{33}''
\end{bmatrix}
\begin{bmatrix}
X_1
X_2
X_3
\end{bmatrix}
= \begin{bmatrix}
b_1'''

b_2'''
b_3'''
\end{bmatrix}
\]

Step 3: \(b_3'''' = b_3'' / a_{33}''\) (normalize pivotal element to 1)

\[
b_2'''' = b_2'''' - b_3'''' \cdot a_{23}'''
\]
\[
b_1'''' = b_1'''' - b_3'''' \cdot a_{13}'''
\]

Subtract both rows 1 and 2 above from pivotal row. There is no row below the pivotal row.

\[
\begin{bmatrix}
1 & 0 & 0
0 & 1 & 0
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
b_1'''

b_2'''
b_3'''
\end{bmatrix}
\]

Therefore, \(X^T = b''''^T\): solution.

No back substitution is required for a unit matrix!

Make a note that G-J method can also be used to determine inverse of the matrix, if working on the augmented matrix. Check text books for more details.

Example:

\[
\begin{bmatrix}
1 & -1 & 2
1 & 1 & 1
2 & -2 & 3
\end{bmatrix}
\begin{bmatrix}
X_1
X_2
X_3
\end{bmatrix}
= \begin{bmatrix}
-8
-2
-20
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & -1 & 2
0 & 2 & -1
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
-8
6
-4
\end{bmatrix}
\]

(pivotal element already had ‘1’ ⇒ no need to normalize!)

\[
\begin{bmatrix}
1 & -1 & 2
0 & 1 & -0.5
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
-8
3
-4
\end{bmatrix}
\]

(pivotal row was divided by 2 to make pivotal element to be 1)

\[
\begin{bmatrix}
1 & 0 & 1.5
0 & 1 & -0.5
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
-5
3
-4
\end{bmatrix}
\]

(rows were subtracted above and below from the pivotal row)

\[
\begin{bmatrix}
1 & 0 & 1.5
0 & 1 & -0.5
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
-5
3
4
\end{bmatrix}
\]

(pivotal element was made to be 1)

\[
\begin{bmatrix}
1 & 0 & 0
0 & 1 & 0
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1
x_2
x_3
\end{bmatrix}
= \begin{bmatrix}
-11
5
4
\end{bmatrix}
\]

(rows 1 & 2 subtracted from the pivotal row)

\[
X^T = [-11 \ 5 \ 4]^T
\]

You should ensure that the programming code for G-J is written by modifying the code for GE, instead of writing afresh, by changing the indices for i and j.
LU Decomposition: Two similar/identical methods

Dolittle ∎ Crout’s

$$A\bar{X} = \bar{b}$$

(This method is considered to be the best if only right hand side $\bar{b}$ changes from problem/case to problem/case)

$$A = LU$$

decompositions into L and U

\[
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}
\begin{bmatrix}
U_{11} & U_{12} & U_{13} \\
0 & U_{22} & U_{23} \\
0 & 0 & U_{33}
\end{bmatrix}
\]

\[
\begin{bmatrix}
l_{11} & 0 & 0 \\
l_{21} & l_{22} & 0 \\
l_{31} & l_{32} & l_{33}
\end{bmatrix}
\begin{bmatrix}
1 & U_{12} & U_{13} \\
0 & 1 & U_{23} \\
0 & 0 & 1
\end{bmatrix}
\]

In either case, steps are as follows:

1. $A = LU$  (decompose)

2. $LU\ \bar{X} = \bar{b}$  \Rightarrow  $L\dd = \bar{b}$,  solve for $\dd$ first

3. $U\bar{X} = \dd$  \Rightarrow  solve for $\bar{X}$

example: \[
\begin{bmatrix}
1 & -1 & 2 \\
1 & 1 & 1 \\
2 & -2 & 3
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3
\end{bmatrix}
= \begin{bmatrix}
-8 \\
-2 \\
-20
\end{bmatrix}
\]

\[
A =
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}_{3 \times 3}
\begin{bmatrix}
U_{11} & U_{12} & U_{13} \\
0 & U_{22} & U_{23} \\
0 & 0 & U_{33}
\end{bmatrix}_{3 \times 3}
\]

\[
= \begin{bmatrix}
U_{11} & U_{12} & U_{13} \\
l_{21}U_{11} & (l_{21}U_{12}) & (l_{12}U_{13}) \\
l_{31}U_{11} & (l_{31}U_{12}) & (l_{12}U_{13})
\end{bmatrix}_{3 \times 3}
\]

\[
= \begin{bmatrix}
U_{11} & U_{12} & U_{13} \\
l_{21}U_{11} & (l_{21}U_{12}) & (l_{12}U_{13}) \\
l_{31}U_{11} & (l_{31}U_{12}) & (l_{12}U_{13})
\end{bmatrix}_{3 \times 3}
\]

Match,

$$U_{11} = 1, \quad U_{12} = -1, \quad U_{13} = 2$$
\[ l_{21}U_{11} = 1, \quad l_{21}U_{12} + U_{12} = 1, \quad l_{21}U_{13} + U_{23} = 1 \]
\[ l_{31}U_{11} = 2, \quad l_{31}U_{12} + l_{32}U_{23} = -2, \quad l_{31}U_{13} + l_{32}U_{23} + U_{33} = 3 \]

Therefore, there are 9 unknown & 9 equations to solve and the problem is well defined.

**How do we obtain LU?**

Well, use GE to reduce \( A \) to an upper triangular matrix \( U \).

However, in addition, store multiplication coefficients of GE as the coefficients \( l_{21}, l_{31}, \text{and} \ l_{32} \) of lower triangular matrix

\[
l_{ik} = \frac{a_{ik}}{a_{kk}}; \quad k = 1 \text{ to } n - 1 \}
\]

\[
i = k + 1 \text{ to } n \}
\]

It is clear that the programming code used for GE to reduce the matrix \( A \) to UTM (U) is the same as that for determining \( U \) of LU decomposition method. In addition, the coefficients of \( L \) are also determined automatically by defining an extra coefficient \( l_{ik} \) (see above) in the same programming code.

Example:

\[
A = \begin{bmatrix}
1 & -1 & 2 \\
1 & 1 & 1 \\
2 & -2 & 3
\end{bmatrix}
\]

**Step 1:**

\[
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix} \begin{bmatrix}
U_{11} & U_{12} & U_{13} \\
0 & U_{22} & U_{23} \\
0 & 0 & U_{33}
\end{bmatrix}
\]

\[
L \quad U
\]

\[
\begin{bmatrix}
l_{21} = 1 & 1 & 0 \\
l_{31} = 2 & l_{32} = 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & -1 & 2 \\
0 & 2 & -1 \\
0 & 0 & -1
\end{bmatrix}
\]

**Note** \( l_{21} \) is the required coefficient to make the coefficient \( U_{12} = 0 \). Similarity, \( l_{31} \) and \( l_{32} \) are determined

\[
L \quad U
\]

Therefore,

\[
\begin{bmatrix}
1 & -1 & 2 \\
1 & 1 & 1 \\
2 & -2 & 3
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
2 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & -1 & 2 \\
0 & 2 & -1 \\
0 & 0 & -1
\end{bmatrix}
\]

A \quad L \quad U
Step 2: \[ L\tilde{d} = \tilde{b} \]

\[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
2 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
d_1 \\
d_2 \\
d_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
-2 \\
-20
\end{bmatrix}
\]

\[ d_1 = -8 \; ; \; d_2 = -2 + 8 = 6 \; ; \; d_3 = -20 + 16 = -4 \]

\[ d^T = [-8 \; 6 \; -4] \]

(Note: you have done forward substitution or solved a LTM, which is just opposite to solving an UTM or reverse/backward substitution step of GE)

Step 3: \[ U\tilde{X} = \tilde{d} \]

\[
\begin{bmatrix}
1 & -1 & 2 \\
0 & 2 & -1 \\
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3
\end{bmatrix} =
\begin{bmatrix}
-8 \\
6 \\
-4
\end{bmatrix}
\]

(Note this is the reverse/backward substitution of GE)

\[ X_3 = 4, \; \; X_2 = (6 + 4)/2 = 5, \; \; X_1 = -8 + 5 - 8 = -11 \]

\[ X^T = [-11 \; 5 \; 4]^T \quad \text{Ans.} \]

You should ensure that you are not writing a fresh code for the LU decomposition method. Rather, you simply modify the code you wrote earlier for the 1st part (forward substitution) of GE, or the code to reduce \( A \) to UTM. Then, write a code for inverting LTM and use the previously written code for the back-substitution or the reverse substitution of GE. In other words, the code of LU decomposition has three sub-parts.
Lecture #05

Matrix Inverse $A^{-1} = I$ (definition for $A^{-1}$)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$, then

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

$$A \cdot A^{-1} = I$$

Making use of the rule of matrix multiplication (row×column), it is clear that

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow A\overrightarrow{b_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow A\overrightarrow{b_2} = [0 \ 1 \ 0]^T$$

$$\begin{bmatrix} b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow A\overrightarrow{b_3} = [0 \ 0 \ 1]^T$$

Therefore, $b_{ij}$ can be calculated from as many equations.

However, LU decomposition can be used to determine $A^{-1}$ as well:

$$A\overrightarrow{b_1} = LU \overrightarrow{b_1} = [1 \ 0 \ 0]^T \quad \overrightarrow{b_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

or $L \overrightarrow{d_1} = [1 \ 0 \ 0]^T \Rightarrow \text{determine } \overrightarrow{d_1}$

So, $U\overrightarrow{b_1} = \overrightarrow{d_1} \Rightarrow \text{determine } \overrightarrow{b_1}$

Similarly,

$$A\overrightarrow{b_2} = LU \overrightarrow{b_2} = [0 \ 1 \ 0]^T$$

or $L \overrightarrow{d_2} = [0 \ 1 \ 0]^T \Rightarrow \text{determine } \overrightarrow{d_2}$
So, $Ub_2 = d_2 \Rightarrow$ determine $b_2$, so forth.

Example: $[A] = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{bmatrix} = LU$

Apply GE (Forward step)

$$\begin{bmatrix} 25 & 5 & 1 \\ 0 & -4.8 & 0.56 \\ 0 & -16.8 & -4.36 \end{bmatrix} \Rightarrow \text{coefficients used to make 1st elements of the rows, bottom of the pivotal row to be zero}$$

$$= \frac{64}{25} \text{ and } \frac{144}{25} \text{ or } 2.56 \text{ and } 5.76$$

This is a UTM.

$$\begin{bmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \end{bmatrix} \Rightarrow \text{coefficients used to make the 1st element of the last row to be zero}$$

$$= \frac{16.8}{4.8} = 3.5$$

Therefore,

$L = \begin{bmatrix} 1 & 0 & 0 \\ 2.56 & 1 & 0 \\ 5.76 & 3.5 & 1 \end{bmatrix}$

or,

$$\begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2.56 & 1 & 0 \\ 5.76 & 3.5 & 1 \end{bmatrix} \begin{bmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & 0 & 0.7 \end{bmatrix}$

$A = L \cdot U$

Now, determine $A^{-1}$

Step 1: $LU \bar{b}_1 = [1 \ 0 \ 0]^T$

$\bar{d}_1$
\[
\begin{bmatrix}
1 & 0 & 0 \\
2.56 & 1 & 0 \\
5.76 & 3.5 & 1
\end{bmatrix}
\begin{bmatrix}
d_1 \\
d_2 \\
d_3
\end{bmatrix} =
\begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\]

\[
L \overline{d_1}
\]

\[
d_1 = 1, \quad d_2 = 0 - 2.56 = -2.56, \quad d_3 = 0 - 5.76 + 3.50 \times 2.56 = 3.2
\]

2.
\[
\begin{bmatrix}
25 & 5 & 1 \\
0 & -4.8 & -1.56 \\
0 & 0 & 0.7
\end{bmatrix}
\begin{bmatrix}
b_{11} \\
b_{21} \\
b_{31}
\end{bmatrix} =
\begin{bmatrix}
d_1 \\
d_2 \\
d_3
\end{bmatrix}
\]

\[
U \overline{b_1}
\]

\[
b_{31} = \frac{3.2}{0.7} = 4.571, \quad b_{21} = \frac{-2.56 - 4.571 \times (-1.56)}{-4.8} = -0.9524
\]

\[
b_{11} = \frac{(1 - 4.571 - 5 \times (-0.9524))}{25.0} = 0.04762
\]

or, \( \overline{b_1^T} = [0.04762 \quad -0.9524 \quad 4.571]^T \)

3.
\[
\begin{bmatrix}
1 & 0 & 0 \\
2.56 & 1 & 0 \\
5.76 & 3.5 & 1
\end{bmatrix}
\begin{bmatrix}
d_1 \\
d_2 \\
d_3
\end{bmatrix} =
\begin{bmatrix}
0 \\
1 \\
0
\end{bmatrix} \Rightarrow d_1 = 0, \quad d_2 = 1, \quad d_3 = -3.5
\]

\[
L \overline{d_1}
\]

\[
\begin{bmatrix}
25 & 5 & 1 \\
0 & -4.8 & -1.56 \\
0 & 0 & 0.7
\end{bmatrix}
\begin{bmatrix}
b_{12} \\
b_{22} \\
b_{32}
\end{bmatrix} =
\begin{bmatrix}
0 \\
1 \\
-3.5
\end{bmatrix}
\]

\[
U \overline{b_2}
\]

\[
\overline{b_2^T} = [-0.0833 \quad 1.417 \quad -5.0]^T
\]

Similarly,
\[
\overline{b_3^T} = [0.03571 \quad -0.4643 \quad 1.429]^T
\]

\[
A^{-1} =
\begin{bmatrix}
0.04762 & -0.0833 & 0.03571 \\
-0.9524 & 1.417 & -0.4643 \\
4.571 & -0.5 & 1.429
\end{bmatrix}
\]

\( \Rightarrow \text{Methods so far to solve } A\overline{X} = \overline{b} \text{ and programming codes:} \)

(a) GE: To solve a large number of algebraic equations, a programming code is required. As earlier shown, GE contains two distinct steps: (1) forward elimination to convert \( \overline{A} \) to an upper triangular matrix (UTM), (2)
Backward/reverse substitution to solve UTM. Therefore, while writing the code; it is recommended that one clearly distinguishes the two steps as “subroutines” (# 1 and 2). You will later see that the other methods also often require one or both of the steps/codes.

(b) GJ: The programming code for GJ is similar to that (code 1) of the forward elimination of GE, with some modification. The second part (code) of GE, ie. reverse substitution is not required. The modification or extra step is simple. First, pivotal elements should be normalized to 1; in addition to subtracting the modified pivotal row from the rows below the ‘ith’ row of GE, the pivotal row is also subtracted from the rows above. Therefore, one extra line is included in the code for GJ.

(c) The code to determine the determinant of the matrix A is the same as that (code 1) for converting A to UTM, ie. the first or forward elimination step of GE. Further, add a line to determine the $\prod_{i=1}^{n} a_{ii}$ product of the diagonal elements of the UTM ie. one can put the flag to check if any of the $a_{ii}$ is zero, the matrix is singular and cannot be inverted. One can also write a simple code to check the number # of zeros on the diagonal elements and therefore, determine the rank of the matrix as $(n - #)$.

(d) LU method: The programming code (# 3) is similar to that (code 1) of the forward step of GE, to determine UTM. An extra line in the programming loop is required to store the coefficients $l_{ik} = \left( \frac{a_{ik}}{a_{kk}} \right)$, used to make the elements of the first column of the UTM to be zero, as the elements of the empty columns of the LTM.

$$A = \begin{bmatrix} l_{11} & \cdots & \cdots \\ l_{21} & l_{22} & \cdots \end{bmatrix} \times \begin{bmatrix} - & - & - \\ & & \end{bmatrix}$$

LTM     UTM

(e) **Inverse of the matrix:** First, the programming code (# 3) of LU is to be used. The forward substitution on LTM will yield/give $\vec{d}$ (column matrix) and the backward substitution on UTM will yield $\vec{b}$ (column matrix) (already available; # 2). Therefore, one new code/subroutine (# 4) is required for the forward substitution on LTM, which is not different from that (# 2) for the reverse substitution on UTM:

- $A = LU$ (code # 3)
- *Forward substitution* $\rightarrow \vec{d}$ (code #4)
- *Backward substitution* $\rightarrow \vec{b}$ (code # 2)
Lecture #06

Thomas Algorithm (Tridiagonal matrix)

\[ A\bar{X} = \bar{b} \]

\[
\begin{bmatrix}
    a_{11} & a_{12} & 0 & \cdots & 0 & 0 & 0 \\
    a_{21} & a_{22} & a_{23} & \cdots & 0 & 0 & 0 \\
    0 & a_{32} & a_{33} & a_{34} & \cdots & 0 & 0 \\
    0 & 0 & a_{43} & a_{44} & a_{45} & \cdots & 0 \\
    0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & 0 & 0 & \cdots & a_{nn} \\
\end{bmatrix}
\begin{bmatrix}
    X_1 \\
    X_2 \\
    X_3 \\
    \vdots \\
    X_{n-1} \\
    X_n \\
\end{bmatrix} = 
\begin{bmatrix}
    b_1 \\
    b_2 \\
    b_3 \\
    \vdots \\
    b_{n-1} \\
    b_n \\
\end{bmatrix}
\]

Tridiagonal matrix has 3 non-zero elements in all of its rows, except in the 1st and last row, with one element each on the left and right of the diagonal element. The 1st and last rows have one element on the right and left of the diagonal elements, respectively. It is a banded matrix around its diagonal:

Therefore, a tridiagonal matrix can be represented using single subscripted indices for its elements:

\[
\begin{bmatrix}
    b_1 & c_1 & 0 & 0 & \cdots & 0 \\
    a_2 & b_2 & c_2 & 0 & \cdots & 0 \\
    0 & a_3 & b_3 & c_3 & \cdots & 0 \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & 0 & \cdots & a_n \\
\end{bmatrix}
\]

Note that the element in the \(i^{th}\) row is represented as \((a_i, b_i, c_i)\). All \(b\) are on the diagonal.

Therefore, \(A\bar{X} = \bar{d}\), where \(A\) is a tridiagonal matrix and it represents the following set of linear algebraic equations:
\[
\begin{align*}
&b_1 x_1 + c_1 x_2 = d_1 \\
&a_2 x_1 + b_2 x_2 + c_2 x_3 = d_2 \\
&a_3 x_2 + b_3 x_3 + c_3 x_4 = d_3 \\
&a_4 x_3 + b_4 x_4 + c_4 x_5 = d_4 \\
&\vdots \\
&a_n x_{n-1} + b_n x_n = d_n
\end{align*}
\]

⇒ You must have noted that for the \(i^{th}\) row, \(b_i\) is the diagonal element multiplied with the variable \(x_i\) in the same row, whereas \(a_i\) is multiplied with \(x_{i-1}\)(the variable above \(i^{th}\) row) and \(c_i\) is multiplied with \(x_{i+1}\)(the variable below \(i^{th}\) row). Naturally, the first and last rows have only two variables.

⇒ (Tridiagonal system is quite common when using Finite Difference 2\(^{nd}\) order method to solve boundary value problems or partial differential eqns.)

**Thomas algorithm to solve such system:**

Step 1:

\(x_1\) is eliminated : \((b_2 - \frac{c_2}{b_1} a_2) x_2 + c_2 x_3 = (d_2 - \frac{d_1}{b_1} a_2)\)

(by dividing 1\(^{st}\) row with \(b_1\), multiplying with \(c_1\) and subtracting from row 2)

\(x_2\) is eliminated : \((b_3 - \frac{c_2}{b_2} a_3) x_3 + c_3 x_4 = (d_3 - \frac{d_2}{b_2} a_3)\)

\vdots

\(x_{k-1}\) is eliminated : \((b_k - \frac{c_k-1}{b_{k-1}} a_k) x_k + c_k x_{k+1} = (d_k - \frac{d_{k-1}}{b_{k-1}} a_k)\)

It is clear that,

\[
\begin{align*}
\text{All } a_s \text{ are eliminated} \\
\text{All } b_s \text{ are modified} \\
\text{All } c_s \text{ are unaltered} \\
\text{All } d_s \text{ are modified}
\end{align*}
\]

Last row:  \(b_n = b_n - \frac{c_{n-1}}{b_{n-1}} a_n; \quad d_n = d_n - \frac{d_{n-1}}{b_{n-1}} a_n\)

(Note that \(a_n\) has moved to RHS, leaving behind \(b_n x_n\) only on the LHS)
(It is important to note that $k^{th}$ row uses the latest modified values of $b_k$ and $d_k$ from the previous (k-1) step. Therefore, there is no need to store the previous values of $(a, b, d)$ while writing the code)

**Back-substitution:**

$$X_n = \frac{d_n}{b_n} \quad \text{(nth row has only $b_n$ and $d_n$)}$$

$$X_{n-1} = \frac{d_{n-1} - c_{n-1}X_n}{b_{n-1}} \quad \text{(n - 1 row has $b_{n-1}, c_{n-1},$ and $d_{n-1}$)}$$

⇒ A pseudo programming code can be written as follows:

**Tridiagonal (N, a, b, c, d, X)**

```plaintext
do \hspace{1mm} i = 1, N
   a(i) =
   b(i) =
   c(i) =
   d(i) =
end do

do \hspace{1mm} i = 2, N

   \begin{align*}
   b(i) &= b(i) - a(i) \frac{c(i - 1)}{b(i - 1)} \quad \text{forward substitution} \\
   d(i) &= d(i) - a(i) \frac{d(i - 1)}{b(i - 1)}
   \end{align*}

end do

$$X_n = \frac{d(n)}{b(n)}$$

**do (i) = N - 1, 1, -1**

$$X_i = \frac{d(i) - c(i)X_{i+1}}{b(i)} \quad \text{reverse substitution}$$

end do
Indirect Methods

GE, GJ, LU decomposition are the direct methods to solve $A\bar{X} = \bar{b}$. Simple iterations can also be done to solve a set of algebraic equations, Jacobi and Gauss-Seidel being the two commonly used indirect methods.

Ex. \[
\begin{bmatrix}
2 & 1 & 0 \\
1 & 2 & 1 \\
0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
\end{bmatrix} = \begin{bmatrix}
1 \\
2 \\
4 \\
\end{bmatrix}
\]

Make a guess $X_1^{(1)}, X_2^{(1)}, X_3^{(1)}$

**Jacobi**

\[
X_1^{(2)} = \frac{1-X_2^{(1)}}{2}
\]
\[
X_2^{(2)} = \frac{2-X_1^{(1)}-X_3^{(1)}}{2}
\]
\[
X_3^{(2)} = \frac{4-X_2^{(1)}}{1}
\]

Takes all $X_s^{(k-1)}$ of the previous iteration

**Gauss-Seidel**

\[
X_1^{(2)} = \frac{1-X_2^{(1)}}{2}
\]
\[
X_2^{(2)} = \frac{2-X_1^{(1)}-X_3^{(1)}}{2}
\]
\[
X_3^{(2)} = \frac{4-X_2^{(1)}}{1}
\]

Uses the most latest iterated values of $X_s$.

In general: $A\bar{X} = \bar{b}$

**Jacobi:**

\[
X_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{n} a_{ij} X_j^{(k)}}{a_{ii}(\neq 0)} \quad (j \neq i)
\]

**G-S:**

\[
X_i^{(k+1)} = \left\{ \begin{array}{l}
\text{updated} \\
\text{latest} \\
\text{values}
\end{array} \right\} - \sum_{j=i+1}^{n} a_{ij} X_j^{(k)} \quad \sum_{j=1}^{i-1} a_{ij} X_j^{(k)}
\]

\[
\text{old} \\
\text{values}
\]

\[
a_{ii}(\neq 0)
\]
At any $i^{th}$ row:

\[
\begin{array}{cccccc}
1 & \cdots & i-1 & i & i+1 & \cdots & N \\
\text{updated values} & & & & & & \text{old values} \\
(k+1) & & & & & & (k)
\end{array}
\]

Schematically,

G-S (modified)

\[
X_i = \lambda X_i^{(new)} + (1 - \lambda) X_i^{(old)}
\]

\[
\lambda = 1 \quad \text{(unmodified)}
\]

\[
0 < \lambda < 1 \Rightarrow \text{under} - \text{relaxation factor}
\]

\[
1 < \lambda < 2 \Rightarrow \text{over} - \text{relaxation factor}
\]

('\lambda' is often assumed depending upon guesses, etc)

Look differently! : LDU method
\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & & \vdots \\
a_{n1} & \cdots & \cdots & a_{nn}
\end{bmatrix}_{n \times n} = \begin{bmatrix}
0 & \cdots & \cdots & \cdots \\
a_{21} & 0 & \cdots & \cdots \\
a_{n1} & \cdots & \cdots & a_{nn}
\end{bmatrix} + \begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{bmatrix} + \begin{bmatrix}
a_{11} & \cdots & \cdots & a_{nn}
\end{bmatrix}
\]

strictly lower triangular matrix \((L)\) + diagonal elements only \((D)\) + strictly upper triangular matrix \((U)\)

\[
A = L + D + U
\]

\[
A\bar{X} = \bar{b}
\]

\[
(L + D + U)\bar{X} = \bar{b} \Rightarrow D\bar{X} = \bar{b} - L\bar{X} - U\bar{X}
\]

or \[
\bar{X} = \frac{\bar{b} - L\bar{X} - U\bar{X}}{D}
\]

Jacobi:

\[
\bar{X}^{(k+1)} = \frac{\bar{b} - (L + U)\bar{X}^{(k)}}{D}
\]

or \[
X_i^{(k+1)} = b_i - \sum_{j=1}^{n} X_j^{(k)} a_{ij} a_{ii} \quad (j \neq i \text{ (} j = i \text{ will represent diagonal element)})
\]

\[
G - S : (L + D)\bar{X} = \bar{b} - U\bar{X}
\]

\[
(L + D)\bar{X}^{(k+1)} = \bar{b} - U\bar{X}^{(k)}
\]

\[
D\bar{X}^{(k+1)} = \bar{b} - U\bar{X}^{(k)} - L\bar{X}^{(k+1)}
\]

\[
X_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} X_i^{(k+1)} a_{ij} - \sum_{j=i+1}^{n} X_i^{(k)} a_{ij}}{a_{ii}} \quad \text{(same as before) \quad (} a_{ii} \neq 0 \text{)}
\]

Both methods will lead to the same solutions, with different # of iterations.
Lecture #07

Homogeneous linear algebraic equations

- Represents a special class of problems, also known as the Eigenvalue or Characteristic type of problems

- Mathematically represented as \( A\vec{x} = 0 \) or \( \vec{b} = 0 \) (null vector)

\[
\begin{align*}
\left( a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0 \right) \\
\vdots \\
\left( a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = 0 \right)
\end{align*}
\]

Naturally, a trivial solution is \( \vec{x} = \vec{0} \). In the simplest geometrical term, two straight lines or three planes intersect at the origin:

\[
\begin{align*}
x_1 - x_2 &= 0 \\
x_1 - 0.3x_2 &= 0
\end{align*}
\]

\( \Rightarrow \) A more interesting problem to solve is in seeking a non-trivial solution when \( \text{det}(A) = 0 \) or \( r < m \). Such simultaneous set of homogeneous linear equations when \( \text{det}(A) = 0 \) is common in several engineering and mechanics applications, and also in the initial and boundary value problems. Such situation is better known as the eigenvalue problem and is represented by eigenvalues and the corresponding eigenvectors.

Compare this situation to the earlier discussed set of non-homogeneous linear algebraic equation \( A\vec{X} = \vec{b} \), where we sought a unique solution, when \( \text{det}|A| \neq 0 \). Comparatively,

<table>
<thead>
<tr>
<th>Non-homogeneous</th>
<th>Homogeneous equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A\vec{X} = \vec{b} )</td>
<td>( A\vec{X} = 0 )</td>
</tr>
<tr>
<td>( \text{det}(A) \neq 0 )</td>
<td>( \text{det}(A) = 0 )</td>
</tr>
<tr>
<td>( r = m )</td>
<td>( r &lt; m )</td>
</tr>
<tr>
<td>Unique solution</td>
<td>Non-trivial solution</td>
</tr>
</tbody>
</table>

**Eigenvalue problem**
Let us look at

\[ A\vec{X} = \lambda\vec{X} \]  \( \text{1} \) where \( A \) is the coefficient matrix and \( \lambda \) is a non-zero number.

matrix scalar quantity

or \[ (A - \lambda I)\vec{X} = 0 \]  \( \text{2} \) where \( I \) is an identity matrix

\[ I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Equation 1 or 2 represents eigenvalue problem and such equations commonly occur in several dynamic studies of distillation, adsorption and CSTR, viz. in solving \( \frac{d\vec{Y}}{dt} = 0 \). Equation (2) can also be written as

\[
\begin{align*}
(a_{11} - \lambda)X_1 + a_{12}X_2 + \ldots + a_{1n}X_n &= 0 \\
a_{21}X_1 + (a_{22} - \lambda)X_2 + \ldots + a_{2n}X_n &= 0 \\
&\vdots \\
a_{n1}X_1 + a_{n2}X_2 + \ldots + (a_{nn} - \lambda)X_n &= 0
\end{align*}
\]

or

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{bmatrix}
= \lambda
\begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{bmatrix}
\]

A(coefficient matrix) column vector \( \lambda \) a scalar column quantity vector

Re-visit \[ A\vec{X} = \lambda\vec{X} \] - eigenvalue problem

Most vectors \( \vec{X} \) will not satisfy such an equation. A common vector \((iX_1 + jX_2)\) will change direction and magnitude to \((ib_1 + jb_2)\) on the transformation by \( A \). Only certain special vectors \( \vec{X} \), called eigenvectors, corresponding to only special numbers, \( \lambda \) (either +ve or -ve), called eigenvalues, will satisfy the above equation. In such case, the eigenvector \( \vec{X} \) does not change its direction or does not rotate when transformed by the coefficient matrix \( A \), but is only scaled by the eigenvalue \( \lambda \). See the geometrical representation below:
It turns out that \((n \times n)\) matrix \(A\) will give \(n\) \(\lambda\) \((\text{eigenvalues})\), and each eigenvalue will give \(n\) eigenvectors that will also be linearly independent. As an example,

\[
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}_{2 \times 2}
\text{will have 2 } \lambda_s \text{ (} \lambda_1 \text{ & } \lambda_2 \text{)} \Rightarrow \\
\lambda_1 \rightarrow \{X\}_1 \text{ & } \lambda_2 \rightarrow \{X\}_2
\]

Both vectors \(\{X\}_1\) & \(\{X\}_2\) will satisfy \(A\bar{X} = \lambda\bar{X}\). They will also be linearly independent so that they can be used as a basis for the space-description. They may be orthogonal or non-orthogonal.

In the figure above, dotted and solid lines represent two vectors scaled by the respective eigenvalues \(1\) and \(2\), respectively.

Similarly, \((3 \times 3)\) \(A\) matrix will give \(3\lambda\) \((\lambda_1, \lambda_2, \lambda_3)\). Each \(\lambda\) will give three independent vectors \(\{X\}_1, \{X\}_2, \{X\}_3\) satisfying \(A\bar{X} = \lambda\bar{X}\)
or \((A - \lambda I)\vec{X} = 0\)

Linear algebra tells us that for \(\vec{X}\) to have a non-trivial solution \(\det|(A - \lambda I)| = 0\).

**Example:** \(A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\)

\(AX = \lambda X\)

\[
\det \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} = 0 \Rightarrow (2 - \lambda)^2 - 1 = 0
\]

\(\lambda_1 = 1, 3\)

\(\lambda_1 = 1:\)

\[
\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 1 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}
\]

Both equations yield \(X_1 = -X_2\) or \(X_1 + X_2 = 0\) \((r = 1)\)

So, \(\vec{X}\) \((\lambda_1 = 1)\):

\((i - j) \equiv \begin{bmatrix} 1 \\ -1 \end{bmatrix}\)

\(\lambda_2 = 3:\)

\[
\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 3 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}
\]

Both equations yield \(X_1 = X_2\) or \(X_1 - X_2 = 0\) \((r = 1)\)

\(\vec{X}\) \((\lambda_2 = 3)\):

\((i + j) \equiv \begin{bmatrix} 1 \\ 1 \end{bmatrix}\)

Note two vectors \((i - j)\) and \((i + j)\) are linearly independent and can be used as a basis for defining 2D space. They need not be orthogonal! They can be used as a linear combination of any other vector \(\vec{B}\) in 2D space:

\[
\vec{B} = c_1(i + j) + c_2(i - j)
\]
System of ODEs

It may represent dynamics, i.e. how a system behaves due to a perturbation around steady state value. Such situation is common in kinetics and the heat and mass transfer related problems in chemical engineering.

\[
\begin{align*}
\frac{dx_1}{dt} &= f_1(X_1, ..., X_n) \\
\frac{dx_2}{dt} &= f_2(X_1, ..., X_n) \\
& \vdots \\
\frac{dx_n}{dt} &= f_n(X_1, ..., X_n)
\end{align*}
\Rightarrow
\]

\[
\frac{d\bar{x}}{dt} = \bar{f}(\bar{X}) \quad [\bar{X}]^T \equiv [X_1, ..., X_n]
\]

Under SS, \(\bar{f}(\bar{X}) = 0\) \(\Rightarrow\) \(\bar{X} = \bar{X}_s\)

or

\[
\begin{align*}
f_1(X_1, ..., X_n) &= 0 \\
f_2(X_1, ..., X_n) &= 0 \\
& \vdots \\
f_n(X_1, ..., X_n) &= 0
\end{align*}
\]

These sets of algebraic equations may represent energy or species balance under SS.

Define, \(\xi(t) = \bar{X}(t) - \bar{X}_s\) (deviation/departure/error variables)

Element-wise

\[
\xi_1(t) = X_1(t) - X_{1s}
\]

or

\[
\xi_i(t) = X_i(t) - X_{is} \quad (i = 1, ..., n)
\]

or, \(\frac{d\xi_i}{dt} = \frac{dX_i(t)}{dt} = \bar{f}(\bar{X})\)

or, \(\frac{d\xi_i}{dt} = f_i(X_1, ..., X_n) \quad (i = 1, ..., n)\)

\[
= f_i(X_{1s} + \xi_1, X_{2s} + \xi_2, ..., X_{ns} + \xi_n)
\]

\[
= f_i(X_{1s}, X_{2s}, X_{3s}, ..., X_{ns}) + \frac{\partial f_i}{\partial x_1} \bigg|_{X_{1s}} \xi_1 + \frac{\partial f_i}{\partial x_2} \bigg|_{X_{2s}} \xi_2 + \cdots + \frac{\partial f_i}{\partial x_n} \bigg|_{X_{ns}} \xi_n
\]
\[ \xi_1 = (X_1 - X_{1s}), \quad \xi_2 = (X_2 - X_{2s}), \ldots, \xi_n = (X_n - X_{ns}) \]

(Taylor series expansion for multivariables: retain 1st order terms only, i.e. small departure from SS)

\[
\begin{bmatrix}
\frac{d\xi_1}{dt} \\
\vdots \\
\frac{d\xi_n}{dt}
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \cdots & \frac{\partial f_1}{\partial X_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial X_1} & \frac{\partial f_n}{\partial X_2} & \cdots & \frac{\partial f_n}{\partial X_n}
\end{bmatrix}
\begin{bmatrix}
\xi_1 \\
\xi_2 \\
\vdots \\
\xi_n
\end{bmatrix}
\]

(row \times column multiplication rule)

Note: For two variables \( X_1 \) and \( X_2 \)

\[
\begin{align*}
\frac{d\xi_1}{dt} &= \frac{\partial f_1}{\partial X_1} \bigg|_{X_{1s}} (X_1 - X_{1s}) + \frac{\partial f_1}{\partial X_2} \bigg|_{X_{2s}} (X_2 - X_{2s}) \\
\frac{d\xi_2}{dt} &= \frac{\partial f_2}{\partial X_1} \bigg|_{X_{1s}} (X_1 - X_{1s}) + \frac{\partial f_2}{\partial X_2} \bigg|_{X_{2s}} (X_2 - X_{2s})
\end{align*}
\]

\[
\begin{bmatrix}
\frac{d\xi_1}{dt} \\
\frac{d\xi_2}{dt}
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} \\
\frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2}
\end{bmatrix}
\begin{bmatrix}
X_1 - X_{1s} = \xi_1 \\
X_2 - X_{2s} = \xi_2
\end{bmatrix}
\]

or,

\[
\frac{d\vec{\xi}}{dt} = A\vec{\xi} \quad \text{or} \quad \vec{\xi}' = A\vec{\xi}
\]

\[
A = \begin{bmatrix}
\frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial X_1} & \cdots & \frac{\partial f_n}{\partial X_n}
\end{bmatrix}; \quad \text{A is called the Jacobian matrix of } f.
\]

Thus, we started from

\[
\frac{dX}{dt} = \vec{f}(X)
\]

And derived:

\[
\frac{d\vec{\xi}}{dt} = A\vec{\xi}
\]

where \( \vec{\xi} = (\vec{X} - \bar{X}_s) \), and \( A \) is the Jacobian matrix of \( \vec{f} \) as

\[
A = \begin{bmatrix}
\frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial X_1} & \cdots & \frac{\partial f_n}{\partial X_n}
\end{bmatrix}
\]
Seeking solution to \( \frac{d\bar{\xi}}{dt} = A\bar{\xi} \) or \( \bar{\xi}' = A\bar{\xi} \)

Let us assume, \( \bar{\xi}_i = z_i e^{\lambda t}, \quad i = 1 \ldots n \) (solution)

(This is the most plausible solution to a simple ODE)

or, \( \bar{\xi} = \bar{z} e^{\lambda t} \quad (\lambda = scalar, \quad \bar{z} = constant \ vector) \)

Substitute,
\[
\begin{align*}
z_i \lambda e^{\lambda t} &= a_{i1} z_1 e^{\lambda t} + a_{i2} z_2 e^{\lambda t} + \cdots + a_{in} z_n e^{\lambda t} \\
(i &= 1 \cdots n)
\end{align*}
\]

or,
\[
\begin{align*}
a_{11} z_1 + a_{12} z_2 + \cdots + a_{1n} z_n &= \lambda z_1 \\
\vdots \\
a_{n1} z_1 + a_{n2} z_2 + \cdots + a_{nn} z_n &= \lambda z_n
\end{align*}
\]

\( (\sum_{j=1}^{n} a_{ij} z_j = \lambda z_i; \quad i = 1, \ldots n) \)

Thus, we have got a set of homogeneous equations which describes the characteristics of the system dynamics AND this equation can represent an eigenvalue problem. In summary, we started from

\[
\begin{align*}
\frac{d\bar{X}}{dt} &= \bar{f}(\bar{X}) \\
\frac{d\bar{\xi}}{dt} &= A\bar{\xi}
\end{align*}
\]

where, \( A = Jacobian \ matrix \ of \ \bar{f} \) and \( \bar{\xi} = \bar{z} e^{\lambda t} \)

Therefore, for a non-trivial solution (eigenvalue problem)

\[
\det|A - \lambda I| = 0 \quad (r < m)
\]
or \[ \det \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} - \lambda \end{vmatrix} = 0 \]

Solve for \( n \lambda_s \) (eigenvalues) from the \( n^{th} \) degree polynomial on \( \lambda' \) after you have expanded the \((n \times n) |A - \lambda I|\) matrix to determine its determinant. Recall, every \( \lambda_i (i = 1 \ldots n) \) will give one \( \bar{Z}^{(i)} \) eigenvector so that \( A \bar{Z}^{(i)} = \lambda_i \bar{Z}^{(i)} \). In other words,

\[
\begin{align*}
\lambda_1 & \rightarrow \bar{Z}^{(1)} e^{\lambda_1 t} \\
\lambda_2 & \rightarrow \bar{Z}^{(2)} e^{\lambda_2 t} \\
\vdots & \\
\lambda_n & \rightarrow \bar{Z}^{(n)} e^{\lambda_n t}
\end{align*}
\]

And all these eigenvectors are linearly independent

Therefore, we have a general solution:

\[
\bar{\xi} = c_1 \bar{Z}^{(1)} e^{\lambda_1 t} + c_2 \bar{Z}^{(2)} e^{\lambda_2 t} + \cdots + c_n \bar{Z}^{(n)} e^{\lambda_n t}
\]

where, \( c_1, c_2 \ldots c_n \) are arbitrary constants to be determined from initial conditions.

\[
\begin{pmatrix}
\xi_1 \\
\xi_2 \\
\vdots \\
\xi_n
\end{pmatrix} = c_1 e^{\lambda_1 t} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}^{(1)} + c_2 e^{\lambda_2 t} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}^{(2)} + \cdots + c_n e^{\lambda_n t} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}^{(n)}
\]

(Note \( \bar{\xi} = \bar{X}_s + \bar{\xi} \))

Numerical values of \( \lambda' \) decides how solution will behave i.e. error \( \xi \) or perturbation or deviation from SS will grow or decay. If the real part of \( \lambda \) is \( -ve' \), \( y \rightarrow y_s \) or solution will decay; if \( +ve' \) solution will grow ⇒ (unstable)

Example: Study the dynamics

\[
\begin{align*}
X_1' &= -3X_1 + 2X_2 \\
X_2' &= X_1 - 2X_2
\end{align*}
\]

\( \textbf{Note:} \) If there is a non – homogeneous part in the equations, ignore it because only homogenous parts contribute to the growth or decay of the solution.

or \( \frac{d\bar{X}}{dt} = A\bar{X} \), \( A = \begin{bmatrix} -3 & 2 \\ 1 & -2 \end{bmatrix} \)

Determine \( \lambda_s \): \( \det \begin{bmatrix} -3 - \lambda & 2 \\ 1 & -2 - \lambda \end{bmatrix} = \lambda^2 + 5\lambda + 4 = 0 \) (characteristic polynomial)

\( \lambda_1 = -1, \quad \lambda_2 = -4 \) (two unique solutions or eigenvalues)
Determine corresponding eigenvectors:

\[ \lambda_1 = -1: \quad \begin{bmatrix} -3 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = -1 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \]

or \[ -3X_1 + 2X_2 = -X_1 \]

and \[ X_1 - 2X_2 = -X_2 \] \( \Rightarrow X_1 = X_2 \) \( \Rightarrow \bar{X}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \) \( (r = 1 < m) \)

\[ \lambda_2 = -4 \quad X_1 = -2X_2 \quad \Rightarrow \bar{X}^{(2)} = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \]

Check \( \bar{X}_1^{(1)} \) and \( \bar{X}^{(2)} \) are linearly independent.

\[ \Rightarrow \text{Note: both } \lambda \text{ are } -ve. \text{ Therefore, the solutions will decay to the SS values or perturbation will die/decay, or solutions will not grow, or the systems will be stable.} \]

General solution:

\[ \bar{X} = C_1 \bar{X}^{(1)} e^{\lambda_1 t} + C_2 \bar{X}^{(2)} e^{\lambda_2 t} \]

or \[ \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = C_1 e^{-t} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + C_2 e^{-4t} \begin{bmatrix} -2 \\ 1 \end{bmatrix} \]

or \[ X_1 = C_1 e^{-t} - 2C_2 e^{-4t} \]

\[ X_2 = C_1 e^{-t} + C_2 e^{-4t} \]

(Note: For repeat or complex \( \lambda \), check the books for the methods to determine \( \bar{X} \))

Ex. \[ [A] = \begin{bmatrix} 0 & 2 & 3 \\ -10 & -1 & 2 \\ -2 & 4 & 7 \end{bmatrix} \] determine all \( \lambda \) and \( \bar{X} \).

From \( \text{det}[A - \lambda I] = 0 \) \( \Rightarrow \lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3 \)

(A 3 \times 3 matrix will give 3\( \lambda \))

\[ \lambda_1 = 1 \quad [A - \lambda I] \bar{X} = 0 \Rightarrow \begin{bmatrix} -1 & 2 & 3 \\ -10 & -2 & 2 \\ -2 & 4 & 6 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = 0 \]

Apply Gauss Elimination:

\[ \begin{bmatrix} -1 & 2 & 3 \\ 0 & -22 & -28 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = 0 \]
You will get only 2 independent eqns: \((\text{rank } r = 2 < m, \det() = 0)\)

\[
\begin{align*}
X_1 &= 2X_2 + 3X_3 \\
-22X_2 - 28X_3 &= 0
\end{align*}
\]

\(\Rightarrow\) choose \(X_3 = 1\)

\[
\begin{align*}
X_1 &= 5/11 \\
X_2 &= -14/11
\end{align*}
\]

Therefore,

\[
\bar{X}^{(1)} = \begin{bmatrix} 5/11 \\ -14/11 \\ 1 \end{bmatrix}
\]

Similarly for

\(\lambda_2 = 2:\)

\[
[\bar{X}^{(2)}]^T = \begin{bmatrix} 1/2 & -1 & 1 \end{bmatrix}
\]

\(\lambda_3 = 3:\)

\[
[\bar{X}^{(3)}]^T = \begin{bmatrix} 1/2 & -3/4 & 1 \end{bmatrix}
\]

**Note:** These are all linearly independent vectors \((\bar{X}_0^{(1)} \neq C_1 \bar{X}^{(2)} + C_2 \bar{X}^{(3)}).\) They may not be orthogonal.

**Spring-problem (2nd order ODEs can also describe/represent an eigenvalue problem)**

Force balance or mass \(m_1\) and \(m_2\), when the two blocks are displaced by \(X_1\) & \(X_2\) in the right direction from its original position:
\[ m_1 \frac{d^2 X_1}{dt^2} = -K X_1 + K (X_2 - X_1) \]
\[ m_2 \frac{d^2 X_2}{dt^2} = -K (X_2 - X_1) - K X_2 \]

\[ X_i = A_i \sin(\omega t) \quad \text{(assume a solution)} \]

( or \( \vec{X} = \vec{A} \sin(\omega t) \))

Therefore,

\[ X_i'' = -A_i \omega^2 \sin \omega t \]

Substitute,

\[
\begin{cases}
\left( \frac{2K}{m_1} - \omega^2 \right) A_1 - \frac{K}{m_1} A_2 = 0 \\
- \frac{K}{m_2} A_1 + \left( \frac{2K}{m_2} - \omega^2 \right) A_2 = 0
\end{cases}
\]

or

\[
\begin{vmatrix}
\left( \frac{2K}{m_1} - \omega^2 \right) & - \frac{K}{m_1} \\
- \frac{K}{m_2} & \left( \frac{2K}{m_2} - \omega^2 \right)
\end{vmatrix}
\begin{bmatrix}
A_1 \\
A_2
\end{bmatrix} = 0
\]

Therefore, the characteristic equation describes an eigenvalue problem (or spring dynamics is an eigenvalue problem). It has assumed the form of

\[ (A - \omega^2 I) \vec{X} = 0 \]

where \( \omega^2 = \lambda \)

\[ A = \begin{bmatrix}
\frac{2K}{m_1} & - \frac{K}{m_1} \\
- \frac{K}{m_2} & \frac{2K}{m_2}
\end{bmatrix} \]

and \( \vec{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \)

\text{eigenvalue} \quad \text{eigenvector}

Example (From Kreysig's Math book)

\[ m_1 = m_2 = 40 \text{ kg} \; ; \; K = 200 \text{ N/m} \]

For a non-trivial solution \( \det(A - \omega^2 I) \vec{X} = 0 \)

\text{eqns:} \quad (10 - \omega^2)A_1 - 5A_2 = 0 \\
\quad -5A_1 + (10 - \omega^2)A_2 = 0 \]
$\text{det}(\cdot) : (10 - \omega^2)^2 - 25 = 0$

$\lambda_1 = 15, \quad \lambda_2 = 5 \quad \text{or} \quad \omega_1 = \sqrt{\lambda_1}, \quad \omega_2 = \sqrt{\lambda_2}$

$\lambda_1 = 15 \quad A_1 = -A_2 \quad \text{or} \quad A^{(1)} = \left\{ \begin{array}{c} 1 \\ -1 \end{array} \right\}$

$\lambda_2 = 5 \quad A_1 = A_2 \quad \text{or} \quad A^{(2)} = \left\{ \begin{array}{c} 1 \\ 1 \end{array} \right\}$

General solution:

$\bar{X} = C_1 \bar{A}^{(1)} \sin \omega_1 t + C_2 \bar{A}^{(2)} \sin \omega_2 t$

or  $\begin{cases} X_1 \\ X_2 \end{cases} = C_1 \begin{cases} 1 \\ -1 \end{cases} \sin \omega_1 t + C_2 \begin{cases} 1 \\ 1 \end{cases} \sin \omega_2 t$

or  $X_1 = C_1 \sin \omega_1 t + C_2 \sin \omega_2 t$

and  $X_2 = -C_1 \sin \omega_1 t + C_2 \sin \omega_2 t$

Graphical representation of block oscillations:
Lecture #09

One more example of eigenvalue or characteristic type of problem:

Let us revisit Jacobi iteration to invert a matrix.

\[ A\vec{X} = \vec{b} \]

\[ \Rightarrow (L + D + U)\vec{X} = \vec{b} \]

\[ \Rightarrow D\vec{X} = \vec{b} - (L + U)\vec{X} \]

or \[ \vec{X} = D^{-1}\vec{b} - D^{-1}(L + U)\vec{X} \]

or \[ \vec{X} = S\vec{X} + \vec{C} \]

or \[ \vec{X}^{(k+1)} = S\vec{X}^{(k)} + \vec{C} : \text{ Jacobi Iteration formula} \]

S is called stationary matrix because very often, matrix \( A \) or \( (L, D, U) \) do not change and are fixed. It is \( \vec{b} \) (right hand side) that varies from one problem to the other, as a forcing function.

Define, error vector \( \vec{e}^{(k)} = \vec{X}^{(k)} - \vec{X} \)

Approximate value at iteration \# k

\[ \Rightarrow \vec{X}^{(k+1)} = \vec{X}^{(k)} - \vec{X} \]

\[ = S(\vec{X}^{(k)} - \vec{X}) = S\vec{e}^{(k)} \]

or \[ \vec{e}^{(k+1)} = S\vec{e}^{(k)} \]

or \[ \vec{e}^{(k)} = S^k \vec{e}^{(0)} \]

(you should realize that it is an eigenvalue problem/equation \( A\vec{X} = \lambda \vec{X} \) with \( \lambda = 1 \))

For \( k \to \infty \) \( S^k \to 0 \) for convergence because \( \vec{e}^{(k)} \to 0 \) as \( \vec{e}^{(0)} \neq 0 \)

Therefore, criteria for the convergence: \( S^k \to 0 \) as \( k \to \infty \)

Let \( \{y_j\}_{j=1,n} \) be the eigenvectors corresponding to the \( \lambda_j \) eigenvalue of the stationary matrix
$S(n \times n)$. Recall eigenvectors are linearly independent. So, they can form the basis of $n$ dimensional space. Alternatively, any vector can be represented as the linear combination of the eigenvectors.

Therefore,

\[ e^{(0)} = \sum_{j=1}^{n} c_j v_j \]

where $Sv_j = \lambda_j v_j$ or $Sv_1 = \lambda_1 v_1$, etc.

\[ e^{(1)} = Se^{(0)} = \sum_{j=1}^{n} c_j v_j \lambda_j \]

\[ e^{(2)} = S^2 e^{(0)} = \sum_{j=1}^{n} c_j v_j \lambda_j^2 \]

\[ \vdots \]

\[ e^{(k)} = \sum_{j=1}^{n} c_j v_j \lambda_j^k \]

For convergence, $k \to \infty$  $e^{(k)} \to 0$ or $|\lambda_j| < 1$  $(j = 1 \ldots n)$

For a $n \times n$ $S$ matrix, there will be $\lambda_1 \ldots \lambda_n$ eigenvalues.

If $|\lambda_{\text{max}}| < 1$ then all $\lambda_j < 1$

Therefore, for Jacobi Iteration to converge: $|\lambda_{\text{max}}| < 1$

$$ (\text{Max} \ (\lambda_j) \equiv \text{spectral radius} \quad \rho(S) < 1) $$

There is the Gershgorin Theorem (Check the book by Strang) which states that in such case

\[ \sum |a_{ij}| < |a_{ii}| \]

non-diagonal diagonal elements elements

Such matrix is said to be diagonally strong. It is a desirable characteristic of a matrix to be inverted. Compare two extreme situations:

\[ \begin{bmatrix}
  \times \ & \times \\
  \times \ & \times
\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}
  \circ \ & \circ \\
  \circ \ & \circ
\end{bmatrix} \]

Note that zero or no iterations are required to solve an identity matrix! Also, a diagonally strong matrix will require relatively fewer number of iterations. On the other hand, the above right hand matrix is a singular matrix and cannot be inverted. Similarly, a matrix with smaller
value-numbers on its diagonal relative to the other elements in the same row will require a relatively larger number of iterations to converge.

**Power Method**

The previous examples have shown that the ‘dynamics’ of a 1st order system can be inspected or characterized by determining $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$, instead of all $\lambda_s$. Therefore, a $n \times n$ matrix will have $n\lambda_s$, and the overall response of the system will be the least sensitive to $\lambda_{\text{min}}$ and most sensitive to $\lambda_{\text{max}}$. The response corresponding to the other $\lambda_s$ will be intermediate. **Power method** is commonly used to determine $\lambda_{\text{max}}$ and also $\lambda_{\text{min}}$.

If $[A]_{n \times n}$ matrix has $n\lambda$s (eigenvalues), then one can write $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$

And $A\vec{X} = \lambda_{\text{max}}\vec{X}$ and $A\vec{X} = \lambda_{\text{min}}\vec{X}$

**Method:** Make a guess of $X^T = [1 \ 0 \ 1]$ or $[1 \ 1 \ 0]$ (say, for a $3 \times 3$ matrix)

Then $AX^{(0)} \Rightarrow \vec{X}^{(1)}$

Use $\vec{X}^{(1)} = \frac{X^{(1)}}{\|X^{(1)}\|} = \frac{AX^{(0)}}{\|AX^{(0)}\|}$

\[
\begin{pmatrix}
\text{magnitude (you are scaling or normalizing the vector, i.e. $\vec{a}/|a|$ or $\vec{a}/\sqrt{a_1^2 + a_2^2 + a_3^2}$)} \\
\end{pmatrix}
\]

$\vec{X}^{(k)} = \frac{A\vec{X}^{(k-1)}}{\|A\vec{X}^{(k-1)}\|}$, $k = 1, 2, 3$ (# of iterations)

Can be shown that when $k \to \infty$ $\|A\vec{X}^{(k-1)}\| \to |\lambda_{\text{max}}|$

In other words, after sufficient # of iterations, one obtains maximum eigenvalue and the vector $\vec{X}^{(k-1)}$ when transformed by ‘$A$’ matrix does not rotate and is just scaled by $\lambda_{\text{max}}$, i.e.

$A\vec{X}^{(k-1)} = \lambda_{\text{max}}\vec{X}^{(k)}$

In other words, $\vec{X}^{(k-1)}$ and $\vec{X}^{(k)}$ have the same directions.
Example: Determine $\lambda_{\text{max}}$ of $A = \begin{bmatrix} 0 & 2 & 3 \\ -10 & -1 & 2 \\ -2 & 4 & 7 \end{bmatrix}$

\[ v^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \equiv [ 1 \ 0 \ 0 ]^T \quad (1^{st} \text{ guess}) \]

\[ v^{(2)} = \frac{\begin{bmatrix} 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{\|A^1\|} = \frac{\begin{bmatrix} 0 \\ -10 \\ 2 \end{bmatrix}}{\sqrt{100+4}} = \begin{bmatrix} 0.0000 \\ -0.9806 \\ -0.1961 \end{bmatrix} \]

\[ |\lambda| = \sqrt{104} = 10.198 \]

\[ v^{(3)} = \frac{\begin{bmatrix} A \\ -2.5495 \\ -5.2951 \end{bmatrix}}{\|A^2\|} = \frac{\begin{bmatrix} 0.0000 \\ -0.9806 \\ -0.1961 \end{bmatrix}}{\sqrt{2.5495^2+5.884^2+5.2951^2}} \]

\[ \lambda_2 = 5.9063 \]

After 20 iterations,

\[ X^{(20)} = \begin{bmatrix} -0.3714 \\ 0.55698 \\ 0.7428 \end{bmatrix} \quad \text{with} \lambda^{(20)} = 3.0014 \]

You will see a gradual convergence from 10.198 to 3.0014.

The basics remain the same:

$AX^{(19)} = \lambda_{\text{max}} X^{(20)}$ and $X^{(19)}$ and $X^{(20)}$ will have approximately the same direction.

$\Rightarrow$ It can be shown that $\lambda_{\text{min}}$ of $A = \frac{1}{\lambda_{\text{max}}}$ where $\lambda_{\text{max}}$ is the highest eigenvalue of the inverse of matrix $A$ or $A^{-1}$:

\[ A\bar{X} = \lambda_{\text{max}}\bar{X} \]

or $\bar{X} = A^{-1}(\lambda_{\text{max}}\bar{X}) = \lambda_{\text{max}}(A^{-1}\bar{X})$

or $A^{-1}\bar{X} = \left( \frac{1}{\lambda_{\text{max}}} \right)\bar{X}$

or $\bar{B}X \equiv \lambda'_{\text{max}}\bar{X}$

where $B$ is $A^{-1}$ and $\lambda_{\text{min}}$ of $A$ is $\left( \frac{1}{\lambda'_{\text{max}}} \right)$ of $B$ where $\lambda'_{\text{max}}$ is the highest $\lambda$ of $B$. 

4
The \( \lambda_{\text{max}} \) to \( \lambda_{\text{min}} \) ratio is known as stiffness ratio and indicates the stiffness (maximum rate of change in the functional value corresponding to \( \lambda_{\text{max}} \) relative to minimum rate corresponding to \( \lambda_{\text{min}} \)) of the system. A ratio > 10 indicates the system to be stiff requiring a fine step size for calculations (to be discussed later).

**Notes (From Strang's book):**

1. If \( X \) is an eigenvector of \( A \) corresponding to \( \lambda_{\text{max}} \) and \( A \) is invertible, then \( X \) is also an eigenvector of \( A^{-1} \) corresponding to the inverse of its \( \lambda_{\text{max}} \).

2. Eigenvectors corresponding to different \( \lambda_{s} \) are linearly independent.

3. A matrix and its transpose have the same eigenvalues.

4. A matrix is singular if and only if it has zero \( \lambda \). A non-singular matrix has all non zero \( \lambda_{s} \).

5. \( \sum \lambda_{i} = \text{trace of } A = \sum_{i=1}^{n} a_{ii} \)

6. \( \prod \lambda_{i} = \det(A) \)

7. The \( \lambda_{s} \) of an upper or lower triangular matrix are the elements on its main diagonal.

\[ \text{It also follows that } \det A = \det \left( \text{UTM of } A \right) = \prod (a_{ii})_{\text{UTM}} \]

**Quiz 1**
Lecture #10

Nonlinear Algebraic Equations

Solving\[ \begin{align*}
F_1(X_1 \ldots X_n) &= 0 \\
F_2(X_1 \ldots X_n) &= 0 \\
\vdots \\
F_n(X_1 \ldots X_n) &= 0
\end{align*} \]
or \[ \bar{F}(\bar{X}) = 0 \]

or \[ \bar{F}(\bar{X}) = \left\{ \begin{aligned}
F_1(\bar{X}) \\
\vdots \\
F_n(\bar{X})
\end{aligned} \right\} = 0 \]

where, \( F_1(\bar{X}) \) takes the form such as \( (X_1 - \sin X_1) = 0 \)

\[ \begin{aligned}
or & \quad X_1 - \sin X_2 = 0 \\
\text{and} & \quad X_1 = e^{X_2}
\end{aligned} \]

Solving a set of nonlinear algebraic equations is naturally difficult.

Let us begin with solving one nonlinear equation:

\[ f(\alpha) = 0 \quad (\text{Root finding}) \quad \alpha = ? \]

Closed Methods

1. Bracketing method (for a monotonically increasing or decreasing function)

\[ f(X) \]

\[ f(\alpha) = 0 \]

\[ X_{l1} \quad X_{l2} \quad X_{m2} \quad X_{m1} \quad X_U \quad X \]

Step 1: Corner the root. Find \( X_l \) and \( X_U \) such that

\[ f(X_l) f(X_U) < 0 \]

All it means is that one root is cornered or bracketed. Some functions may have multiple
roots (be careful and corner all three roots separately by plotting the function qualitatively but accurately).

\[ N_0 \text{t}_e_2: T\_U = f\_n\_t_i_o_n \_t_h_r_e_e \_r_o_o_t_s \_b_e_t_w_e_e_n \_X_L \_a_n_d \_X_U. \]

**Step 2:**

\[ X_m^{(1)} = \frac{X_U + X_I}{2} = X_{old} \]

Check \( f(X_m) = 0 \). If true, you have the root or \( X_m = \alpha \)

If not, check ⇒ if \( f(X_m)f(X_I) < 0 \).

- If true, \( X_U = X_m \) (replace \( X_U \) with \( X_m \))
- else \( X_I = X_m \) (replace \( X_I \) with \( X_m \))

(This way, you are cornering the root or coming closer to the root)

(\textbf{Note:} The other check \( f(X)f(X_U) < 0 \) will also work. If -ve \( X_I = X_m \) else \( X_U = X_m \))

**Step 3:**

\[ X_m^{(2)} = \frac{X_U + X_I}{2} = X_{new} \quad (X_U \_a_n_d \_X_I \_a_r_e \_n_e_w \_v_a_l_u_e_s) \]

\( \begin{cases} \text{you can begin error checks after 2nd iterations only} \\ |\xi_a| = \left| \frac{X_{m_{new}} + X_{m_{old}}}{X_{m_{old}}} \right| \times 100 < \varepsilon \quad (s_p_e_c_i_f_i_e_d \_b_y \_u_s_e_r) \\ \text{else go to step 2 (keep iterating)} \\ \text{till} \quad \left| \frac{X^{(i+1)} - X^{(i)}}{X^{(i)}} \right| \times 100 < \varepsilon \end{cases} \)

**Example:** Van der Waals gas law/equation:

\[ \left(P + \frac{a}{V^2}\right)(V - b) = \text{constant} \]
It is clear that this equation is polynomial in $V$ with $n = 3$.

Let us assume that the other variables and constants are known and the simplified equation takes the following form:

\[ f(V) = 0 \]
\[ V^3 - 0.165V - 3.993 \times 10^{-4} = 0 \]

Calculate $V$ or find the roots

**Step 0:** Bracket the root: $V[0, 0.11]$

\[ X_l = 0, \quad X_U = 0.11 \text{ by plotting} \]

1. Check $f(X_l)f(X_U) < 0$

So that the root is bracketed (cornered)

2. $X_m^{(1)} = \frac{0 + 0.11}{2} = 0.055$ (3 digits accurate after decimal)

\[ f(0.055)f(0) = 3.993 \times 10^{-4} \times 6.655 \times 10^{-5} > 0 \]

(You would have checked whether $f(0.055) \times f(0.11) < 0$ or not as well!)

\[ X_l = 0.055, \quad X_U = 0.11 \text{ (remains the same)} \]

3. $X_m^{(2)} = \frac{X_l + X_U}{2} = \frac{0.055 + 0.011}{2} = 0.0825 = X_{\text{new}}$

\[ f(X_m^{(2)}) = -1.622 \times 10^{-4} \]

Check $f(X_l)f(X_m) < 0$

\[ X_U = 0.0825, \quad X_l = 0.055 \]

4. Check $\varepsilon_a = \left| \frac{0.0825 - 0.055}{0.055} \right| \times 100 = 50.00\% > 0.2\% (\varepsilon_s)$

Therefore, $X_m = \frac{0.055 + 0.0825}{2} = 0.06875$

Do 10 iterations to obtain $X_m = 0.0625$ with $\varepsilon_a = 0.1721\% < 0.2\%$
Ans. \( V \) (root) = 0.0625.

You should also prepare a table while solving:

\[
\begin{array}{ccc}
X_l & X_U & X_m & \in & f(X_m)f(X_l) > 0 \\
& & & \text{or} & f(X_m)f(X_U) > 0 \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Decision</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

2. False – Position method (Regular-Falsi)

(This method is similar to the previous one, except draw a straight line connecting \( X_U \) and \( X_m \) to determine the intersection on X axis as the approximate value for the root)

Step1: Bracket root

\[
f(X_U)f(X_l) < 0
\]

2. \( X_r^{(1)} = X_U - \frac{f(X_U)}{f(X_l) - f(X_U)} \times (X_l - X_U)\)

\[
\left( \frac{f(X_U)}{X_U - X_r} = \frac{0 - f(X_l)}{X_r - X_l} \right)
\]

\[
= \frac{X_U f(X_l) - X_l f(X_U)}{f(X_l) - f(X_U)} = X_{old}
\]

3. Check: \( f(X_r)f(X_l) < 0 \) or \( f(X_r)f(X_U) < 0 \)

\[
X_U = X_r \quad \text{or} \quad X_l = X_r
\]

\[
X_r^{(2)} = \frac{X_U f(X_l) - X_l f(X_U)}{f(X_l) - f(X_U)} = X_{new}
\]

Check \( \epsilon_a < \epsilon_s \) else go to 2; iterate till convergence. Note that it is difficult to choose between two methods. In general, this method is faster. There are always exceptions. Try \( f(X) = X^{10} - 1! \)
Open Methods

1. Fixed Point Iteration.

Step 1. \( f(X) = 0 \) ⇒ Modify/transform it to

\[ f(X) = X - g(X) = 0 \]

eg. \( f(X) = X - \cos X \)

or \( X - e^X \)

(Try for \( x^2 - 1 \) as \( X - g(X) = 0 \) or \( X = g(X) \))

Sometimes it is not straight forward & there may be more than one combination.

The working formula is

\[ Y = X = g(X) \]  

(2 equations and find its interaction)

\[ X_{n+1} = g(X_n) \]

Sometimes, one has to start from left to the root; sometimes there is a divergence or spiral convergence, depending on type of functions.

Criterion for convergence: \( |g'(X)| < Y'(X) = 1 \) at \( X = X_n \)
**Note:** Closed methods often generate at least one root. Open methods may sometimes result in divergence depending upon type of function and starting guess, $X_0$. However, if they converge, they will do more quickly than the closed methods.

Step 2: Start with $X_0$

$$X_1 = g(X_0)$$

Step 3: Check $\varepsilon_a = \left| \frac{X_1 - X_0}{X_1} \right| < \varepsilon_s$

If not, $X_{old} = X_r$ (iterated value)

$$X_r = g(X_{old})$$

Check $\varepsilon_a = \left| \frac{X_r - X_{old}}{X_{old}} \right| < \varepsilon_s$

Else keep on iterating, keeping in mind the sequence $X_{n+1} = g(X_n)$.

### 2. Newton-Raphson Method

*(Most popular method to solve a non-linear algebraic equation)*

Step 1: Choose

$$X_r = X_0 \quad \text{(guess)}$$

2. $f'(X_r) = \text{tangent at } X_r = \frac{f(X_r)}{X_0 - X_1}$

$$X_1 = X_r - \frac{f(X_r)}{f'(X_r)}$$

(Note: $f'(X_r) \neq 0$)

3. If $\varepsilon_a = \left| \frac{X_1 - X_r}{X_r} \right| \times 100 < \varepsilon_s$ stop

$$\alpha = X_1 \quad \text{(root)}$$

else $X_r = X_1$ (repeat the steps)

Sequence: $X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)}$; $f'(X_i) \neq 0$

and $\varepsilon_{i+1} = \left| \frac{X_{i+1} - X_i}{X_i} \right| \times 100$
Notes: ⇒
- \( f'(X) \neq 0 \) is the major drawback of the method. There is a price for computing \( f'(X) \)
- Simple to code and convergence is fast (see later)
- There are cases of divergence and 'missing' roots, depending on function and initial guess

\[ f(X_{i+1}) = f(X_i) + f'(X_i)(X_{i+1} - X_i) + \frac{f''(X_i)}{2!}(X_{i+1} - X_i)^2 + \cdots \]
\[ \approx f(X_i) + f'(X_i)(X_{i+1} - X_i) + 0(h^2) \] (error)

If \( f(X_{i+1}) = 0 \), root is located
\[ X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} \]

Example (NR): \( f(X) = X^3 - 0.165X^2 + 3.993 \times 10^{-4} \)
\[ f'(X) = 3X^2 - 0.33X \]

Step 1. \( X_0 = 0.05 \): Plot \( f(X) \) and choose \( X_0 \) close to the intersection of \( f(X) \) with \( X \)-axis

2. \( f'(X_0) = 3X_0^2 - 0.33X_0 \)
\[ X_1 = X_0 - \frac{f(X_0)}{f'(X_0)} = 0.06242 \]

3. \( \varepsilon = \left| \frac{0.06242 - 0.0500}{0.0500} \right| \times 100 = 24.84\% \)
\[ X_2 = X_1 - \frac{f(X_1)}{f'(X_1)} = 0.06238 \]

\[ \epsilon = 0.06411 \Rightarrow \alpha = 0.06238 \text{ (root)} \]

(accurate to the 1st significant digit after decimal)
Lecture #11

Newton-Raphson (Multi-variables)

\[ F_1(X_1, X_2 \ldots X_n) = 0 \]
\[ F_2(X_1, X_2 \ldots X_n) = 0 \]
\[ \vdots \]
\[ F_n(X_1, X_2 \ldots X_n) = 0 \]

\[ \Rightarrow F(\bar{X}) = 0 \]
\[ \text{or } F_i(\bar{X}) = 0, \quad i = 1 \ldots n \]
\[ \text{or } F_i(X_i) = 0, \quad i = 1 \ldots n \]

Taylor-expansion of multi-variables:

\[ F_1(\bar{X}^{(k+1)}) \approx F_1(\bar{X}^{(k)}) + \frac{\partial F_1}{\partial X_1} \bigg|_{\bar{X}^{(k)} \neq X_1} \left( X_1^{(k+1)} - X_1^{(k)} \right) + \frac{\partial F_1}{\partial X_2} \bigg|_{\bar{X}^{(k)} \neq X_2} \left( X_2^{(k+1)} - X_2^{(k)} \right) + \ldots \]

\[ \frac{\partial F_1}{\partial X_n} \bigg|_{\bar{X}^{(k)} \neq X_n} \left( X_n^{(k+1)} - X_n^{(k)} \right) \quad (1^{st} \text{order accurate}) \]

\[ F_2(\bar{X}^{(k+1)}) \approx F_2(\bar{X}^{(k)}) + \frac{\partial F_2}{\partial X_1} \bigg|_{\bar{X}^{(k)} \neq X_1} \left( X_1^{(k+1)} - X_1^{(k)} \right) + \frac{\partial F_2}{\partial X_2} \bigg|_{\bar{X}^{(k)} \neq X_2} \left( X_2^{(k+1)} - X_2^{(k)} \right) + \ldots \]

\[ \frac{\partial F_2}{\partial X_n} \bigg|_{\bar{X}^{(k)} \neq X_n} \left( X_n^{(k+1)} - X_n^{(k)} \right) \]

\[ F_n(\bar{X}^{(k+1)}) \approx F_n(\bar{X}^{(k)}) + \frac{\partial F_n}{\partial X_1} \bigg|_{\bar{X}^{(k)} \neq X_1} \left( X_1^{(k+1)} - X_1^{(k)} \right) + \frac{\partial F_n}{\partial X_2} \bigg|_{\bar{X}^{(k)} \neq X_2} \left( X_2^{(k+1)} - X_2^{(k)} \right) + \ldots \]

\[ \frac{\partial F_n}{\partial X_n} \bigg|_{\bar{X}^{(k)} \neq X_n} \left( X_n^{(k+1)} - X_n^{(k)} \right) \]

If \( \bar{X}^{(k+1)} = \alpha \) (root at \((k+1)^{th}\) iteration for all variables) \( \Rightarrow F_i(\bar{X}^{(k+1)}) = 0, \)
\[ (i = 1 \ldots n) \]
\[
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
= 
\begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}^{(k)} + 
\begin{bmatrix}
\frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\
\frac{\partial F_2}{\partial x_1} & \cdots & \cdots & \frac{\partial F_2}{\partial x_n} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n}
\end{bmatrix}
\begin{bmatrix}
\Delta X_1 \\
\Delta X_2 \\
\vdots \\
\Delta X_n
\end{bmatrix}
\]

\( \Delta X_1 = X_1^{(k+1)} - X_1^{(k)}, \text{etc} \)

or
\[
\begin{bmatrix}
\frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\
\frac{\partial F_2}{\partial x_1} & \cdots & \cdots & \frac{\partial F_2}{\partial x_n} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n}
\end{bmatrix}
\begin{bmatrix}
\Delta X_1 \\
\Delta X_2 \\
\vdots \\
\Delta X_n
\end{bmatrix}
= - \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}^{(k)}
\]

or \( \overline{J} \Delta \overline{x}^{(k)} = - \overline{F}^{(k)} \)

or \( \Delta \overline{x}^{(k)} = - \overline{J}^{-1} \overline{F}^{(k)} \)

or \( \overline{x}^{(k+1)} = \overline{x}^{(k)} - \overline{J}^{-1} \overline{F}^{(k)} \) \( \Rightarrow \) Multi-variable NR method (similar in form to single variable NR method)

**Example:**

\( F_1(\overline{Y}) = 4 - 8Y_1 + 4Y_2 - 2Y_1^3 = 0 \)

\( F_2(\overline{Y}) = 1 - 4Y_1 + 3Y_2 + Y_2^2 = 0 \)

Starting guess values

\( \overline{Y}^{(1)} = [Y_1 \quad Y_2]^{T(1)} = [0.5 \quad 0.5]^T \)

Apply NR multivariable method formula:

\[
\overline{x}^{(k+1)} = \overline{x}^{(k)} - \overline{J}^{-1} \overline{F}^{(k)}
\]

or

\[
\begin{bmatrix} Y_1^{(k+1)} \\ Y_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} Y_1^{(k)} \\ Y_2^{(k)} \end{bmatrix} - \begin{bmatrix}
\frac{\partial F_1}{\partial Y_1} & \frac{\partial F_1}{\partial Y_2} \\
\frac{\partial F_2}{\partial Y_1} & \frac{\partial F_2}{\partial Y_2}
\end{bmatrix}^{-1}
\begin{bmatrix} F_1^{(k)} \\ F_2^{(k)} \end{bmatrix}
\]
\[
\begin{align*}
\{Y_1\}^{1} &= \{0.5\} - \left[\begin{array}{cc}
-8 - 6Y_1^2 & 4 \\
-(3 + 2Y_2) & 4
\end{array}\right]^{-1}\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}_{0.5,0.5} \\
&= \{0.5\} - \left[\begin{array}{cc}
-9.5 & 4 \\
-4 & 4
\end{array}\right]^{-1}\{1.75\} \\
&= \{0.5\} + \frac{1}{22}\left[\begin{array}{cc}
4 & -4 \\
4 & -9.5
\end{array}\right]\{1.75\}
\end{align*}
\]

(Note: For 2 \times 2 matrix simple Cramer’s rule was applied to determine the inverse of the matrix, else apply \(G - J\) or LU decomposition method can be used for a large size matrix)

\[Y_1^{(1)} = 0.5 + \frac{1}{22}(4 \times 1.75 - 4 \times 0.75) = 0.6818\]

\[Y_2^{(1)} = 0.5 + \frac{1}{22}(4 \times 1.75 - 9.5 \times 0.75) = 0.4943\]

- Continue iteration till there is a convergence. One can prepare a table like this:

<table>
<thead>
<tr>
<th>#</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>( \frac{\partial F_1}{\partial Y_1} )</th>
<th>( \frac{\partial F_1}{\partial Y_2} )</th>
<th>( \frac{\partial F_2}{\partial Y_1} )</th>
<th>( \frac{\partial F_2}{\partial Y_2} )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( \varepsilon_{a1} )</th>
<th>( \varepsilon_{a2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Secant Method**

(Another common open method; it does not require derivative to compute. Requires two starting adjacent guess values instead)

**Step1:**

1. Choose \( X_{i-1} \) and \( X_{i} \) adjacent guess values near the root \( \alpha \).

2. Draw straight line connecting \( f(X_{i-1}) \) and \( f(X_{i}) \). Find intersection on X axis \( \rightarrow X_{i+1} \)

\[
\begin{align*}
f'(X_{i+1}) &= \frac{f(X_{i}) - f(X_{i-1})}{X_{i} - X_{i-1}} = \frac{f(X_{i})}{X_{i} - X_{i+1}} \\
X_{i+1} &= X_{i} - f(X_{i}) \times \frac{(X_{i} - X_{i-1})}{f(X_{i}) - f(X_{i-1})} \\
&: \text{sequence}
\end{align*}
\]
Check if \( f(X_{i+1}) \to (\text{you have hit the root}) \), else repeat with \( X_i \) and \( X_{i+1} \) following the above sequence.

- The method converges faster than Bisection method, but slower than NR. It is preferred over NR to avoid \( f' = 0 \).
- If \( X_i \to X_{i-1} \), note \( X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} \) (same as NR!)

At this stage, let us look at Taylor’s series. Any function \( f(X) \) can be expanded as:

\[
f(X) = f(X_0) + f'(X_0)(X - X_0) + \frac{f''(X_0)}{2!}(X - X_0)^2 + \cdots + \frac{f^n(X_0)(X - X_0)^n}{n!} + \cdots \text{ to } \infty
\]

or, \( f(X_{i+1}) = f(X_i) + f'(X_i)(X - X_i) + \frac{f''(X_i)}{2!}(X_{i+1} - X_i)^2 + \cdots \frac{f^n(X_i)(X_{i+1} - X_i)^n}{n!} + \cdots \text{ to } \infty \)

\[
\approx f(X_i) + \frac{f'(X_i)(X_{i+1} - X_i)}{1!} + \frac{f''(X_i)(X_{i+1} - X_i)^2}{2!} + \cdots 0(h^3)
\]

or truncation error is \( 0(h^3) \) (accurate to 2\(^{nd} \) order)

\[
\approx f(X_i) + \frac{f'(X_i)(X_{i+1} - X_i)}{1!} + 0(h^2)
\]

(accurate to 1\(^{st} \) order or truncation error is \( 0(h^2) \))

(error is one order greater than the accuracy of the method)

\[
= f(X_i) + f'(X_i)(X_{i+1} - X_i) + \cdots + \frac{f^n(X_i)(X_{i+1} - X_i)^n}{n!} + R_{n+1}
\]

exact!

where, \( R_{n+1} = \frac{f^{n+1}(\xi)(X_{i+1} - X_n)^{n+1}}{n+1!} \) (can be exactly determined)

However, \( \xi \) is 'some' number between \( X_{i+1} \) and \( X_i \).

Therefore,

\[
f(X_{i+1}) = f(X_i) + \frac{f'(X_i)(X_{i+1} - X_i)}{1!} + \frac{f''(\xi)(X_{i+1} - X_i)^2}{2!}
\]

exact
\[ f(X_i) + \frac{f'(\xi)(X_{i+1} - X_i)}{1!} \quad (1^{st} \text{Mean Value Theorem}) \]

In other words, \[ f'(\xi) = \frac{f(X_{i+1}) - f(X_i)}{(X_{i+1} - X_i)} \]

All it means there is \( \xi \) somewhere between \( X_i \) and \( X_{i+1} \), where the tangent equals the divided difference between two end points \( X_i - X_{i+1} \).

**Note:** Taylor series is used to determine the error or accuracy of a numerical method, or to determine the convergence of the method.

### Convergent Analysis

**Fixed Point Iteration method**

\[ f(X) = X - g(X) \]

or \( X^{(n+1)} = g(X^{(n)}) \) (iteration)

If \( \alpha \) is the root then \( \alpha - g(\alpha) = 0 \)

Error at \( (n+1)^{th} \) iteration \= \( |\text{True Value} - \text{Approximate Value}| \)

\[ e^{n+1} = |\alpha - X^{(n+1)}| \]

\[ = |g(\alpha) - g(X^{(n)})| \]

\[ = |g'(\bar{X})(\alpha - X^{(n)})| \quad : \text{Recall Mean Value Theorem.} \bar{X} \text{ is between} \]

\[ \alpha \text{ and } X^{(n)} \]

\[ = |g'(\bar{X})e^n| \]

\[ \frac{e^{n+1}}{e^n} = |g'(\bar{X})| < 1 \text{ for error to decrease as } n \rightarrow \infty \text{ where } \bar{X} \text{ is } \{\alpha, X^n\} \]

**NR Convergent**

\[ X^{k+1} = X^k - \frac{f_k}{f'_k} ; f'_k \neq 0 \]

\[ e^{(k+1)} = |\alpha - X^{k+1}| = \left| \alpha - X^k + \frac{f_k}{f'_k} \right| \]
\[ f(X^{k+1}) = f(X^k) + f_k'(X^{k+1} - X^k) + f''(\bar{X}) \frac{(X_{k+1} - X_k)^2}{2!} \]

\[ (\bar{X} : [X_k, X_{k+1}]) \]

If \( X^{k+1} \) is the root (\( \alpha \))

\[ 0 = f(X^k) + f_k'(\alpha - X^k) + \frac{(\alpha - X_k)^2}{2!} f''(\bar{X}) \]

\[ \therefore \quad e^{(k+1)} = \left| \frac{f''(\bar{X})}{2f'_k} (\alpha - X_k)^2 \right| \]

\[ = \left| \frac{f''(\bar{X})}{2f'_k} (e^k)^2 \right| \]

\[ \frac{e^{(k+1)}}{(e^k)^2} = \left| \frac{f''(\bar{X})}{2f''(\alpha)} \right| \quad ; \quad f'(\alpha) \neq 0 \]

If \( k \to \infty \), \( e^{(k+1)} \) decreases by the square of the error in the previous step. In other words, the method has quadratic convergence (fast convergence), although the method is only 1st order accurate. This is the special feature of NR method.
Lecture #12

Function Approximation: Interpolation

Fit a line passing through the data points to predict the functional value at an intermediate data. The line must pass through the data. The method is used to interpolate, for example, certain properties such as enthalpy and vapour pressure, at an intermediate point where the functional values or properties are not available.

Therefore, the data provided for the interpolation must be reliable/accurate.

- Mathematically, interpolating function must be smooth (differential); continuous and pass through all data points.
- Compare it to ‘regression’ (best fit to the data) when the line need not pass through each data. Regression is used for prediction when data are experimentally measured or approximately calculated. Therefore, the regressed data may have errors.

- (No regression topic to be covered in this course; no extrapolation as well!)

The main motivation for developing an interpolating function is not only to predict the functional value at an interpolating/intermediate point, but also to approximate a continuous f(x) or a discrete data set, y(x) as the nth degree polynomial, Pn(x); thus the name of the current topic: function approximation. You will see in the later lessons that once you have your own approximated simple Pn(x), you will forgo the complex f(x) and work on Pn(x) instead, not only for calculating functional values at the intermediate points but also for computing f’(x) and f”(x) as Pn’(x) and Pn’’(x), respectively. Also, all formulae for integration are also derived from Pn(x).
Interpolating function for ‘n’ data points

\[ P_{n-1}(X) = a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + \cdots + a_0: \] a polynomial of degree \((n - 1)\).

If there are ‘n’ data points, a unique polynomial of degree \((n-1)\) passes through the points:

Ex. 1. \(y = mx + b\) (1\textsuperscript{st} order polynomial) passing through two data points. Two unknown & two equations to solve \(m\) and \(b\):

2. Three data points

\[ y = ax^2 + bx + c \] (2\textsuperscript{nd} order polynomial, a quadratic equation)

(3 equations to solve three unknowns \(a, b\) & \(c\))

3. Thus, ‘n’ data points will have a unique polynomial of degree \((n - 1)\) passing through the points.

\[ y = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots a_0 \]

Therefore,

\[
\begin{align*}
y_1 &= a_{n-1}x_1^{n-1} + a_{n-2}x_1^{n-2} + \cdots a_0 \\
y_2 &= a_{n-1}x_2^{n-1} + a_{n-2}x_2^{n-2} + \cdots a_0 \\
\vdots \\
y_n &= a_{n-1}x_n^{n-1} + a_{n-2}x_n^{n-2} + \cdots a_0
\end{align*}
\]

\(\text{n equations}\)

or

\[
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{pmatrix} = 
\begin{pmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\
1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & x_n^2 & \cdots & x_n^{n-1}
\end{pmatrix} \begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{n-1}
\end{pmatrix}
\]

Although the problem is well defined, it is tedious to solve \((a_0 \cdots a_{n-1})\) from a large sized \(n \times n\) matrix.

\[
\bar{y} = A_{n \times n} \bar{a}
\]

The common methods to fit \(P_{n-1}\) polynomial passing through ‘n’ data points, or \(P_n\) through \((n + 1)\) data points:

(1) Newton’s divided-difference scheme

Consider 2 data points

Let a line pass through \((X_0, Y_0)\) & \((X_1, Y_1)\):
\[
Y - Y_0 = \frac{Y_1 - Y_0}{X_1 - X_0}
\]
\[
\text{or } Y = Y_0 + \frac{Y_1 - Y_0}{X_1 - X_0} (X - X_0)
\]
\[
\text{or } Y = a_0 + a_1 (X - X_0)
\]

where \(a_0 = Y_0\)

\[
\begin{aligned}
a_1 &= \frac{Y_1 - Y_0}{X_1 - X_0} = Y(X_1, X_0) \text{ defined as Newton First Divided Difference (N1DD)} \\

\text{where } a_0 &= Y_0 \\

\text{and } a_1 &= \frac{Y_1 - Y_0}{X_1 - X_0} = Y(X_1, X_0)
\end{aligned}
\]

Similarly, define

\[
Y[X_2, X_1, X_0] \equiv \text{Newton 2}\text{nd divided difference}
\]

\[
= \frac{Y[X_2, X_1] - Y[X_1, X_0]}{X_2 - X_0} \quad (N2DD)
\]

\[
Y[X_n, X_{n-1}, \ldots, X_0] = \frac{Y[X_n, X_{n-1}, \ldots, X_1] - Y[X_{n-1}, X_{n-2}, \ldots, X_0]}{X_n - X_0} \quad (NnDD)
\]

Take 3 data points \((X_0, X_1, X_2)\):

\[
Y(X) \equiv P_2(X) = a_0 + a_1 (X - X) + a_2 (X - X_0) (X - X_1)
\]

By fitting \(P_2(X)\) to three data points, the three coefficients can be determined as

\[
a_0 = Y_0, \quad a_1 = \frac{Y_1 - Y_0}{X_1 - X_0} \equiv Y[X_1, X_0]
\]

\[
\text{and } a_2 = \frac{Y[X_2, X_1] - Y[X_1, X_0]}{X_2 - X_0} \equiv Y[X_2, X_1, X_0]
\]

Similarly for \((n+1)\) data

\[
Y(X) \text{ or } P_n(X)
\]

\[
= a_0 + a_1 (X - X_0) + a_2 (X - X_0) (X - X_1) + \cdots a_n (X - X_0) (X - X_1) \cdots (X - X_{n-1})
\]

where \(a_0 = Y_0\)

\[
a_1 = Y[X_1, X_0] = \frac{Y_1 - Y_0}{X_1 - X_0} \quad (1\text{st divided difference})
\]

\[
a_n = Y[X_n, X_{n-1}, \ldots, X_0] \quad (n\text{th divided difference})
\]
(Substitute $x = x_0, x_1, x_2 \ldots x_n$ in $P_n(x)$ to explore the sequence of polynomial)

In the tabular form,

<table>
<thead>
<tr>
<th>$i$</th>
<th>$X_i$</th>
<th>$Y_i$</th>
<th>$1^{st}$</th>
<th>$2^{nd}$</th>
<th>$\ldots$</th>
<th>$n^{th}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0$</td>
<td>$Y_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$X_1$</td>
<td>$Y_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$X_2$</td>
<td>$Y_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n-1$</td>
<td>$X_{n-1}$</td>
<td>$Y_{n-1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td>$X_n$</td>
<td>$Y_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Top row (circles) provides all NDD coefficients, $a_0, a_1, a_2 \ldots$

2. Lagrangian Interpolation

$y(x) = \sum_{k=0}^{n} l_k(x) y_k \quad (n + 1 \text{ data})$

Lagrangian coefficient of the $k^{th}$ term

- Function must be a linear combination of $y_k$.
- $l_k(x)$ must not depend on $y_k$
- $l_k(x)$ is defined as $l_j(x_i) = \delta_{ij}$ (Kronecker delta)

Propose, $l_j(x) = C_j(x - x_0)(x - x_1) \ldots (x - x_{j-1})(x - x_{j+1}) \ldots (x - x_n)$, and

$= 1 \quad \text{when} \quad i = j \quad \text{and} \quad 0 \quad \text{when} \quad i \neq j$

$= 1 \quad (x = x_j)$

Therefore, $C_j = (x_j - x_0)^{-1}(x_j - x_1)^{-1} \ldots (x_j - x_{j-1})^{-1}(x_j - x_{j+1})^{-1} \ldots (x_j - x_n)^{-1}$

and, $l_j(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{j-1})(x - x_{j+1}) \ldots (x - x_n)}{(x_j - x_0)(x_j - x_1) \ldots (x_j - x_{j-1})(x_j - x_{j+1}) \ldots (x_j - x_n)}$

The interpolating function is as follows:

$Y_n(x) = \sum_{k=0}^{n} l_k(x) y_k \equiv P_n$

where

$l_j(x) = \prod_{i=0}^{n} \frac{(x - x_i)}{(x_j - x_i)}$
Suppose 2 data points: \((X_0, Y_0)\) and \((X_1, Y_1)\)

\[
Y_1(x) = l_0(x)Y_0 + l_1(x)Y_1
\]

\[
l_0 = \frac{x-x_1}{x_0-x_1}, \quad l_1 = \frac{x-x_0}{x_1-x_0}
\]

or \(Y_1(x) = \left(\frac{x-x_1}{x_0-x_1}\right)Y_0 + \left(\frac{x-x_0}{x_1-x_0}\right)Y_1\)

(Since \(Y_1(x)\) is unique, it cannot be different from Newton-Divided difference formula; only forms are different)

Suppose there are 3 data points: \((X_0, Y_0), (X_1, Y_1),\) and \((X_2, Y_2)\)

\[
Y_2(x) = l_0(x)Y_0 + l_1(x)Y_1 + l_2(x)Y_2
\]

\[
l_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}; \quad l_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)};
\]

\[
Y(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}Y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}Y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}Y_2
\]

\(\equiv\) same as \(\text{NDDE or } aX^2 + bX + c\)

**Ex.**

<table>
<thead>
<tr>
<th>(X_0)</th>
<th>(X_1)</th>
<th>(X)</th>
<th>(X_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth (m)</td>
<td>-1</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>(Y(t^0c))</td>
<td>10.1</td>
<td>11.3</td>
<td>11.9</td>
</tr>
</tbody>
</table>

\[Y_2(x) = C_0 + C_1x + C_2x^2\] (unique)

3 data points and 3 eqns to solve \(C\):

\[
\begin{bmatrix}
1 & -1 & 1 & C_0 \\
1 & 0 & 0 & C_1 \\
1 & 1 & 1 & C_2
\end{bmatrix} = \begin{bmatrix} 10.1 \end{bmatrix} \ldots \text{Solve by any previously learnt method to obtain}
\]

\[
C_0 = 11.3, \quad C_1 = 0.9, \quad \text{and} \quad C_2 = -0.3 \Rightarrow Y(0.5) = 11.675
\]

(1) NDD

1st\(\text{DD} Y[X_0,X_1]

2nd\(\text{DD} Y[X_0,X_1,X_2]

\[\begin{array}{c|c}
X_0 & 10.1 \\
\hline
X_1 & 11.3 \\
\hline
X_2 & 11.9 \\
\end{array}\]

\[\begin{array}{c}
\frac{11.3-10.1}{0+1} = 1.2 \\
\frac{11.9-11.3}{1-0} = 0.6
\end{array}\]

5
Therefore, $Y_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$

$a_0 = Y_0 = 10.1, \quad a_1 = FDD = 1.2, \quad a_2 = SDD = -0.3$

$= 10.1 + 1.2(x + 1) - 0.3(x + 1)x$

- (Same as before, check) $Y(0.5) = 11.675$

(2) Lagrange $Y_2(x) = \sum_{k=0}^{2} l_k Y_k$; \quad $l_j(x) = \prod_{i=0, i \neq j}^{2} \frac{(x - x_i)}{(x_j - x_i)}$

$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}; \quad l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$, etc.

$Y_2(x) = \frac{(x - 0)(x - 1)}{-1 \times -2} \times 10.1 + \frac{(x + 1)(x - 1)}{1 \times -1} \times 11.3 + \frac{(x + 1)x}{2 \times 1} \times 11.9$

- (Same as before, check) $Y(0.5) = 11.675$

**Note:** ⇒ If ‘n’ is large, (> 4), the polynomial shows oscillation. In principle, one uses ‘piece-wise’ interpolation for large # of data points.

⇒ Less smooth if data points are less or sparse.

⇒ ‘Problem’ at the end points.

⇒ NDD uses the previous calculations, if an extra point is to be included in finding a new polynomial, unlike each coefficient has to be recalculated in ‘LI’. However, it is easy to program ‘LI’.

**Error analysis:** For (n+1) data points, $P_n$ (order n) is the unique polynomial to pass through (n+1) data points. This polynomial can be used to predict $P_n(X)$ at a new data points ‘X’. How much error can be expected in this prediction/interpolation? From engineering point of view, how much confidence an user has in the interpolated functional value? Well, in such case one should use an additional data $(\xi, Y(\xi))$ over the range $(X_0 \cdots X_n)$ and check the convergence in the re-interpolated value at $P_{n+1}(X)$. Note, now you have (n+2) data points including $\xi$.

Theoretically, it can be shown using Taylor’s series that the error in the interpolation using $P_n(X)$ for (n+1) data is

$$y^{n+1}(\xi) \frac{(x - x_0)(x - x_1) \cdots (x - x_n)}{n+1!}$$

where $(x_0 < \xi < x_n)$

additional data pt.
You should have also noted that

\[ y[x_0, x_1] = \frac{y(x_1) - y(x_0)}{x_1 - x_0} = y' \text{ (accurate to the 1st order)} \]

\[ y[x_0, x_1, x_2] = \cdots = y'' \text{ (accurate to the 2nd order)} \]

and

\[ y[\xi, x_0, x_1, \cdots x_n] = \frac{y[\xi, \cdots x_{n-1}] - y[x_0, \cdots x_n]}{\xi - x_n} = y^{n+1}(\xi) \]

Therefore, the error calculated is nothing but \( R_{n+1} \) (remainder) of the Taylor’s series. For more on this, read the book by Ferziger.
Lecture #13

Similar to the Newton’s Divided Difference and Lagrangian interpolation schemes, there is the **Newton’s interpolation formula** that makes use of ‘Δ’ operator (Forward Difference Operator)

\[
\Delta y(x) = y(x + \Delta x) - y(x) : \text{Forward - difference operator}
\]

\[
\begin{align*}
E y(x) &= y(x + \Delta x) : \text{Shift Operator} \\
E^2 y(x) &= E[y(x + \Delta x)] = y(x + 2\Delta x) \\
E^n y(x) &= y(x + n\Delta x)
\end{align*}
\]

Therefore, \( \Delta y(x) = E y(x) - y(x) = (E - 1)y(x) \)

or \( \boxed{E = 1 + \Delta} \) : \( E \) & \( \Delta \) operators

\[
\begin{align*}
E^\alpha &= (1 + \Delta)^\alpha = 1 + \alpha \Delta + \frac{\alpha(\alpha - 1)}{2!} \Delta^2 + \cdots + \frac{\alpha(\alpha - 1) \cdots (\alpha - n + 1)}{n!} \Delta^n + \cdots \\
E^\alpha y(x_0) &= y(x_0 + \alpha \Delta x) = (1 + \Delta)^\alpha y(x_0)
\end{align*}
\]

or \( E^\alpha y(x_0) = y(x_0 + \alpha \Delta x) = (1 + \Delta)^\alpha y(x_0) \)

\[
y(x) = y(x_0 + \alpha \Delta x) = y_0 + \alpha \Delta y_0 + \frac{\alpha(\alpha - 1)}{2!} \Delta^2 y_0 + \cdots + \frac{\alpha(\alpha - 1) \cdots (\alpha - n + 1)}{n!} \Delta^n y_0 + R
\]

\[
R = \frac{\alpha(\alpha - 1)(\alpha - n)}{(n+1)!} \Delta^{n+1} y(\xi) \quad \text{where } x_0 < \xi < x_n
\]

- Newton Forward-Difference Interpolation formula
  
  In general
  
  \[
  \begin{align*}
  \Delta y_i &= y(x_i + \Delta x) - y(x_i) = y_{i+1} - y_i \\
  \Delta^2 y_i &= \Delta(\Delta y_i) = y_{i+2} - 2y_{i+1} + y_i
  \end{align*}
  \]

Similar to ‘Δ’, \( \nabla \) has been defined as **backward difference operator**:

\[
\nabla y_i = y_i - y_{i-1} \quad \text{or} \quad \nabla y(x) = y(x_n + \alpha \Delta x), \quad \text{where } \alpha = \frac{x-x_n}{\Delta x}
\]

And, therefore, there is another Newton’s interpolation formula! Again, note that all interpolating functions are one and the same, but written in different forms. Considering that the Newton’s interpolation formula resembles Taylor’s series, it can also be used for error analysis.

To this end, we have learnt how to approximate \( f(x) \) or \( y(x) \) with \( P_n(x) \). We will see later \( f(x) \) or \( y(x) \) and \( f''(x) \) or \( y''(x) \) will be approximated with \( P'_n(x) \) and \( P''_n(x) \), respectively.
Splines

We have earlier noted that a high order polynomial \((n \geq 4)\) often shows oscillation. Therefore, a piece-wise polynomial \((n \leq 3)\) is usually preferred for most engineering applications.

→ ‘Spline’ is a piece-wise cubic polynomial \((y = ax^3 + bx^2 + cx + d)\) fitting 2 data points in segments or pieces. Is it unique? No. Because there are two extra degrees of freedom. In other words, two extra constraints or conditions may be imposed.

→ Let us define the problem:

\[
S_i = a_i x^3 + b_i x^2 + c_i x + d_i \text{ is the spline fitting the data } x_i \text{ and } x_{i+1}.
\]

Thus, it is a higher order polynomial for two data points.

1. Each spline \((S_i)\) is a piece-wise cubic.
2. Curve must pass through the points \((x_i \& x_{i+1})\)
3. \(f'_i\) and \(f''_i\) are continuous at all ‘knots’, i.e., curve is smooth and curvature is the same at all nodes.

\[
\begin{align*}
S_{i-1}(x_i) &= S_i(x_i) \\
S'_{i-1}(x_i) &= S'_i(x_i) \\
S''_{i-1}(x_i) &= S''_i(x_i)
\end{align*}
\]

\[
\begin{align*}
S^+ &= S^- \\
S'^+ &= S'^- \\
S''^+ &= S''^-
\end{align*}
\]

Therefore, for \(n\) data points there are \((n-1)\) spline segments or \(4(n-1)\) unknown quantities (coefficients) to be determined:

Equations: \(y_i = S_i(x_i)\) : \(n\)

\[
\begin{align*}
S^+ &= S^- : (n - 2) & : & \text{only interior pts.} \\
S'^+ &= S'^- : (n - 2) \\
S''^+ &= S''^- : (n - 2)
\end{align*}
\]

\(4n - 6\)

You have two BCs specified at the end data points or knots.

Therefore, there are as many \((4n-4)\) equations as \# of unknowns.

The problem is now well defined:
\[ S_i(x) = \sum_{t=0}^{3} a_t x^t = ? \]

Note that \( S''(x) \) is a piece-wise linear and continuous (i.e. \( S''(x) = 6a_i + 2b_i \))

\[
\begin{align*}
S''_i & \quad (x) \\
\bar{x}_i & \quad (linear) \\
\bar{x}_{i+1} & \\
S_i(x) \equiv & \quad \text{cubic}
\end{align*}
\]

Therefore, \( S_i(x) = \sum_{k=0}^{1} l_k(x) y_k \) (Lagrangian interpolation)

\[
= \frac{x-x_{i+1}}{x_i-x_{i+1}} S''(x_i) + \frac{x-x_i}{x_{i+1}-x_i} S''(x_{i+1})
\]

On integrating twice, \( S_i(x) = \left( \frac{x^3}{6} - x_{i+1} \frac{x^2}{2} \right) S'(x_i) + \left( \frac{x^3}{6} - x_i \frac{x^2}{2} \right) S'(x_{i+1}) + c_1 x + c_2 \)

Apply the conditions

\[
S_i(x_i) = y_i \quad \text{and} \quad S_i(x_{i+1}) = y_{i+1}
\]

\[
\Rightarrow S_i(x) = \frac{S''(x_i)(x_{i+1} - x)^3}{6\Delta_i} + \frac{S''(x_{i+1})(x - x_i)^3}{6\Delta_i} + \left( \frac{y_i - \Delta_i}{\Delta_i} \frac{S''(x_i)}{6} \right) (x_{i+1} - x) \]

\[
+ \left( \frac{y_{i+1} - \Delta_i}{\Delta_i} \frac{S''(x_{i+1})}{6} \right) (x - x_i)
\]

Apply another condition:

\[
S_{i-1}'(x_i) = S_i'(x_i) \quad \text{on the following}
\]

\[
S_i'(x) = -\frac{S''(x_i)(x_{i+1} - x)^2}{2\Delta_i} + \frac{S''(x_{i+1})(x - x_i)^2}{2\Delta_i} - \left( \frac{y_i - \Delta_i}{\Delta_i} \frac{S''(x_i)}{6} \right) \]

\[
+ \left[ \left( \frac{y_{i+1} - \Delta_i}{\Delta_i} \frac{S''(x_{i+1})}{6} \right) \right]
\]

On re-arrangement it can be shown

\[
\frac{\Delta_{i-1}}{6} S''(x_{i-1}) + \frac{\Delta_{i-1} + \Delta_i}{3} S''(x_i) + \frac{\Delta_i}{6} S''(x_{i+1})
\]

\[
= \frac{y_{i+1} - y_i}{\Delta_i} - \frac{y_i - y_{i-1}}{\Delta_{i-1}}
\]

\[
i = 2, n - 1 \quad (\text{interior nodes})
\]
If \( S''(x_1) \) and \( S''(x_n) \) are pre-known as boundary conditions, we have

\[
A\bar{x} = \bar{b} \quad \text{or} \quad AS''(x) = \bar{b} \quad \text{where } A \text{ is the tridiagonal matrix}
\]

For a ‘natural’ spline, \( S''(x_1) = S''(x_n) = 0 \) \( \text{OR} \) \( S''(1) = S''(n) = 0 \)

If \( \Delta_i = \Delta_{i-1} \) (equally spaced data)

\[
\frac{1}{6} [S''(x_{i-1}) + 4S''(x_i) + S''(x_{i+1})] = \left(\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta_i^2}\right)
\]

Call subroutine Tridiag\(\left( n-2, \frac{1}{6}, \frac{2}{3}, \frac{1}{6}, \bar{b} \right) \)

\# of eqns \# coefficients \# RHS or \( b \) vector

\( S''(x_i), \quad i = 2, n - 1 \) are now known on solving the set of eqns.

Thus, \( S_i(x) \) is determined from the boxed equation on the preceding page 3.

**Example:**

\[
\begin{array}{c|c|c|c}
x & 0 & 2.5 \\
\hline
0 & 1.67 & 0
\end{array}
\]

Fit a cubic natural spline.

Use ** above

\[
\frac{1}{6}S''(x_1) + \frac{2}{3}S''(x_2) + \frac{1}{6}S''(x_3) = \frac{0 - 2 \times 1.67 + 0}{2.5^2}
\]

(You have only one linear algebraic eqn)

\( S''(x_2) \) = -0.8016 \( i = 2 \) (middle node only)

\[
S_1(x) = -0.05344(x + 2.5)^3 + 1.002(x + 2.5) \\
S_2(x) = 1.667 - 0.4x^2 + 0.0533x^3
\]

\} \text{ Ans.}
Given a continuous \( f(x) \), numerical differentiation can be performed to determine gradient \( f'(x_i) \) at \( x_i \). Fit a polynomial (usually \( P_{1-3} \)) and then determine \( P'_{1-3} \) to estimate \( y'_i \), depending on desired order of accuracy or error. Taylor's series can be conveniently used to derive \( f' \) or \( f'' \) because the series terms contain \( f' \) or \( f'' \).

**Taylor's series:**

\[
\begin{align*}
\text{Forward difference Scheme (FDS)} & : \quad f(x) + h f'(x) + h^2/2 f''(x) + \cdots \quad \text{(1)} \\
\text{Backward difference Scheme (BDS)} & : \quad f(x) - h f'(x) + h^2/2 f''(x) - \cdots \quad \text{(2)} \\
\text{Central difference Scheme (CDS)} & : \quad f'(x) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + 0(h^3)
\end{align*}
\]

(1) Richardson’s extrapolation technique can be used to take 3 data pts.

\[
\begin{align*}
f(x + 2h) &= f(x) + 2hf'(x) + 4(h^2/2)f''(x) + \cdots \quad \text{(3)} \\
f(x - 2h) &= f(x) - 2hf'(x) + 4(h^2/2)f''(x) + \cdots \quad \text{(4)}
\end{align*}
\]

Use 1 an 3 : Multiply eq(1) by 4 and subtract from 3 to eliminate \( f''(x_i) \)

\[
\begin{align*}
\text{Forward difference Scheme (FDS)} & : \quad f'(x) = \frac{4f(x_{i+1}) - 3f(x_i) - f(x_{i+2})}{2h} + 0(h^3) \\
\text{Backward difference Scheme (BDS)} & : \quad f'(x) = \frac{-4f(x_{i-1}) + 3f(x_i) + f(x_{i-2})}{2h} + 0(h^3)
\end{align*}
\]

You will see later in the course that although BDS and FDS are also 2nd order accurate like CDS, such equations are used at the boundary points or end points only to discretize Neumann or...
mixed or flux (gradient) terms: \(-k \frac{\partial T}{\partial x}\) or \(-D \frac{\partial C}{\partial x}\) or \(-\mu \frac{\partial v}{\partial x}\), whereas CDS is invariably used at interior nodes.

Higher order \(f'(x_i)\) can also be obtained as CDS:

\[
4 \text{ data points } \begin{cases} f'(x_i) = \frac{-4f(x_{i+2}) - 8f(x_{i-1}) + 8f(x_{i+1}) + 4f(x_{i+2})}{12h} + 0(h^4) \end{cases}
\]

(See the book by Ferziger)

Similarly, use eqs 1 & 2 to obtain

\[
f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + 0(h^3)
\]

\[
= f''(x_{i-1}) = f''(x_{i+1}):
\]

Why? (You have 2\(^{nd}\) order polynomial between 3 data points)

You should note that similar to \(f'(x_i)\), FDS and BDS or formulae can also be derived for \(f''(x_i)\) using three points or manipulating Taylor’s series or equations 1-4. However, these equations are seldom or in fact, never used for most of chemical engineering problems. Why not? Because the N-S or species or thermal energy balance equations (PDEs) are 2\(^{nd}\) order, with the 1\(^{st}\) order (flux) boundary condition. Therefore, CDS for \(f''(x_i)\), applied at interior nodes, suffices.

To this end, apply Richardson’s extrapolation technique to obtain even higher order \(f''\)

\[
f''(x_i) = \frac{-f(x_{i-2}) + 16f(x_{i-1}) - 30f(x_i) + 16f(x_{i+1}) - f(x_{i+2})}{12h^2} + 0(h^4)
\]

(Five data points)

Note the notations: \(f(x_i)\) and \(y_i(x_i)\); \(f'\) and \(f''\) and \(y'\) and \(y''\). Summary at a glance:

\[
f(x) \approx P_n(x), f'(x) \approx P'_n(x), f''(x) \approx P''_n(x)
\]
Lecture #14-15

Numerical Integration

Similar to numerical differentiation, the integration of \( f(x) \), i.e., \( \int f(x) \, dx \) may be performed by approximating \( f(x) \approx P_n(x) \), usually \( n \leq 3 \). Therefore, first calculate \( y_i(x_i) \) from \( f(x) \), then fit a polynomial through the 'n' data points and determine \( \int P_n \, dx \) or area under the approximated curve. Considering usual oscillations set in the fitted polynomial for \( n \geq 4 \), piece-wise integration is performed. Detailed method and analysis are as follows.

\[ P_n(x) \text{ is the interpolating } f^n \text{ between } (x_0, \ldots, x_n) \]

It can be shown/proposed that

\[ \int_a^b f(x) \, dx = \sum_{k=0}^n w_k f(x_k); \quad f(x_k) = y(x_k) \]

where, \( n \) is the number of segments between 'a' and 'b'

and '\( w_i \)' is the respective weight at \( x_i \). It is also known as Quadrature formula. Let us look at the proposed formula differently. Apply Lagrange interpolation between \((n + 1)\) data points or \( n \) segments:

\[ P_n(x) = \sum_{k=0}^n l_k(x)y_k \quad (y_k \equiv y(x_k) \equiv y_k(x_k)) \]

\[ \int_a^b f(x) \, dx = \int_a^b \sum_{k=0}^n l_k(x)y_k \, dx = \sum_{k=0}^n \int_a^b l_k(x)y_k \, dx \]

\[ = (b - a) \sum_{k=0}^n C^k y_k; \quad C^k = \frac{1}{(b-a)} \int_a^b l_k(x) \, dx \]

\[ \text{Cotes number} \]

- It is the same as the Quadrature formula; \( n \equiv \text{parameter(degree of polynomial)} \)

(\text{It can be shown } \sum_{k=0}^n C^k = 1 \quad (\text{put } f(x) = 1))

Therefore, we can have a table as follows:
Newton-Cotes Coefficients (ends points included)

<table>
<thead>
<tr>
<th>n</th>
<th>N</th>
<th>NC₀ⁿ</th>
<th>NC₁ⁿ</th>
<th>NC₂ⁿ</th>
<th>……</th>
<th>NCᴺⁿ</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0.0183Δ³f‴</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td>0.0035Δ⁵f⁴‴</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td>0.0016Δ⁵f⁴‴</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>7</td>
<td>32</td>
<td>12</td>
<td>32</td>
<td>7</td>
<td>…</td>
</tr>
<tr>
<td>5</td>
<td>288</td>
<td>19</td>
<td>75</td>
<td>50</td>
<td>50</td>
<td>75</td>
<td>19</td>
</tr>
</tbody>
</table>

These are called closed N-C formula because they use functions at the end points also \((a, b)\). Note, \(n\) or the number of segments is usually 1 or 2 or 3, or the degree of polynomial is 1 or 2 or 3, respectively. Readers may have recognized that \(n = 1\) corresponds to Trapezoidal rule with \(-\frac{1}{2}\) coefficients of \(y_k\); \(n = 2\) corresponds to Simpson’s \(-\frac{1}{3}rd\) rule with \(\left\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right\}\) coefficients, etc. For more details, you can refer the book by Ferziger or SKG.

The error in the numerical integration can be calculated from the remainder term of the polynomial:

\[
\Rightarrow I = \int (\text{error})dx \Rightarrow \text{where ‘error’} = \frac{y^{n+1}(ξ)}{n+1!} (x - x₀)(x - x₁) \cdots (x - xₙ) \quad \text{for} \ (n + 1) \ \text{data points}
\]

⇒ SKG’s book uses Newton’s Interpolation formula to determine the various integration formulae:

\[
y(x) = y(x₀ + αΔx) = y₀ + α(Δy₀) + \frac{α(α-1)(Δ²y₀)}{2!} + \cdots \frac{α(α-1)(α-2)\cdots(α-n)(Δⁿ⁺¹y₀)}{n+1!} Rₙ
\]

To obtain the different methods for integration, viz. Trapezoidal’s rule, Simpson’s \(\frac{1}{3}rd\), etc, two approaches must give the identical results because polynomial of the highest degree that can fit to \((n + 1)\) data points is unique.

Take \(n = 1\)

\[
∫^b_a f(x)dx = (b - a) \sum_{k=0}^{n} C^n_k f(x_k) = (b - a) \left[\frac{1}{2}y_a + \frac{1}{2}y_b\right] = \frac{(b-a)}{2} (y_a + y_b)
\]
\[ or \quad \frac{(b-a)}{2} (f(a) + f(b)) \]

(Trapezoidal rule)

\[ n = 2 \]

\[
\int_a^b f(x)dx = \frac{(b-a)}{6} [f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]
\]

\( \text{Simpson's}^{\frac{1}{3}} \text{rd} \ \text{formula} \)

\( \text{(higher order accurate)} \)

(Note: \( P_2 \equiv (a + bx + ax^2) \equiv \text{parabola} \))

\[ = \frac{\Delta x}{3} [f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)] \quad \Delta X = \frac{b-a}{2} \]

\[ n = 3 \]

\[
\int_a^b f(x)dx = \frac{(b-a)}{8} [f(a) + 3f(c) + 3f(d) + f(b)]
\]

\[ = \frac{3\Delta x}{8} [\quad \text{ditto} \quad ], \quad \Delta x = \frac{(b-a)}{3} \]

- Simpson's \( 3/8 \) formula.

In actual practice, applying a single Cotes formula over the entire range is rarely done. Considering large error for large '\( \Delta X \)', interval is broken into sub-intervals & then applying piece-wise quadrature formula and summing it over.

\[
\int_{X_{i-1}}^{X_i} f(x)dx = \frac{h}{2} [f(X_{i-1}) + f(X_i)]
\]

\[ h = \frac{(b-a)}{n} \ (n + 1 \ \text{data points or} \ n \ \text{segments}) \]
Therefore,
\[ \int_{a}^{b} f(x)dx = \sum_{i=1}^{n} \frac{h}{2} [f(x_{i-1}) + f(x_{i})] \]
\[ = \frac{h}{2} \left( f(x_0) + 2 \sum_{k=1}^{n-1} f(x_k) + f(x_n) \right) \]
\[ = \frac{(b-a)}{n} \left( \sum_{k=0}^{n} f(x_k) - \frac{1}{2} [f(a) + f(b)] \right) \]

**Trapezoidal rule**

Simpson’s \(1/3\)rd rule: (intervals must be even or data points must be odd),

<table>
<thead>
<tr>
<th>(n)</th>
<th>(N)</th>
<th>(NC_0^n)</th>
<th>(NC_1^n)</th>
<th>(NC_2^n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \int_{a}^{b} f(x)dx = \left( \frac{b-a}{6} \right) (f(a) + 4f(c) + f(b)) \]
\[ h = \frac{b-a}{2} \]

\[ h = \frac{(b-a)}{2n} \quad \text{(n even segments or } \quad \text{n + 1 odd data}) \]

(Note that all even nodes ‘\(X_k\)’ are summed/counted twice in calculating total area)

What do you do if \(n \equiv \text{odd}\) or data are even? Apply the method to one data less so that \(n = \text{even}\). Add the area corresponding to the left-over segment, calculated using Trapezoidal rule!

**One open NC method is popular**: Mid-point rule, the method is frequently applied when the \(f^n\) is discontinuous at end points: \(a\) or \(b\) or both.

\[ \int_{a}^{b} f(x)dx = (b-a) \left[ f \left( \frac{a+b}{2} \right) \right] \]
Example:

Consider the classical heat transfer/transport problem of SS heat transfer at constant wall temperature in the fully developed region of a tubular laminar parabolic flow, often discussed in TP lectures. The dimensionless SS temperature \( \theta(z, r) \) in the fluid is analytically derived. See the solution above. Determine the 'mix-cup' temperature at \( z = 0.5 \).

**Sol**: In a flow-system, mix-cup temperature is the radially emerged temperature, taking into consideration the total heat (cal) carried away by the flowing fluid in unit time across the cross-section of the tube.

\[
\bar{T} = \left( \frac{\int_0^R \rho C_p v(r') T(2\pi r') dr'}{\int_0^R \rho C_p v(r') 2\pi r' dr'} \right) \quad \text{(Note: } dq = v2\pi r dr \text{)}
\]

In the dimensionless quantities,

\[
\bar{\theta}(z) = \frac{\int_0^1 \theta(r,z)(1-r^2)r dr}{\int_0^1 (1-r^2)r dr}
\]

Simpson’s \( 1/3 \) rule can be applied by taking odd number (5) of data points over \( r \equiv (0, 1) \).

Therefore # of segment = 4, step size, \( h = \frac{1-0}{4} = 0.25 \), \( \Delta r \equiv h = 0.25 \); \( r_i = i\Delta r \)
Numerical integration is performed for both numerator and denominator terms.

\[
I = \int_0^1 f(r)dr / \int_0^1 \Phi(r)dr
\]

where \( f(r) = \theta(r, z)(1 - r^2)r \quad \& \quad \Phi(r) = (1 - r^2)r \)

For both functions,

\[
I = \frac{h}{3} \left[ f(X_0) + 4 \sum_{k=1,3,5}^{n-1} + 2 \sum_{k=2,4}^{n-2} + f(X_n) \right]
\]

\[
= \frac{h}{3} \left[ f(X_0) + 4 \sum_{1,3} + 2 \sum_{2} + f(X_4) \right]
\]

Best way is to prepare a table for calculations:

<table>
<thead>
<tr>
<th>( k )</th>
<th>( r_k )</th>
<th>( \theta_k )</th>
<th>( (1 - r_k^2)r_k )</th>
<th>( \theta_k(1 - r_k^2)r_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1.7083</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>-1.7698</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>2</td>
<td>0.50</td>
<td>-1.9427</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>-2.1917</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>4</td>
<td>1.00</td>
<td>-2.4583</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

\[
\sum \text{Add}   \sum \text{Add}
\]

Show,

\[
\bar{\theta}_{\text{mix-cup}}(0.5) \equiv -1.9976 = \frac{T_T - T_w}{r_w}
\]

(wall is colder than bulk fluid)

What does decide the # of data points ‘n’ or the step size ‘h’? It is the level of desired accuracy. Repeat the entire calculations for \( n = 9 \) and check your requirement of an improved \( \bar{\theta}_{\text{mix-cup}} \) from the previous value -1.9976. Sometimes you have to decrease ‘n’ or increase ‘h’ if you have exceeded accuracy limit!

**Error Analysis**

Let us take the simplest Trapezoidal rule applied to one segment over (a,b).

Start from Newton’s Difference Interpolation formula:
\[ f(x) = f(x_0) + \alpha \Delta f(x_0) + \frac{\alpha(\alpha - 1)}{2!} \Delta^2 f(x_0) + \cdots + R_n \]

\[ R_n = \frac{\alpha(\alpha-1) \cdots (\alpha-n)}{n+1!} \Delta^{n+1} f(\xi) \quad \text{where} \quad x_0 < \xi < x \]

\[ \alpha = \frac{x-x_0}{h} \quad \text{or} \quad dx = h \, da \quad ; \quad x_0 = a, \]

If there is only one segment, \( h = (b-a) \)

\[ I = \int_a^b f(x) \, dx = h \int_0^1 \left( f(x_0) + \alpha \Delta f(x_0) + \frac{\alpha(\alpha - 1)}{2!} \Delta^2 f(x_0) \right) \, dx \]

\[ P_1 = (ax + b) \]

\[ = h \left[ f(a) + \frac{1}{2} (f(b) - f(a)) + \left( \frac{\alpha^3}{6} - \frac{\alpha^2}{4} \right) \int_0^1 h^2 f''(\xi) \right] \]

\[ = h \left[ \frac{f(a)+f(b)}{2} \right] - \frac{1}{12} h^3 f''(\xi) \quad (f''(\xi) = \frac{\Delta^2 f(\xi)}{2h}) \]

Therefore, error on one segment is \( \frac{1}{2} h^3 f''(\xi) \) or \( 0(h^3) \)

If there are \( n \) segments between \((a \& b)\) error should also be summed up to

\[ \frac{1}{12} h^3 \sum_{n} f''(\xi) = \frac{1}{12} h^2 f''(\bar{\xi})(b - a) \quad \text{or} \quad 0(h^2) \]

\[ \text{Note:} \quad f''(\bar{\xi}) = \frac{\Sigma f''(\xi)}{n} = \frac{h \Sigma f''(\xi)}{(b-a)}. \]

Similarly, for the error estimation of Simpson\'s \( \frac{1}{3} \) rule start with \( P_2 = ax^2 + bx + c, \) or

\[ R_n = \frac{\alpha(\alpha-1)(\alpha-2) \Delta^3 f(\xi)}{3!} \quad \text{over one segment or 3 data pts.} \]

to show that error \( \equiv 0(h^5) f''''(\xi) \) and if summed over \( 'n' \) segments error \( = 0(h^4) f''''(\bar{\xi}). \)

Simpson\'s \( \frac{3}{8} \) rule: error(one segment) \( \equiv 0(h^5) \)

error( segments) \( \quad \equiv 0(h^4) \)

A common question is asked: Show that Simpson\'s \( \frac{1}{3} \) rule gives the error corresponding to a cubic interpolating function, although it is based on a parabola or a quadratic polynomial. See Chhapra and Cannale\'s book for the details.

Mid-Term
Lecture #15-16

1st order ODE

A set of ODEs takes the following form:

\[
\begin{align*}
\frac{dy_1}{dt} &= f_1(t, y_1, y_2 \cdots y_n) \\
\vdots \\
\frac{dy_n}{dt} &= f_n(t, y_1, y_2 \cdots y_n)
\end{align*}
\]

or \( \frac{dy}{dt} = \bar{f}(t, \bar{y}) \)

Requires initial conditions to solve \( \bar{y} \).

\[ t = 0, \quad y_1 = y_{10} \cdots y_n = y_{n0}, \text{ etc or } \bar{y} = \bar{y}_0 \]

Note that ODEs can also be 1st order on \( x \) (space) & \( \frac{dy}{dx} = \bar{f}(x, \bar{y}) \) with \( x = 0, \quad \bar{y} = \bar{y}_0 \).

⇒ We explore numerical solution to ODEs starting with Euler’s forward and backward methods. These methods are basics, and also set the tone for the stability analysis. Let us begin with one ODE.

**Euler’s Method**

\[
\frac{dy}{dx} = f(x, y) \quad ; \quad y(x_0) = y_0
\]

or \( y_{n+1} = y_n + \frac{y_{n+1} - y_n}{h} = f(x_n, y_n) + 0(h^2) \)

(\( \)This is the general ‘discretization’ of a derivative at the ‘n’th grid. Recall the lecture on numerical differentiation. We have used FDS.)

or \( y_{n+1} = y_n + hf(x_n, y_n) + 0(h^2) \)

**Note:** The RHS terms are known to be 1st order accurate or have 2nd order error.

From the previous calculations, start from \( y(x = 0) = y_0 \to y_1 \to y_2 \cdots \text{ etc} \). Therefore, you march forward and the method is called Euler’s forward or explicit method. Think carefully. The method is all about taking a linear slope/tangent at \((x_n, y_n)\) and moving on to \((x_{n+1}, y_{n+1})\).
Graphically, it is all about calculating true derivatives (gradients) at \( x_n \) as you march forward:

\[
\begin{align*}
y'_0 &= f(x_0, y_0) \\
y'_1 &= f(x_1, y_1) \\
&\vdots \\
y'_n &= f(x_n, y_n)
\end{align*}
\]

It is clear that smaller is ‘h’ (step-size), less is the error, or accuracy is higher: \( O(h^2) \). However, there is a price for long CPU time. Also, stability may become an issue. Let us look at the stability aspect of the method.

\[
\frac{dy}{dx} = f(x, y) = f(x_0 + \Delta x, y_0 + \Delta y)
\]

\[
= f(x_0, y_0) + (y - y_0) \frac{df}{dy}_{x_0, y_0} + (x - x_0) \frac{df}{dx}_{x_0, y_0} + 0(\Delta x^2, \Delta y^2)
\]

Multi-variable Taylor series can be applied to estimate gradient \( y'(x_0, y_0) \):

\[
\frac{dy}{dx} = \alpha y + \beta x + \gamma
\]

This gives exponential solution for linear solution.

\[
\frac{dy}{dx} = \alpha y :\text{ Model equation for testing stability. (Note: Stability is governed by the homogeneous part of the solution)}
\]

Let us look at differently: Introduce perturbation or find true value, \( \bar{y} \).

Therefore, \( \frac{d\bar{y}}{dx} = \alpha \bar{y} + \beta x + \nu \)

or, \( \frac{d\xi}{dx} = \alpha(y - \bar{y}) = \alpha \xi \) \( (\xi = y - \bar{y}) \)

We have the same inference: error satisfies homogeneous part of the solution. The equation describes how ‘\( \xi \)’ can grow because of a small perturbation.

At \( x = x_0, y = y_0 \) ⇒ exact solution: \( y = y_0 e^{\alpha x} \): If \( \alpha > 0 \), \( f^n \) grows, \( \alpha < 0 \), \( f^n \) is bounded.
Now Apply Euler’s method:

\[ y_{n+1} - y_n = h f(x_n, y_n) = h y'_n = h(\alpha y_n) \]

\[ y_{n+1} = y_n (1 + \alpha h) \]

\[ y_1 = y_0 (1 + \alpha h), \quad y_2 = y_0 (1 + \alpha h)^2 \]

\[ y_n = y_0 (1 + \alpha h)^n \Rightarrow \text{Numerical solution} \]

If \( \alpha > 0 \) function grows or unbounded
If \( \alpha < 0 \) function may be bounded

Let us write, \( \alpha = \alpha_r + i\alpha_i \)

\[ y_n = y_0 (1 + \alpha_r h + i\alpha_i h)^n \]

For function to be bounded

\[ |(1 + \alpha_r h) + i\alpha_i h| < 1 \]

\[ (1 + \alpha_r h)^2 + \alpha_i^2 h^2 < 1 \]

\[ 0 > \alpha_r h > -2 \] if \( \alpha_i = 0 \) for stability

**Note**: \( \alpha = -ve \), the exact solution is always bounded! Therefore, Euler’s forward/explicit numerical method is conditionally stable. **Example**:

\[ \frac{dy}{dx} = -y \ (x = 0, y = 1) \Rightarrow y = e^{-x} \]

Numerical sol\( ^n \): exact solution:

\[ \alpha = -1; \quad \alpha_r = -1, \quad \alpha_i = 0 \]

\[ 0 > \alpha_r h > -2 \] for stability

or \[ h < +2 \]

\[ \Rightarrow y_{n+1} - y_n = h f(x_n, y_n) = -hy_n \Rightarrow y_n = (1 - h)^n \]

Plot to see that oscillations start creeping in \( h > 2 \)
Backward or Implicit Euler:

\[ \frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0 \]

\[ \frac{y_{n+1} - y_n}{h} = f(x_{n+1}, y_{n+1}) + 0(h^2) \quad \text{(Note this BDS may require iteration)} \]

\[ y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \]

Test it on the model eq
\[ \frac{dy}{dx} = \alpha y \quad \text{for the stability analysis} \]

\[ y_{n+1} = y_n + h(\alpha y_{n+1}) \]

\[ y_{n+1} = y_n/(1 - \alpha h) \]

\[ = y_0(1 - \alpha_r h - i\alpha_i h)^{-n} \]

For \( f^n \) to be bounded \(|(1 - \alpha_r h - i\alpha_i h)| > 1 \)

or \( (1 - \alpha_r h)^2 + \alpha_i^2 h^2 > 1 \)

or \(|(1 - \alpha_r h)| > 1 \) if \( \alpha_i = 0 \)

Clearly, \( 0 > \alpha_r h > 2 \)

Draw the region:

For all \( \alpha_r h \) in the left half of the plane \( \alpha_r < 0 \) and the method is unconditionally stable

For all \( h \) outside the circle the method is stable (or \( \alpha_r h > 2 \))

It is clear the stability region for the implicit Euler is much more larger than that of the explicit Euler.

Notes: Error decreases with decreasing step sizes \( h \): \( O(h^n) \), however, at the expense of CPU time. This is not surprising. The actual ‘issue’ is with increasing step sizes. There is a tendency to speed up computation by taking large step-size. An explicit method like Euler Forward may not permit or allow you to do so, even if the level of desired accuracy may be acceptable. Instability may set in at large ‘h’. On the other hand, Euler Backward or an implicit method permits you to use relatively larger ‘h’ without instability, yielding a bounded solution.
⇒ ‘Error’ must not be mixed up with the ‘instability’. A method may be relatively higher order accurate, but may show instability at small step-size. In general, implicit methods are more stable than explicit methods, but require iterations to compute $y_{n+1}$. Examples:

\[
\frac{dy}{dx} = f(x, y) = ay^2
\]

**Explicit:** $y_{n+1} - y_n = h ay_n^2$

**Implicit:** $y_{n+1} - y_n = h ay_{n+1}^2$ (requires iterations or NR method?)

⇒ "A method is stable if it produces a bounded solution when it is supposed to and is unstable if it does not." Therefore, while solving $\frac{dy}{dx} = -y$, if the numerical method produces a stable solution, it is stable, else unstable. On the other hand, there is no instability ‘issue’ with $\frac{dy}{dx} = +y$ because the exact solution itself is unbounded. Therefore, both Euler Forward(explicit) or Backward(implicit) will produce an unbounded solutions!

Get back to $\frac{dy}{dx} = -y (\alpha = -1)$ and $y_n = (1 + h)^{-n}$

Error increases with increasing ‘h’ but the solution remains stable. Compare to Forward Euler when $\alpha < 2$ for a stable solution.

⇒ Trapezoidal rule ($O(h^2)$):

\[
y_{n+1} - y_n = \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})]
\]

(Implicit method)

It can also be tested on the model equation:

\[
\frac{dy}{dx} = \alpha y
\]

or $y_{n+1} - y_n = \frac{\alpha h}{2} [y_n + y_{n+1}]$

\[
y_{n+1} = y_n \left( \frac{1+\alpha h/2}{1-\alpha h/2} \right)
\]

or $y_n = y_0 \left( \frac{1+\alpha h/2}{1-\alpha h/2} \right)^n$
For stability $\left| \frac{(1+\alpha h/2)}{(1-\alpha h/2)} \right| < 1$ or

Each method has a stability region for selecting the step-size, 'h' for calculation without instability.

See the textbooks for details.
Lecture #17

Single Step Methods

Forward or Backward Euler is the simplest single step method to solve ODEs. The methods are 1\textsuperscript{st} order accurate. A general higher order method can be derived considering that it is all about choosing a suitable gradient at \((x_n,y_n)\) or \((t_n,y_n)\) to calculate \(y_{n+1}(x_{n+1})\):

\[
\frac{dy}{dt} = f(t,y); \quad y(0) = y_0
\]

\[
y_{n+1} = y_n + h\phi(t_n,y_n,h) \quad (h = \text{step size} = \Delta t)
\]

It is all about finding: \(\phi(t_n,y_n,h) = \left(\frac{dy}{dt}\right)_{t_n,y_n} = ?\)

Propose as

\[
y_{n+1} = y_n + h[ak_1 + bk_2] \quad (1) \Rightarrow \text{general method.}
\]

Here, \(k_1\) and \(k_2\) are the slopes with weights \(a\) & \(b\) respectively (Forward Euler may be considered to assume \(a = 1, b = 0\) and \(k_1 = y'_n, y_n\), whereas Backward Euler assumes \(a = 0, b = 1, k_2 = y'_{n+1}, y_{n+1}\))

Here,

\[
k_1 = f(t_n,y_n) \quad k_2 = f(t_n + ph, y_n + q(hk_1))
\]

Thus, the model has \(4\) parameters; \(a, b, p, q\).

Thus \(k_2\) (slope) is calculated at \((t_n + ph)\), fraction less than \((t_n + h)\) or \(t_{n+1}\), as \(x\) – coordinate, and \(y_n + q(hk_1)\), fraction less than \((y_n + hk_1)\) or \(y_{n+1}\), as \(y\) – coordinate.

\[
k_2 = f(t_n,y_n) + f_t(t_n,y_n).ph + f_y(t_n,y_n)qhk_1 + 0(h^2)
\]

- Taylor multi-variable series applied on \(k_2(t, y)\) with \(\Delta t = ph, \Delta y = qhk_1\)

Substituting in (1)

\[
y_{n+1} = y_n + h[af(t_n,y_n) + bf(t_n,y_n)] + h^2[bpf_t(t_n,y_n) + bpf_y(t_n,y_n)] + 0(h^3) \quad (2)
\]
Also, restart from \( \frac{dy}{dt} = f(t, y(t)) \) as a total derivative on \( t \).

or \( y_{n+1} = y_n + hf + \frac{h^2}{2!}f' + 0(h^3) \): (Single variable(t) Taylor – Series)

where \( f' = \frac{df}{dt}(t_n, y_n) = \left[ \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} \frac{dy}{dt} \right]_{t_n, y_n} = [f_t + f_yf] \)

On substitution,

\[
y_{n+1} = y_n + hf + \frac{h^2}{2}f_t(t_n, y_n) + \frac{h^2}{2}f_yf + 0(h^3)
\]

Thus, model equation (2) can be equated with the fundamental equation (3) derived for \( y_{n+1} \):

\[
\begin{align*}
(a + b &= 1) \\
bp &= 1/2 \\
bq &= 1/2
\end{align*}
\]

Case 1: \( a = 1/2 \), \( b = 1/2 \). \( p = q = 1 \)

\[
y_{n+1} = y_n + \frac{h}{2} \left[ f(t_n, y_n) + f(t_n + h, y_n + hy'_n) \right] + 0(h^3)
\]

Write differently,

\[
y_{n+1}^* = y_n + hf(x_n, y_n) \quad \text{predictor full step}
\]

\[
y_{n+1} = y_n + \frac{h}{2} \left[ f(t_n, y_n) + f(t_n + h, y_{n+1}^*) \right] \quad \text{corrector full step}
\]

Look at graphically,

Therefore, first predict \( y_{n+1}^* \) and then correct as \( y_{n+1} \) by taking the average of the slopes, \( k_1 \) and \( k_2 \): \( y_{n+1} = y_n + h\bar{k} \)
- This method is called Heun's improved method or modified Euler method or Runga-Kutta (RK-2) method.

**Case 2:** $a = 0, b = 1, p = q = \frac{1}{2}

\[ y_{n+1} = y_n + h \left[ f \left( t_n + \frac{h}{2}, y_n + \frac{h}{2} y'_n \right) \right] \]

Write differently,

\[ y^*_{n+1/2} \equiv y_n + \frac{1}{2} h y'_n \quad \text{(Euler predictor $\frac{1}{2}$ step)} \]

\[ y_{n+1} \equiv y_n + h \left[ f \left( t_n + \frac{h}{2}, y^*_{n+1/2} \right) \right] : \text{Mid-point corrector full step} \]

Graphically,

\[ \begin{array}{c}
\text{(1) Start with slope 1} \\
\text{Reach half step.}
\end{array} \]

\[ \begin{array}{c}
\text{(2) Calculate slope 2} \\
\text{(3) Re-start with the slope 2 to reach full step.}
\end{array} \]

- This method is called midpoint method.

3. **RK-4**

\[ y_{n+1} = y_n + \frac{h}{6} \left[ k_1 + 2k_2 + 2k_3 + k_4 \right] + O(h^5) \]

\[ \bar{k} = \text{avg of four slopes with more weights at mid points.} \]

\[ k_1 = f(x_n, y_n) \]

\[ k_2 = f(x_n + \frac{h}{2}, y_n + k_1 \frac{h}{2}) \quad \text{or} \quad f(x_{n+1/2}, y^*_{n+1/2}) \]

\[ y^*_{n+1/2} : \text{Euler predictor $\frac{1}{2}$ step} \]

\[ k_3 = f(x_n + \frac{h}{2}, y_n + k_2 \frac{h}{2}) \quad \text{or} \quad f(x_{n+1/2}, y^{**}_{n+1/2}) \]
\[ y_{n+1/2} : \text{Backward Euler } \frac{1}{2} \text{ step correction} \]

\[ k_4 = f(x_n + h, y_n + k_3 h) : \text{Mid point predictor full step} \]

Write differently,

\[ y_{n+1/2}^* = y_n + \frac{h}{2} f(x_n, y_n) : \text{EP } \frac{1}{2} \text{ step} \]

\[ y_{n+1/2}^{**} = y_n + \frac{h}{2} f(x_{n+1/2}, y_{n+1/2}^*) : \text{BE } \frac{1}{2} \text{ step corrector} \]

\[ y_{n+1}^{***} = y_n + hf(x_{n+1/2}, y_{n+1/2}^{**}) : \text{Mid point predictor full step.} \]

\[ y_{n+1} = y_n + \frac{h}{6} [f(x_n, y_n) + 2f(x_{n+1/2}, y_{n+1/2}^*) + 2f(x_{n+1/2}, y_{n+1/2}^{**}) + f(x_{n+1}, y_{n+1})] \]

\[ : \text{Simpson full step corrector.} \]

Graphically,

\[ \bar{k} = (k_1 + 2k_2 + 2k_3 + k_4)/6 \]

Notes: (1) RK-4 is a 4\textsuperscript{th} order accurate method!

(2) The method is explicit (does not require iterations to calculate \( y_{n+1} \)). You can march ahead from \( x_n \) to \( x_{n+1} \). Yet, considering that it makes use of the value \( y_{n+1} \) at \( x_{n+1} \) ahead of \( x_n \) via a predictor step, the method possesses some ‘characteristics’ of an implicit method. Therefore, the method also produces a stable solution at a relatively larger step size.

(3) Coding (program) is simple.
(4) Even if two or more number of simultaneous ODEs are coupled, the method is explicitly applied w/o requiring iterations,

(5) Considering that the method is explicit, non-linearity is not an issue.

(6) Considering that the method is explicit, the implementation is identical for a system of ODEs.

See example below:

Solve: \( \frac{dy_1}{dt} = -100y_1 \) with \( y_1(0) = 2 \) \( h = \Delta t = 0.02 \)

\( \frac{dy_2}{dt} = 2y_1 - y_2 \) \( y_2(0) = 1 \)

Write alternatively, solve \( \frac{dy}{dt} = f(x, y) \); \( y(t, 0) = \bar{y}_0 \)

or \( \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} -100 \\ 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} ; \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \)

or

\( \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} -100y_1 \\ 2y_1 - y_2 \end{bmatrix} ; \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \)

(1) \( \bar{k}_1 = f(t_n, y_n) \)

\( \bar{k}_1 = \begin{bmatrix} k' \\ k'' \end{bmatrix} = \begin{bmatrix} -100 \times 2 \\ 2 \times 2 - 1 \end{bmatrix} = \begin{bmatrix} -200 \\ 3 \end{bmatrix} \)

(2) \( \bar{k}_2 = \begin{bmatrix} k' \\ k'' \end{bmatrix} \Rightarrow \bar{k}_2 = f(t_{n+1/2}, y_n + \frac{1}{2}\bar{k}_1 h) \)

\( = f \begin{bmatrix} 0.01, & 2 + \frac{1}{2}(-200) \times 0.02 \\ 0.01, & 1 + \frac{1}{2}(3) \times 0.02 \end{bmatrix} = f \begin{bmatrix} 0.01, & 0.00 \\ 0.01, & 1.03 \end{bmatrix} = \begin{bmatrix} 0.00 \\ -1.03 \end{bmatrix} \)

(3) \( \bar{k}_3 = \begin{bmatrix} k' \\ k'' \end{bmatrix} \Rightarrow \bar{k}_3 = f(t_{n+1/2}, y_n + \frac{1}{2}\bar{k}_2 h) \)

\( = f \begin{bmatrix} 0.01, & 2 + 0 \times \frac{0.02}{2} \\ 0.01, & 1 - 1.03 \times \frac{0.02}{2} \end{bmatrix} = f \begin{bmatrix} 0.01, & 2.000 \\ 0.01, & 0.9897 \end{bmatrix} = \begin{bmatrix} -100 \times 2 \\ 2 \times 2 - 0.9897 \end{bmatrix} \)

= \begin{bmatrix} -200 \\ 3.0103 \end{bmatrix} \)
\( (±) \kappa_4 = \begin{bmatrix} k'_4 \\ k''_4 \end{bmatrix} \Rightarrow \kappa_4 = f(t_{n+1}, y_n + h\kappa_3) \\
= f\begin{bmatrix} 0.02, & 2 - 0.02 \times 200 \\ 0.02, & 1 + 0.02 \times 3.0103 \end{bmatrix} = f\begin{bmatrix} 0.02, & -2 \\ & 1.060206 \end{bmatrix} \\
= \begin{bmatrix} -100 \times -2 \\ 2 \times -2 - 1.060206 \end{bmatrix} = \begin{bmatrix} 200 \\ -5.060206 \end{bmatrix} \)

\[
\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \frac{0.02}{6} \begin{bmatrix} -200 + 2 \times 0 - 2 \times 200 + 200 \\ 3 - 2 \times 1.03 + 2 \times 3.0103 - 5.060206 \end{bmatrix} \]

\[
= \begin{bmatrix} 0.66667 \\ 1.00633 \end{bmatrix} \quad \text{Ans.}
\]

\( ⇒ \) How to choose h? Choose a reasonable ‘h’ depending on the physics of the problem. Solve and re-do the calculation for \( h/2 \) and check the convergence. Sometimes, one may have to increase the step-size if the accuracy can be relaxed.

\( ⇒ \) There are R-K-Mersen or R-K-Fehlberg automatic step-size control method where ‘h’ is adjusted (either \( h \to 2h \) or \( h \to h/2 \)) depending upon the error in the RK-4 solution relative to that from error in a higher order method proposed by Mersen or Fehlberg.

**In summary,** the single step methods to solve 1st order ODE(s) are all about finding a relatively “accurate” slope or \( \tan \Theta \) to reach the target at “h” step forward. The difference between the methods is the computation of the slopes, viz.,

\( (t_n, y_n) \) or \( (t_{n+1}, y_{n+1}) \) or \( (t_{n+1/2}, y_{n+1/2}) \) or a combination of these?

Thus, the methods are called differently (explicit/implicit) and have different orders \((1^{st}/2^{nd}/4^{th}, etc)\) of accuracy and different stability regions (smaller or bigger):

\[
y_{n+1} = y_n + h \phi(t_n, y_n): \text{Forward Euler} \\
y_{n+1} = y_n + h \phi(t_{n+1}, y_{n+1}): \text{Backward Euler} \\
y_{n+1} = y_n + h [\phi(t_n, y_n) + (t_{n+1}, y_{n+1})]/2: \text{Trapezoidal Rule} \\
y_{n+1} = y_n + h k: \text{RK-4 predictor-corrector}
\]

*where, k is some weightage average of slopes calculated at some intermediate locations.*

**Note:** All methods predict/estimate the functional value, \( y_{n+1} \) linearly starting with initial condition, \( y_n \) at \( x_n \).
Lecture #18

Example 1 (RK-4)

Consider a special (diameter = 10 cm) ball made of steel \( k = 40 \) \( W/m - K \), \( \rho = 8000 \ kg/m^3 \), \( C_p = 400 \ J/kg - k \). The initial temperature of the ball is 300 K. It is immersed in a large oil tank at 400 K. The convective heat transfer coefficient, \( h \) at the sphere surface is 3000 \( W/m^2 - k \). Assume that there is no radial temperature gradient inside the ball. Use the R-K-4 numerical technique to solve the governing first order energy balance equation for predicting the temperature of the sphere at \( t = 1 \) min after it was immersed in the oil tank. Use \( \Delta t = 10 \) s. Repeat calculations for \( \Delta t = 5 \) s. Draw graphical depiction of the slopes comprising all stages of the RK method. Do calculations up to 4 digits after decimal. Calculate the rate of temperature-decrease at \( t = 0, 0.5 \) and 1.0 min.

![Diagram showing the energy balance equation and calculations](image)

The energy balance (transient/unsteady equation):

\[
\rho C_p \frac{dT}{dt} = -ha(T - T_0)
\]

\( (kg/m^3 \cdot J/kg - k \cdot \frac{k}{s}) = J/s - m^2 - k \ (400 - T) K \left( \frac{4\pi R^2}{4/3\pi R^3} \right) \frac{m^2}{m^3} \)

\( 8000 \times 400 \times \frac{dT}{dt} = +3000 \times \frac{3}{0.05} \times (400 - T) \)

\[
\frac{dT}{dt} = +0.05625(400 - T) = f(t,T)
\]

\( \Delta T = h = 10 \) or 5 s

\( k_1 = f(0,300) = +0.05625(400 - 300) = +5.6250 \)

\( k_2 = f(5, 300 + 5.625 \times 5) = +0.05625(400 - 328.125) = 4.0429 \)

\( k_3 = f(5, 300 + 4.0429 \times 5) = 0.05625(400 - 320.2145) = 4.4879 \)
\[ k_4 = f(10, 300 + 4.4879 \times 10) = 0.05625(400 - 344.879) = 3.1005 \]
\[ \bar{k} = \frac{1}{6} (5.6206 + 2 \times 4.0429 + 2 \times 4.4879 + 3.1005) = 4.2979 \]

\[ T = 300 + 4.2979 \times 10 = 342.9795 \text{ K Ans.} \]

March for the next time steps \( t = 20, 30, 40, 50, 60 \text{ s} \), each time using the calculated values from the previous time steps (viz. 342.9795, etc). Repeat the entire calculations for \( \Delta t = 5\text{s} \).

⇒ In a 20-25 min quiz or examination, such problem can be solved for one time step using calculator. One requires to write a code to solve such problem for a long time or several steps calculations.

⇒ Rate at \( t = 0, 0.5 \) and \( 1.0 \text{ min} \)

\[
\frac{dT}{dt} \bigg|_0 = 0.05625(400 - T) = 0.05625 \times 100 = 5.6250 \text{ k/s}
\]

(Note: Once the functional values or temperatures are determined or known, the derivatives can also be calculated using the formulae you learnt! But, here is the analytical solution.)

Plot the graph as done in the previous lecture and approximately draw the steps for \( k_1, k_2, k_3, k_4, \bar{k} \), and functional values \( (T_n, T^*_n, T^{**}_n, T^{***}_n, T^*_n+1, T^*_n+1) \) on y-axis.

⇒ From the chemical engineering point of view, temperature-gradient in (steel) sphere was neglected considering a large thermal conductivity of the material, \( k \gg h'R' \) or small Biot #. Therefore, the lump body approach was used to determine the sphere temperature, and the resulting energy balance equation is derived to be a 1st order ODE, else.

**Ex2.** Consider a steady-state 1D reactive flow of a gaseous species, \( A \) in a long quartz tubular reactor (\( ID = D \), length = \( L \)) (Re > 5000) radiated by UV light. The average velocity in the tube is \( \bar{V} \). The species \( A \) is converted to B by the 1st order homogeneous reaction \( (r = kC_A) \), as it flows in the tube. The inlet concentration of \( A \) is \( C_{Ao} \). Determine the axial concentration profiles of \( A \) in the reactor. The reaction is exothermic: \( (-\Delta H; \text{cal/mol}) \).

Inlet temperature is \( T_o \)

\[
\begin{align*}
&T_o \\
&C_{Ao} \\
\end{align*}
\]

\[
\begin{align*}
\bar{V} & \\
C_A(x) &=? \\
(r &= kC_A) \\
x &= L \\
\end{align*}
\]
A species balance can be derived as follows, neglecting radial concentration profiles \((\text{Pe, } r>>1)\)

\[
\frac{\partial c_A}{\partial t} + \vec{V} \frac{\partial c_A}{\partial x} = D_x \frac{\partial^2 c_A}{\partial x^2} - k c_A \quad (\text{mol/(s - m}^3))
\]

or, \(\vec{V} \frac{dc_A}{dx} = D_x \frac{d^2 c_A}{dx^2} - k c_A \quad - (1)\)

Note that the reaction is exothermic and the rate constant is temperature dependent. Write down the energy-balance eqn:

\[
\rho C_p (\frac{\partial T}{\partial t} + \vec{V} \frac{\partial T}{\partial x}) = \frac{k}{\rho C_p} \frac{\partial^2 T}{\partial x^2} + k c_A (-\Delta H) \quad (\text{cal/(s - m}^3))
\]

\[
\vec{V} \frac{dt}{dx} = \alpha \frac{\partial^2 T}{\partial x^2} + \frac{k}{\rho C_p} c_A (-\Delta H) \quad - (2)
\]

Further approximation can be made if mass/thermal diffusion terms are neglected \((\vec{V} \nabla c_A \gg D \nabla^2 c_A \text{ and } \vec{V} \nabla T \gg \alpha \nabla^2 T)\), considering a large axial Pe, x # for mass as well as thermal transport. The equations (1-2) are modified as

\[
\vec{V} \frac{dc_A}{dx} = -k c_A = -k_0 \exp\left(-\frac{E}{RT}\right) c_A \quad - (3)
\]

\[
\vec{V} \frac{dT}{dx} = \frac{k_0 \exp\left(-\frac{E}{RT}\right)}{\rho C_p} (-\Delta H) c_A \quad - (4)
\]

or \(\frac{dy_1}{dx} = f_1(y_1, y_2) = \frac{k_0 \exp\left(-\frac{E}{Ry_2}\right)}{\vec{V}} y_1 = A \exp\left(-\frac{B}{y_2}\right)y_1\)

\(\frac{dy_2}{dx} = f_2(y_1, y_2) = \frac{k_0 \exp\left(-\frac{E}{Ry_2}\right)(-\Delta H) y_1}{\vec{V} \rho C_p} = C \exp\left(-\frac{B}{y_2}\right)y_1\)

The two 1\textsuperscript{st} order equations are coupled and the right hand side terms have non-linear dependence on \(y_1\) and \(y_2\). RK-4 can be applied without any problem, with the conditions:

\(X = 0; \quad y_1 = y_{10} \quad \text{and} \quad y_2 = y_{20}\). You can choose a step-size of \(\Delta x = L/10\).

\textbf{Notes:} - If the axial diffusion terms are not neglected, the governing equations will assume a form of 2\textsuperscript{nd} order ODE, which will be discussed in the next lecture as Boundary Value Problems.

- There are higher-order RK methods and even implicit RK method discussed in literature to solve ODE or initial value problem.
There are two commonly used multi-step methods for solving ODEs (Note: RK-4 is a single step multi-stage method): Adams-Bashforth and Adams-Moulten. These methods use a polynomial on the derivatives of a function, passing through the single or multi-points.

**Adams-Bashforth (Explicit)**
\[
\frac{dy}{dt} = f(y, t) \equiv f(y(t), t)
\]
\[t = 0, \quad y(0) = y_0\]
Solve for \(y(t) = ?\)

\[
\begin{align*}
\{ t_0 & \quad t_1 & \quad t_n & \quad t_{n+1} \\
y_0 & \quad y_1 & \quad y_n & \quad y_{n+1} \quad : \text{solve} \\
y'_0 & \quad y'_1 & \quad y'_n & \quad y'_{n+1} \quad : \text{gradient}
\end{align*}
\]

We have,
\[
y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} y' \, dt
\]

Define \(\alpha = \frac{t-t_n}{h} : \Rightarrow y_{n+1} = y_n + h \int_0^1 y'(\alpha) \, d\alpha \quad (0 \leq \alpha \leq 1)\) \quad -(1)

Fit a \(j\)th order polynomial of \(y'_n\) through \(X_n\) and \(X_{n-j}\) \((j + 1 \text{ pts preceding } X_n)\) using backward difference formula \((\nabla y_i = y_i - y_{i-1})\).

\[
y'(\alpha) = \left[ 1 + \alpha \nabla + \frac{\alpha(\alpha + 1)}{2!} \nabla^2 + \ldots \frac{\alpha(\alpha + 1) \cdots (\alpha + j - 1)}{j!} \nabla^j \right] y'_n + R(\xi)
\]

(Thus, \(y'(\alpha)\) is the polynomial between \((X_n, X_{n-1}, X_{n-2} \cdots X_{n-j})\), substituting in \((1)\)

\[
y_{n+1} = y_n + h \left[ 1 + \frac{1}{2} \nabla + \frac{5}{12} \nabla^2 + \ldots \frac{3}{8} C_j \nabla^j \right] y'_n + h \int_0^1 R(\xi) \, d\xi
\]

\(j = 0\) : you have fitted a constant polynomial of the derivative \((y')\)

\[
y'_{n+1} = y_{n+1} = y_n + h y'_n + \frac{h^2}{2} y''(\xi) \quad (y_n < \xi < y_{n+1})
\]

\(j = 1\), you have fitted a line (linear polynomial) between \(X_n\) and \(X_{n-1}\)
\[ y_{n+1} = y_n + h \left( y_n' + \frac{1}{2} vy_n' \right) + \frac{5}{12} h^3 f''(\xi) \]

\[ = y_n + \frac{h}{2} [3f(y_n) - f(y_{n-1})] \quad - 2^{nd} \text{ order} \quad (Adam - Bashforth) \]

\[ j = 2, \quad \text{you have a quadratic polynomial between} \; (X_n, X_{n-1}, X_{n-2}) \]

\[ j = 2, \quad \text{you have a quadratic polynomial between} \; (X_n, X_{n-1}, X_{n-2}) \]

Therefore, by taking different numbers of points preceding \( t_n \) or \( X_n \) one can get a set of working formulae having different \( O(h^n) \)

\[ y_{n+1} = y_n + h \sum_{m=1}^{k} \beta_{km} f_{n+1-m} \]

<table>
<thead>
<tr>
<th>( \beta_{1m} )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2\beta_{2m} )</td>
<td>3</td>
<td>-1</td>
<td></td>
<td></td>
<td>( 1^{st} \text{ order} )</td>
</tr>
<tr>
<td>( 12\beta_{3m} )</td>
<td>23</td>
<td>-16</td>
<td>5</td>
<td></td>
<td>( 3^{rd} \text{ order} )</td>
</tr>
</tbody>
</table>

**Graphically,**

\[ j = 1 \text{ (linear)} \]

\[ j = 0 \text{ (constant)} \]

\[ \Rightarrow \]

\[ j = 2 \text{ (quadratic)} \]

\[ \Rightarrow \text{ From the above description, it is clear that the method has a ‘starting’ problem because only one condition is known:} \; y(0) = y_0. \text{ In practice one starts with the Euler’s forward/explicit approach to calculate} \; y_1, \text{ then use} \; 2^{nd} \text{ order method to predict} \; y_2, \text{ and then use} \; (y_0, y_1, y_2) \text{ to calculate} \; y_3. \text{ Use of the preceding ‘3’ data-points gives} \; 3^{rd} \text{ order accurate method.} \]

\[ \Rightarrow \text{ Similar to the Adams-Bashforth (explicit), there is Adams-Moulton (Implicit) method based on forward difference interpolation formula. For details, refer the textbooks.} \]
Lecture #19

Instability and stiffness for a system of ODEs

In the previous lectures, we have shown that a system of ODEs (IVP) \( y'_i = f_i(y_1 \ldots y_m) \) \( i = 1,2,\ldots m \) with \( y_i(0) = y_0 \) can be written as the coupled ODEs: \( \vec{y}'_i = A\vec{y} \), where \( A \) is the Jacobian matrix.

It is also clear that for the \( m \times m \) matrix \( A \), there are \( m \) eigenvalues: \( A\phi_k = \lambda_k\phi_k \), where \( \phi_k \) is the eigenvector corresponding to \( \lambda_k \) eigenvalue. Now, make use of linear transformation: one can construct a matrix, \( S \) whose columns are the eigenvectors of \( A \) such that \( S^{-1}AS = \Lambda \), where \( \Lambda \) is the diagonal matrix of the elements which are \( \lambda_k \) of \( A \), as

\[
\begin{bmatrix}
\lambda_1 & \cdots & 0 \\
0 & \cdots & \lambda_k \\
\end{bmatrix}
\]

Then one can write,

\[ \vec{z}' = (S^{-1}\vec{y}') = (S^{-1}A\vec{y}) = (S^{-1}AS)S^{-1}\vec{y} = \Lambda \vec{z} \]

or

\[ z'_k = \lambda_k z_k, \quad k = 1,2,\ldots n \]

− cannonical form.

where, \( z'_k \) are uncoupled linear ODEs. Therefore, all (stability) characteristics of single ODE will also be shown by the system of ODEs.

We have also seen earlier shown that a general solution to the system of ODEs can be written as

\[ C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} + \cdots \]

where the terms in the parenthesis are the eigenvectors corresponding to the eignevalues. In such case, the growth or decay of the function depends on \( \lambda_{\min} - \lambda_{\max} \). A large numerical value of \( \lambda_{\max} \) will cause the function to grow or decay at a faster rate than that corresponding to \( \lambda_{\min} \).
\( \lambda_{\text{min}} \). In other words, small change in ‘t’ will cause a large change/variation in \( y(\lambda_{\text{max}}) \). Solving such type of system of ODEs is numerically not trivial because a fine grid or step-size \( \Delta t \) or \( h \) is required to accurately predict change in \( y(\lambda_{\text{max}}) \), whereas a coarse grid can be used to predict \( y(\lambda_{\text{min}}) \).

- Such equations are called stiff ODEs. In general, \( \text{Stiff ratio} (SR) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} > 10 \) represents a system of stiff ODEs.
- Examples:

  (1) A common example is the boundary layer problem in hydrodynamics or heat or mass transport. Functional value changes sharply within the boundary layer, from that at the surface to that in the potential region far from the surface. Therefore, a numerical computation of the functional values in the boundary layer (bottom region), in principle, requires fine grid size, \( \Delta x_1 \), while that in the potential region (top region) requires coarse grid size, \( \Delta x_2 \). However, the sets of the conservation equations in two regions are coupled through some interfacial boundary conditions, and therefore, must be solved simultaneously using \( \Delta x_1 \), which is CPU-wise extensive!

  (2) A combination of series and parallel chemical reactions with low and high rate constants:

  ex: \( \frac{dy}{dt} = \begin{bmatrix} -100 & 0 \\ 2 & -1 \end{bmatrix} y \); \( y(0) = [2 \ 1]^T \)

  on coefficient matrix ‘A’: \( (-100 - \lambda)(-1 - \lambda) = 0 \Rightarrow \lambda_1 = -100, \lambda_2 = -1 \)

  \( \lambda_1 = -100 : \begin{bmatrix} -100 & 0 \\ 2 & -1 \end{bmatrix} \{Y_1\} = 100\{Y_1\} \{Y_2\} \) (Note: \( SR = 100 \))

  \( 2X_1 = -99X_2 \Rightarrow Y^{(1)} = \begin{bmatrix} 1 \\ -2/99 \end{bmatrix} \)

  \( \lambda_2 = -1 : \begin{bmatrix} -100 & 0 \\ 2 & -1 \end{bmatrix} \{Y_1\} = -1\{Y_1\} \{Y_2\} \)

  \(-100Y_1 = -Y_1 \) and \( 2Y_1 - Y_2 = -Y_2 \Rightarrow Y^{(2)} = \{0 \ \ 1\} \)
GS: $Y = C_1 e^{-100t} \left\{ \begin{array}{c} 1 \\ -\frac{2}{99} \end{array} \right\} + C_2 e^{-t} \{0\}

or

$Y_1 = C_1 e^{-100t}$

$Y_2 = -\frac{2}{99} C_1 e^{-100t} + C_2 e^{-t}$

Apply: $\bar{Y} = \frac{2e^{-100t}}{99} - \frac{4}{99} e^{-100t}$} 

Analytical solution.

To solve numerically, a fine grid size $\Delta t$ will be initially required to accurately predict a fast decay in the functional value of $y_1$, which can be relaxed or step-size can be increased only during the later part of calculations.

In other words, time change in $y_2$ is gradual (slow). Therefore, one would have used a coarse grid size for $y_2$ but for a rapid change in $y_1$ during initial time of calculations which requires fine grid sizes.

$\Rightarrow$ SKG’s book has given one good example of stiff equations

$\frac{dy_1}{dt} = 77.27(y_2 - y_1 y_2 + y_1 - 8.375 \times 10^{-6} y_1^2)$

$\frac{dy_2}{dt} = \frac{-y_2 - y_1 y_2 + y_3}{77.23}$

$\bar{Y}(0) = [4 1 1.4]^T$

$\frac{dy_3}{dt} = 0.161(y_1 - y_3)$

In this case, stiffness changes with time, requiring adaptive mesh sizes: coarse when $f_n$ values slowly change and fine when they change rapidly.

$\Rightarrow$ Fine mesh/step – size is required
If you use one fixed fine step-size throughout the calculations, it is a wastage of CPU time.

**Gear's Technique**: Used to solve stiff ODEs, based on multiple steps but using predictor-corrector approach:

**Predictor (explicit)**: 
\[ \bar{y}_{n+1} = \alpha_1 y_n + \alpha_2 y_{n-1} + \cdots + \alpha_k y_{n-(k-1)} + h \beta_0 y_n' \]

**Corrector (implicit)**: 
\[ y_{n+1} = \alpha_1 y_n + \alpha_2 y_{n-1} + \cdots + \alpha_k y_{n-(k-1)} + h \beta_0 y_{n+1}' \]

You should note that this method also has a "starting" problem, similar to the previously discussed multi-step methods.

**Predictor Table**

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
<th>( \alpha_3 )</th>
<th>( \ldots )</th>
<th>( \alpha_6 )</th>
<th>( \beta_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>(-3/2)</td>
<td>6/2</td>
<td>(-1/2)</td>
<td>\ldots</td>
<td>0</td>
<td>(6/2)</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>6</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

**Corrector Table**

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
<th>( \alpha_3 )</th>
<th>( \ldots )</th>
<th>( \alpha_6 )</th>
<th>( \beta_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(4/3)</td>
<td>(-1/3)</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>(2/3)</td>
</tr>
<tr>
<td>3</td>
<td>(18/11)</td>
<td>(-9/11)</td>
<td>(2/11)</td>
<td>\ldots</td>
<td>0</td>
<td>(6/11)</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>6</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

The first row \((k = 1)\) of each table represents \( j = 0 \) (no previous data in the interpolating function); the second row \((k = 2)\) represents \( j = 1 \) (one preceding data in the interpolating function or a linear function), and the third row \((k = 3)\) represents \( j = 2 \) (two preceding data in the interpolating function or a quadratic function or parabola), etc.

**Quiz II**
Lecture #20

BVP (Boundary Value Problems)/$2^{nd}$ order ODEs

In this lecture, we will learn how to apply finite difference (direct) methods to solve BVPs. Such problems are common in 1D steady-state heat and mass transport in solid or fluid. The general form of the BVP assumes a $2^{nd}$ order ODE:

$$\frac{d^2y}{dx^2} = f(x, y, y')$$

with two boundary conditions required at two ends of the domain ($x = 0$ and $x = L$ or $(0,1)$).

A general form of BVP can also be written as:

$$A(x) \frac{d^2y}{dx^2} + B(x) \frac{dy}{dx} + C(x)y + D(x) + E = 0; \quad (A(x) \neq 0)$$

The required boundary conditions at two ends of the domain can assume any form: Dirichlet or Danckwarts or Neumann. That is, either functional value or gradient or mixed condition can be specified:

$$y = c, \text{ or } y' = c, \text{ or } ay' + by = c.$$  

Let us solve a general BVP equation:

$$A(x) \frac{d^2y}{dx^2} + B(x) \frac{dy}{dx} + C(x)y + D(x) + E = 0;$$

Step 1: divide the domain into $N$ equal steps/grids or grids. Therefore, there are $(N+1)$ nodes.

Step 2: discretize the equation or each terms of the equation at $i^{th}$ grid. In this course, we will discretize the terms using $2^{nd}$ order accurate method (recall numerical differentiation).

$$A(x_i)\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + B(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + C(x_i)y_i + D(x_i) + E = 0$$

Step 3: Arrange the terms:

$$\left(\frac{A(x_i)}{h^2} - \frac{B(x_i)}{2h}\right)y_{i-1} - \left(\frac{2A(x_i)}{h^2} - C(x_i)\right)y_i + \left(\frac{A(x_i)}{h^2} + \frac{B(x_i)}{2h}\right)y_{i+1} = -D(x_i) - E$$

or

$$a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i \quad \text{← discretized form of equation}$$
In principle, this equation should be valid for \( i = 0 \cdots N \). However, there may be problem (!) at the boundary nodes \( i = 0 \) and \( N \), because \( y_{-1} \) and \( y_{N+1} \) may not be known or defined or calculated. You should carefully inspect the given boundary conditions and decide to apply the discretized equation over \( (i = 0 \cdots N) \) or \( (i = 1 \cdots N) \) or \( (i = 1 \cdots N - 1) \) or \( (i = 0 \cdots N - 1) \) so that the number of equations is the same as that many unknowns \( (y_i) \). For the moment, let us take the simplest boundary conditions, \( y(0) = y_0 \) and \( y(N) = y_N \).

Apply the discretized equation over \( i = 1 \cdots N - 1 \) as

\[
\begin{align*}
i = 1: \quad & a_1 y_0 + b_1 y_1 + c_1 y_2 = d_1 \\
& a_2 y_1 + b_2 y_2 + c_2 y_3 = d_2 \\
& \vdots \\
N - 1: \quad & a_{N-2} y_{N-3} + b_{N-2} y_{N-2} + c_{N-2} y_{N-1} = d_{N-2} \\
& a_{N-1} y_{N-2} + b_{N-1} y_{N-1} + c_{N-1} y_N = d_{N-1}
\end{align*}
\]

Step 4: Apply the BCs and bring the terms containing \( y_0 \) and \( y_N \) to RHS:

\[
\begin{align*}
b_1 y_1 + c_1 y_2 &= (d_1 - a_1 y_0) = d'_1 \\
a_2 y_1 + b_2 y_2 + c_2 y_3 &= d_2 \\
& \vdots \\
a_{N-2} y_{N-3} + b_{N-2} y_{N-2} + c_{N-2} y_{N-1} &= d_{N-2} \\
a_{N-1} y_{N-2} + b_{N-1} y_{N-1} &= (d_{N-1} - c_{N-1} y_N) = d'_{N-1}
\end{align*}
\]

There are exactly \((N-1)\) unknowns and \((N-1)\) equations. Notably, set of equations can be written as

\[
A\bar{y} = \bar{b} \quad \text{where} \quad A = \begin{bmatrix} b_1 & c_1 & 0 & \cdots \\
a_2 & b_2 & c_2 & \cdots \\
0 & 0 & a_{N-2} & b_{N-2} \\
0 & a_{N-1} & b_{N-1} & \end{bmatrix}
\]

is the tridiagonal matrix.

Step 5: **Call Thomas Algorithm (learnt earlier) to invert such matrix:**

Tridiagonal \((N - 1, a, b, c, d, Y)\)

\# of eqns \(= \) coefficients of matrix \( A \)

output \((i = 1, \cdots N - 1)\)

You should be able to clearly recognize that while preparing to call tridiagonal matrix, the following coefficients were changed:

\[
d'_1 = d_1 - a_1 y_0 \quad : \ 1^{st} \text{ row } (i = 1)
\]

The other coefficients remained the same: \((i = 2, N-2)\)

\[
a_i = \frac{A(x_i)}{h^2} - \frac{B(x_i)}{2h} \\
b_i = C(x_i) - \frac{2A(x_i)}{h^2}
\]
\[ c_i = \frac{A(x_i)}{h^2} + \frac{B(x_i)}{2h}; \quad d_i = -D(x_i) - E; \]

Also, \( d_{N-1} \) was changed as \( d'_{N-1} = d_{N-1} - c_{N-1}Y_N \): last row \( i = N - 1 \)

⇒ Let us consider another example where the BVP equation is the same. At boundary, \( y(0) = y_0 \)(also same as before). However, at the other boundary, gradient is specified instead of functional value: \( y'_N = 0 \).

The boundary condition \( y'_N \) can be discretized as \( \frac{y_{N+1} - y_{N-1}}{2h} = 0 \) \( (0(h^2)) \) or \( Y_{N+1} = Y_{N-1} \)

Considering that \( Y_N \) is NOT known in this case, the coefficients \( (a_i, b_i, c_i, d_i) \) are extended to the rows: \( i = 1 \ldots N, \) i.e., the matrix \( A \) will have \( 'N' \) rows and the 1st & last row will be modified as

\[ d_1 = d_1 - a_1Y_0 \quad \text{(same as before)} \]

and

\[ a_N = a_N + c_N; \quad \text{why?} \]

because the \( N^{th} \) equation

\[ a_N Y_{N-1} + b_N Y_N + c_N Y_{N+1} = d_N \]

is modified as

\[ a_N Y_{N-1} + b_N Y_N + c_N Y_{N-1} = d_N \] on substitution of the bc.

Finally, call \( \text{Tridiagonal} (N, a, b, c, d, Y); \quad i = 1, N \)

⇒ It is recommended that you should do your best to keep consistency between the order of methods used to discretize the equation and boundary conditions. Also note that, 2nd order discretization method is reasonably good/accurate for engineering calculations. If you use a higher order method, the coefficient matrix \( A \) will no more be tridiagonal that is easy and fast to invert.

Ex: Consider the steady-state one dimensional heat conduction in a fine \( (\text{length } L, \text{ cross-section } A, \text{ perimeter } P) \), whose one end is at a constant temperature \( T_f \) and the other end is insulated. The ambient temperature is \( T_a \) and the convective heat transfer coefficient at the fix surface is \( h \).

Solve the steady-state temperature profiles in the fin.

An energy balance over 'AΔx' control volume will result in the following differential equation:

\[ \rho C_p \left( \frac{dT}{dx} \right)_\text{Solid} + \nabla \cdot \nabla T = k \nabla^2 T + S; \quad \frac{\text{cal}}{\text{s} - \text{m}^3} \cdot \frac{m^2}{(P \Delta x)} \Rightarrow \frac{\text{m}^2}{(A \Delta x)} \Rightarrow \frac{\text{m}^3}}{\text{m}^3} \]
Note that there is no source or sink of heat as such in the fin. However, considering 1D axial heat transfer, heat dissipated from the fin surface is homogenously distributed within the CV: \( S = -h(T - T_a)A \) where \( A \) is the surface area per unit volume of the fin.

\[
\Rightarrow k \frac{d^2T}{dx^2} - \frac{h(T-T_a)P}{A} = 0
\]

or \( \frac{d^2T}{dx^2} + BT + C = 0 \) where \( B = \frac{-hP}{kA} \); \( C = \frac{hP}{kA}T_a \)

and choose \( \Delta h = \frac{L}{10} \)

\[\text{BC: } T(x = 0) = T_f \quad \text{or} \quad -k \nabla T (x = L) = 0 \quad \text{or} \quad \frac{dT}{dx} = 0 \]

Discretize the equation using the 2nd order method.

Soln:

\[
\left( \frac{d^2T}{dx^2} + BT + C = 0 \right)
\]

\[
T_{i-1} - 2T_i + T_{i+1} = -Ch^2
\]

\[ T_{i-1} + (Bh^2 - 2)T_i + T_{i+1} = -Ch^2 \quad i = 1, N \]

• Prepare the tridiagonal matrix

\[
a_i = 1, \quad b_i = (Bh^2 - 2), \quad c_i = 1, \quad d_i = -Ch^2, \quad i = 1, N
\]

Apply BC/discretize BC:

\[ T_0 = T_f (i = 0), \quad T_N+1 = T_{N-1} (i = N) \]

Therefore, 1st & last row will be modified as

\[
d_1 = d_1 - a_1T_f
\]

\[a_N = a_N + c_N \]

How? 1st row is modified as \( (Bh^2 - 2)T_1 + T_2 = -Ch^2 - T_f \)

Last row is modified as \( T_{N-1} + (Bh^2 - 2)T_N + T_{N-1} = -Ch^2 \)

Therefore, you have the tridiagonal matrix A. Call the subroutine as:

**Tridiagonal (N, a, b, c, d, y)** (Note that the total number of equations is \( N \))

**As an example**, take the fin-problem of heat transfer: \( L = 1 \text{ cm} \), width of the fin = 0.1 cm, \( T_f = 100^\circ C \), \( T_a = 30^\circ C \), \( h = 100 \text{ w/m}^2K \), \( k = 20 \text{ W/m} - K \). Take \( \Delta h = 0.1 \text{ cm} \), write a programming code to obtain the solution and plot the temperature profiles:

\[ \begin{array}{c}
100^\circ C \\
T \\
30^\circ C
\end{array} \]

\[ \begin{array}{c}
\text{1 cm}
\end{array} \]
Calculate the rate of heat-loss/dissipation to atmosphere from the fin.

Note: \( Q \text{ (cal/s)} = \int h \, A \, (T(x) - T_a) \, dx = \sum h \, A \, (T_i - T_a) \, \Delta x, \ i = 0, \ n \), where \( n \) is the number of nodes of the computational domain. Use Simpson’s 1/3 integration or Trapezoidal formula. Recall your knowledge of heat transfer. \( Q \) can also be calculated by determining the heat leaking into the fin at \( x = 0 \), viz. \( Q = -k \, A \, \frac{dT}{dx} \). You can calculate the derivative by the 2nd order FDS using \( i = 0, \ 1 \) and 2 nodes. With a sufficient number of nodes or grids, two computations must produce the same results (cal/s). You should revisit Heat Transfer taught at the UG level. You can also calculate the fin-efficiency.

**Ex:** Consider the catalytic oxidation of \( SO_2 \) into \( SO_3 \) under the constant temperature and pressure conditions, over a spherical (diameter \( d_p = 1 \text{ mm} \)) pt-dispersed porous carbon catalyst. \( SO_2 \) in air (1\%) diffuses \( (D = 1 \times 10^{-9} \text{ m}^2 / \text{s}) \) into the pores of the catalyst and is oxidized to \( SO_3 \) by a pseudo-zeroth order reaction, \( r = 0.01 \text{ moles s}^{-1} \text{m}^{-3} \). The convective film mass transfer coefficient is \( k_m = 0.1 \text{ m/s} \).

Determine the steady-state concentration profiles of \( SO_2 \) within the pellet.

**Sol:**
\[
P, T = \text{constant}
\]

\[
P_{o,SS} \begin{aligned} 
\frac{\partial C}{\partial r} + V \cdot \nabla C_A &= D_{pore} \nabla^2 C_A - k \\
\text{Solid}
\end{aligned}
\]

\[
\text{or } \frac{D_{pore} \frac{d}{dr} \left( r^2 \frac{dC_A}{dr} \right)}{r^2} - k = 0 \quad (0 \leq r \leq r_p)
\]

continue in the next lecture………
Lecture #21

(continued.....from the previous lecture)

\[
\text{or} \quad D_p \frac{d^2 C_A}{dr^2} + \frac{2D_p}{r} \frac{dC_A}{dr} - k = 0
\]

\[
\text{or} \quad \frac{d^2 y}{dr^2} + \frac{2}{r} \frac{dy}{dr} + A = 0 \quad ; \quad A = \left( -\frac{k}{D_p} \right)
\]

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad i-1 & \quad i & \quad i+1 & \quad N-1 & \quad N \\
\end{align*}
\]

BCs. \quad \begin{align*}
  r = 0 & \quad \frac{dC_A}{dr} = 0 \quad \text{(symmetric)} \\
  r = r_p & \quad -D_p \frac{dC_A}{dr} = k_m(C_A - C_b)
\end{align*}

Discretized eqn:

\[
\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \frac{2}{r_i} \frac{y_{i+1} - y_{i-1}}{2h} + A = 0 \quad \quad \text{(1)}
\]

\[
(r_i = ih)
\]

Discretized BCs:

\[
\begin{align*}
  & \begin{array}{c}
    \eta_i = 0 \\
    (i = 0)
  \end{array} & \quad \frac{y_1 - y_{-1}}{2h} = 0 & \begin{array}{c}
    \eta_i = r_p \\
    (i = N)
  \end{array} & \quad -D_p \frac{y_{N+1} - y_{N-1}}{2h} = k_m(y_N - C_b)
\end{align*}
\]

\[
\begin{align*}
  & \begin{array}{c}
    \gamma_{-1} = y_1
  \end{array} & \quad \gamma_{N+1} = \left( y_{N-1} - \frac{2h}{D_p} k_m y_N + \frac{2hk_m}{D_p} C_b \right)
\end{align*}
\]

Before you re-arrange the terms, you should realize that there is a clear discontinuity at \(r_i = 0\) (or 1st node). Therefore, you cannot proceed without removing discontinuity. One way of doing this is to approximate \(\frac{1}{r \frac{dy}{dr}}\) as \(\frac{d^2 y}{dr^2}\) because \(\forall y = 0\) at \(r = 0\)

\[
\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + 2 \frac{y_1 - 2y_0 + y_{-1}}{h^2} + A = 0
\]

\[
\left( 3 \frac{d^2 y}{dr^2} + A \right) = 0
\]

\[
\text{or} \quad \frac{y_{i+1} - 2y_0 + y_{-1}}{h^2} + A = 0 \quad \text{at} \quad i = 0
\]

Eq. (1) is applied over the next nodes, i.e., \(i = 1 \cdots N\) without any discontinuity
Arrange the terms now,

\[
\begin{align*}
  y_{-1} + \left( A \frac{h^2}{3} - 2 \right) y_0 + y_1 &= -A & r &= 0 \\
  \left( \frac{1}{h^2} - \frac{1}{r_1 h} \right) y_0 - \frac{2}{h^2} y_1 + \left( \frac{1}{h^2} + \frac{1}{r_1 h} \right) y_2 &= -A & r &= 1 \\
  \left( \frac{1}{h^2} - \frac{1}{r_2 h} \right) y_1 - \frac{2}{h^2} y_2 + \left( \frac{1}{h^2} + \frac{1}{r_2 h} \right) y_3 &= -A & r &= 2 \\
  \vdots & & \\
  \left( \frac{1}{h^2} - \frac{1}{r_N h} \right) y_{N-1} - \frac{2}{h^2} y_N + \left( \frac{1}{h^2} + \frac{1}{r_N h} \right) y_{N+1} &= -A & r &= N
\end{align*}
\]

Apply the BCs \((i = 0 \text{ and } i = N)\)

\[\begin{align*}
  \text{1st row:} & \quad \left( A \frac{h^2}{3} - 2 \right) y_0 + 2y_1 = -A \\
  \text{Last row:} & \quad \frac{2}{h^2} y_{N-1} - \left( 2 + \frac{2hk_m}{D_p} \times \left( \frac{1}{h^2} + \frac{1}{r_N h} \right) \right) y_N = -A - \frac{2hk_m}{D_p} \left( \frac{1}{h^2} + \frac{1}{r_N h} \right) C_b
\end{align*}\]

- From the ‘preparation of tridiagonal matrix’ point of view,

\[a_0 = 1, b_0 = \left( A \frac{h^2}{3} - 2 \right), c_0 = 1, \quad d_0 = -A\]

and

\[a_i = \frac{1}{h^2} - \frac{1}{r_i h}, \quad b_i = \frac{2}{h^2}, \quad c_i = \frac{1}{h^2} + \frac{1}{r_i h}; \quad d_i = -A \quad ; \quad i = 1 - N\]

However, on the substitution of BCs,

\[c_0 = c_0 + a_0 \quad \text{(1st row @ } i = 0)\]

\[a_N = a_N + c_N, \quad b_N = b_N - c_N \frac{2hk_m}{D_p}, \quad \quad d_N = d_N - c_N \frac{2hk_m}{D_p} \cdot C_b\]

Therefore, \(A \bar{X} = B\). There are \((N + 1)\) equations to solve \((N + 1)\) unknown \((Y_0 \cdots Y_N)\) and \(A\) is the tridiagonal matrix.

**call Tridiagonal \((N + 1, a, b, c, d, y)\) \(i = 0 \cdots N\)**

Write a programming code to solve the SS profiles of \(C_A(r)\) for different \(r_p = 0.25, 0.5\) and 1 mm, \(k_m = 0.01, 0.1, 0.5\) m/s and \(D_p = 10^{-7}, 10^{-9}, 10^{-11}\) m²/s.

Plot the profiles and interpret the results.
Recall the course on chemical reaction engineering. Once $SS C_A(r)$ is calculated, you can calculate the (total) effectiveness factor considering both inter- and intraphase diffusion resistances by computing the total rate of the consumption of the species, $A$ within the catalyst and that corresponding to the bulk (gas) phase concentration. The computations may require numerical integration (maybe, Simpson’s 1/3rd method) to calculate the volume-average quantities and/or numerical differentiation (may be, 2nd order BDS) to calculate the flux at the surface of the sphere.

Ex. Consider the steady-state 1D reactive flow ($Re > 5000$) of a gaseous species $A$ in a long quartz tubular reactor ($1D = D$, length $= L$) radiated by UV light. Velocity in the tube is $V$.

The species $A$ is converted into $B$ by the 1st order homogeneous reaction ($r = kC_A$), as it flows in the tube. The inlet concentration of $A$ is $C_{Ao}$. Determine the axial concentration profiles $C_A(X) =$?. The reaction takes place under isothermal conditions and $\Delta H \approx 0$.

Soln: $\frac{dC_A}{dt} + V \cdot \nabla C_A = D \nabla^2 C_A + (r_A)$

$\nabla \frac{dC_A}{dx} = D \frac{d^2 C_A}{dx^2} - kC_A$

BC. $X = 0 \quad C_A = C_{Ao}, \quad X = L \quad \nabla C_A = 0 \quad \text{or} \quad \frac{dC_A}{dx} = 0 \quad \text{(long tube approximation)}$

$\frac{dC_A}{dx} + B \frac{d^2 C_A}{dx^2} + CC_A = 0, \quad B = -\frac{D}{\overline{V}}; \quad C = \frac{k}{\overline{V}}, \quad \Delta h = \frac{L}{N}$
\[
\frac{y_{i+1} - y_{i-1}}{2h} + B \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + Cy_i = 0, \quad i = 1, N
\]

Discretize BCs
\[y_0 = C_{Ao}; \quad \frac{y_{N+1} - y_{N-1}}{2h} = 0 \Rightarrow y_{N+1} = y_{N-1}\]

Collect the terms
\[
\left(\frac{B}{h^2} - \frac{1}{2h}\right)y_{i-1} - \left(\frac{2B}{h} - C\right)y_i + \left(\frac{B}{h^2} + \frac{1}{2h}\right)y_{i+1} = 0, \quad i = 1, N
\]

Prepare the tridiagonal matrix:
\[
a_i = \left(\frac{B}{h^2} - \frac{1}{2h}\right), \quad b_i = \left(\frac{2B}{h} - C\right), \quad c_i = \left(\frac{B}{h^2} + \frac{1}{2h}\right), \quad d_i = 0; \quad i = 1, N
\]

On substitution of BCs,
\[
d_1 = d_1 - a_1C_{Ao} \quad (i = 1, \ 1^{st} \text{row})
da_N = a_N + c_N \quad (i = N, \ \text{last row})
\]

You have \(A\vec{y} = \vec{b}\), where \(A\) is the tridiagonal matrix.
\[
\vec{y} = 1, N
\]

call Tridiagonal \((N, a, b, c, d, y)\)

Plot (numerical solutions using reasonable values for \(D, \vec{V}, k, C_{Ao}, L\))
Partial Differential Equations (PDEs)

- Parabolic equations

\[
1\text{D} : A \frac{\partial y}{\partial t} + B \frac{\partial^2 y}{\partial x^2} \text{ (or } \frac{B}{r} \frac{\partial}{\partial r} \left( r \frac{\partial y}{\partial r} \right) \text{ or } \frac{B r^2}{\partial^2 r} \left( r^2 \frac{\partial y}{\partial r} \right) \text{)} + C \frac{\partial y}{\partial X} + Dy + E = 0
\]

Such time dependent 1D (spatial) equations are common in heat/mass/momentum transport. The PDE representing conservation of transport variables usually contains a convective term \((V \cdot \nabla T \text{ or } V \cdot \nabla C \text{ or } V \cdot \nabla V)\) and a diffusion term, \((\alpha \nabla^2 T \text{ or } D \nabla^2 C \text{ or } \nu \nabla^2 V)\), besides the unsteady-state or transient term \((\rho C_p \frac{\partial T}{\partial t} \text{ or } \frac{\partial C}{\partial t} \text{ or } \rho \frac{\partial V}{\partial t})\). Therefore,

\[
\begin{align*}
\frac{\partial C_A}{\partial t} + V_x \frac{\partial C_A}{\partial X} &= D \frac{\partial^2 C_A}{\partial X^2} - k C_A \\
\text{or } \rho C_p \left( \frac{\partial T}{\partial t} + V_x \frac{\partial T}{\partial X} \right) &= k \frac{\partial^2 T}{\partial X^2} + (k C_A)(-\Delta H) \\
\text{or } \rho \left( \frac{\partial V_x}{\partial t} + V_x \frac{\partial V_x}{\partial X} \right) &= \mu \frac{\partial^2 V_x}{\partial X^2} - \left( \frac{dp}{dx} \right) + \rho g_x \\
\text{or } \rho C_p \frac{\partial T}{\partial t} &= k \frac{\partial}{\partial X} \left( r^2 \frac{\partial T}{\partial r} \right) \\
\text{or } \frac{\partial C_A}{\partial t} &= D \frac{\partial}{\partial X} \left( r \frac{\partial C_A}{\partial r} \right) - k C_A
\end{align*}
\]

are all examples of the time-dependent 1D parabolic PDE. You may refer the book by Ferziger for the exact mathematical definition of different types of PDE including parabolic, elliptic or hyperbolic.

⇒ Why ‘Parabolic’? Because we march on time ‘endlessly’ covering the space.

⇒ Note: If you drop the time-dependent equation (or the unsteady-state or transient term), what do you get? A boundary value problem (BVP), which we covered in the previous lectures.

⇒ Message is clear. Solving 1D parabolic PDE is not different from solving BVP. The procedure for discretization in the space remains the same as before; it is to be repeated on every time step as you march by refreshing the numerical values from the previous step. In fact, you may like to copy and paste the programming code for BVP and nest it within the time loop. The graphical representation of the numerical calculation is as follows.

March on time

\[
\begin{array}{c}
t = 2 \\
\vdots \\
t = 1 \\
t = 0
\end{array}
\]

\[
\begin{array}{c}
i - 1 & i & i + 1 \\
\vdots \\
i - 1 & i & i + 1 \\
0 & i - 1 & i & i + 1 \\
& & 1
\end{array}
\]

⇒ Why ‘Parabolic’? Because we march on time ‘endlessly’ covering the space.

⇒ Note: If you drop the time-dependent equation (or the unsteady-state or transient term), what do you get? A boundary value problem (BVP), which we covered in the previous lectures.

⇒ Message is clear. Solving 1D parabolic PDE is not different from solving BVP. The procedure for discretization in the space remains the same as before; it is to be repeated on every time step as you march by refreshing the numerical values from the previous step. In fact, you may like to copy and paste the programming code for BVP and nest it within the time loop. The graphical representation of the numerical calculation is as follows.

March on time

\[
\begin{array}{c}
t = 2 \\
\vdots \\
t = 1 \\
t = 0
\end{array}
\]

\[
\begin{array}{c}
i - 1 & i & i + 1 \\
\vdots \\
i - 1 & i & i + 1 \\
0 & i - 1 & i & i + 1 \\
& & 1
\end{array}
\]

⇒ Why ‘Parabolic’? Because we march on time ‘endlessly’ covering the space.
Take a general case:

\[ A \frac{\partial y}{\partial t} + B \frac{\partial y}{\partial X} = C \frac{\partial^2 y}{\partial X^2} + D \] (or \( A \frac{\partial y}{\partial t} = f(y_X, y_{XX}, D) \))

\[
\begin{align*}
& t = 0 \quad y = 0 \text{ for all } L > X > 0 \\
& t = 0^+ \quad X = 0 \quad y = y_0; \quad X = L \quad \frac{\partial y}{\partial X} = 0
\end{align*}
\]

**Steps**

(1) Discretize 'X'-terms as before. On time, apply trapezoidal rule for integration:

\[
A \frac{y_{i+1}^t - y_i^t}{\Delta t} = \frac{1}{2} \left[ -B \left( \frac{y_{i+1}^t - y_{i-1}^t}{2\Delta h} \right)^t - B \left( \frac{y_{i+1}^t - y_{i-1}^t}{2\Delta h} \right)^{t+1} \right]
\]

\[
+ C \left( \frac{y_{i+1}^t - 2y_i^t + y_{i-1}^t}{h^2} \right)^t + C \left( \frac{y_{i+1}^t - 2y_i^t + y_{i-1}^t}{h^2} \right)^{t+1} + D
\]

(Note: ‘D’ is a const. \( \frac{D^t + D^{t+1}}{2} = D! \)) – **This method is better known as Crank-Nicolson.**

Also note that you have used a 2\textsuperscript{nd} order accurate scheme on both time and \( X \).

(2) Collect all terms on \((t + 1)\) on the LHS and \( t \) on the RHS as

\[
Ay^{(t+1)} = d^{(t)}
\]

\[
\begin{align*}
\frac{1}{2} \left( -\frac{C}{h^2} - \frac{B}{2\Delta h} \right) y_{i-1}^{t+1} + \left( \frac{A}{\Delta t} + \frac{2C}{h^2} \right) y_i^{t+1} + \frac{1}{2} \left( -\frac{C}{h^2} + \frac{B}{2\Delta h} \right) y_{i+1}^{t+1} \\
= \frac{1}{2} \left( \frac{C}{h^2} + \frac{B}{2\Delta h} \right) y_{i-1}^t + \left( \frac{A}{\Delta t} - \frac{C}{h^2} \right) y_i^t + \frac{1}{2} \left( \frac{C}{h^2} - \frac{B}{2\Delta h} \right) y_{i+1}^t + D
\end{align*}
\]

\( i = 1, N \)

**Note:** All terms on RHS are known/calculated from the previous time step.
Lecture #22

(.....continue)

(3) Apply BCs on both sides \((t + 1)\) and \((t)\):

\[ i = 0 \quad y_{(o)} = y_o \quad \text{for all } t_s \,(t + 1 \text{ and } t) \]

\[ i = N \quad y_{N+1} = y_{N-1} \quad \text{for all } t_s \,(t + 1 \text{ and } t) \]

1\(^{st}\) row will change as:

\[
\begin{align*}
\left( \frac{A}{\Delta t} + \frac{C}{h^2} \right) y^1 + \frac{1}{2} \left( -\frac{C}{h^2} + \frac{B}{2\Delta h} \right) y^2 + \left( \frac{C}{h^2} + \frac{B}{2\Delta h} \right) y^0 + \left( \frac{A}{\Delta t} - \frac{C}{h^2} \right) y^1 + \frac{1}{2} \left( C \frac{h^2}{h^2} - \frac{B}{2\Delta h} \right) y^2 + D = 0 \quad (i = 1)
\end{align*}
\]

Last row will change as:

\[
\begin{align*}
- \frac{C}{h^2} y^N + \left( \frac{A}{\Delta t} + \frac{C}{h^2} \right) y^N + \frac{C}{h^2} y^N + \left( \frac{A}{\Delta t} - \frac{C}{h^2} \right) y^N + D &= 0 \quad (i = N)
\end{align*}
\]

The other rows \((2 \cdots N - 1)\) will remain the same. Again the coefficient matrix is a tridiagonal one:

\[ Ay^{(t+1)} = d^{(t)} \]

You have prepared the tridiagonal matrix as follows:

\[
\begin{align*}
\alpha_l &= -\frac{1}{2} \left( \frac{C}{h^2} + \frac{B}{2\Delta h} \right), \quad \beta_l = \left( \frac{A}{\Delta t} + \frac{C}{h^2} \right), \quad \gamma_l = \frac{1}{2} \left( -\frac{C}{h^2} + \frac{B}{2\Delta h} \right), \quad \delta_l = \frac{1}{2} \left( \frac{C}{h^2} + \frac{B}{2\Delta h} \right) y^t_{l-1} + \\
\left( \frac{A}{\Delta t} - \frac{C}{h^2} \right) y^t_l + \frac{1}{2} \left( \frac{C}{h^2} - \frac{B}{2\Delta h} \right) y^t_{l+1} + D; \quad &i = 1, N
\end{align*}
\]

on application of BCs \(d_1 = d_1 - a_1 y_0\) \((1^{st}\text{ row}, i = 1)\)

1) Note that \(y_0\) is substituted for \(y^t_{l-1}\) before applying BC.

\[
\begin{align*}
\alpha_N &= \alpha_N + \beta_N \quad \text{(last row, } i = N) \\
d_N &= \frac{C}{h^2} y^N_{N-1} + \left( \frac{A}{\Delta t} - \frac{C}{h^2} \right) y^N_N + D
\end{align*}
\]
Start calculations: RHS is known from $t = 0$ (initial condition)

March on time and call

\[
\begin{align*}
Ay^{(1)} &= b^{(0)} \\
Ay^{(2)} &= b^{(1)} \\
&\vdots \\
Ay^{(t+1)} &= b^{(t)}
\end{align*}
\]

Tridiagonal subroutine at every time step, & update $y_0 \to y^{(1)} \to y^{(2)} \to \ldots$

Tridiagonal $(N, a, b, c, d^t, y^{t+1})$

$\rightarrow i = 1, N$

Convince yourself that the SS values will be the same as solving the corresponding BVP by dropping the transient or unsteady-state term.

Ex:

Consider the SS flow of a liquid through a long tube. Reynolds number is 5000 and a radially flat velocity profile may be assumed. At certain time a tracer is injected into the liquid at the inlet. The dispersion coefficient of the tracer in the liquid may be assumed to be $D \text{ cm}^2/\text{s}$. Radial peclet number is large. Determine unsteady-state or transient axial concentration profiles, $C_A(t, X) = ?$

Soln:

\[
\begin{align*}
\frac{\partial C_A}{\partial t} + V \cdot \nabla C_A &= D \nabla^2 C_A + \frac{r_A}{\rho} \\
\frac{\partial C_A}{\partial t} + V_X \frac{\partial C_A}{\partial X} &= D \frac{\partial^2 C_A}{\partial X^2}; \\
&\text{(no reaction)} \\
&\text{(neglecting radial term)}
\end{align*}
\]

$t = 0, \quad C_A = 0 \quad \text{for all } L > x > 0 \quad \text{(Pure liquid)}$

$0^+, \quad X = 0; \quad C_A = C_{A,in} \quad \text{(Inlet concentration of tracer)}$
\[ X = L, \; \nabla C_A = 0 \text{ (long - tube approximation) or } \frac{\partial C_A}{\partial x} = 0 \]

\[ \frac{\partial C_A}{\partial t} + V x \frac{\partial C_A}{\partial x} = D \frac{\partial^2 C_A}{\partial x^2}; \; V_x = \bar{V} \]

\[ i = 0 \quad i-1 \quad i \quad i+1 \quad N \]

\[ t \quad t+1 \quad \Delta t' \]

\[ \frac{C_{A,i}^{t+1} - C_{A,i}^t}{\Delta t} = \frac{1}{2} \left[ D \left( \frac{C_{A,i+1} - 2C_{A,i} + C_{A,i-1}}{h^2} \right)^{t+1} + D \left( \frac{C_{A,i+1} - 2C_{A,i} + C_{A,i-1}}{h^2} \right)^t \right]
- \bar{V} \left( \frac{C_{A,i+1} - C_{A,i-1}}{2h} \right)^{t+1} - \bar{V} \left( \frac{C_{A,i+1} - C_{A,i-1}}{2h} \right)^t \]

\[ \left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i-1}^{t+1} - \left( \frac{1}{\Delta t} + \frac{D}{h^2} \right) C_{A,i}^{t+1} + \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^{t+1} = - \left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i-1}^t - \left( \frac{1}{\Delta t} - \frac{D}{h^2} \right) C_{A,i}^t - \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^t \]

\[ i = 1, N \]

\[ i = 0: \; C_{A,0} = C_{A,0}^{t+1} = C_{A,in} \text{ (Inlet BC)} \]

\[ i = N: \; C_{A,N+1} = C_{A,N-1}^{t+1} \text{ and } C_{A,N+1} = C_{A,N-1}^t \]

* Prepare the tridiagonal matrix

\[ a_i = \left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right), \; b_i = -\left( \frac{D}{h^2} + \frac{1}{\Delta t} \right), \; c_i = \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right), \; d_i \]

\[ = -\left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i-1}^t - \left( \frac{1}{\Delta t} - \frac{D}{h^2} \right) C_{A,i}^t - \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^t \]

Apply BCs:

\[ d_1 = d_1 - a_1 C_{A,in} : \text{Note that } C_{A,in} \text{ is substituted for } C_{A,i-1}^t \text{ in } d_1. \]

\[ a_N = a_N + c_N \]

\[ d_N = -\left( \frac{1}{\Delta t} - \frac{D}{h^2} \right) C_{A,N}^t - \left( \frac{D}{h^2} \right) C_{A,N-1}^t \]

You have \[ A y^{t+1} = d(t) \]; \( A \equiv \text{Tridiagonal matrix with } i = 1, N \text{ rows} \)
Call Tridiagonal \((N, a, b, c, d, y)\)

Start with \(t = 0\): \(C_A(i = 0, N) = 0\) (pure solvent) or use a small concentration, \(C_{Al} \ll C_{A\text{inlet}}\).

Solve for \(y^{(1)}\) or \(C_A^{(1)}\) at \(t = 1\), recalculate RHS \(d^{(2)}\) and keep on marching on time till a steady-state solution or at \(t = t_{final}\) is reached.

Plot (use \(L = 10\) cm, \(C_{A,i} = 0.001\), \(C_{A\text{inlet}} = 1.0\), \(\bar{V} = 0.1\) cm/s, \(D = 0.01\) cm\(^2\)/s)

\[
\Rightarrow \frac{\partial c}{\partial x} = 0 \bigg|_{x=L}
\]

Note that
- Tridiagonal matrix is called at every time-step.
- \(\vec{b}\) vector is updated with the recent most \(C_A\) values.
- gradient at the exit must be flat (consistent with BC)
- There are two steps \((h & \Delta t)\). How do you choose them?

No one knows! In general \(\Delta t < \frac{\Delta x}{\bar{V}}, \frac{\Delta x^2}{D}\). Why??

One should choose a fixed \(\Delta X\), and then refine \('\Delta t'\) till there is a convergence in the solution.
Again, use \(\Delta X = \Delta X/2\), adjust \('\Delta t'\) till the solution converges, and so forth!

Ex. Repeat the previous problem, assuming that some solute is irreversibly adsorbed at the tube-wall @ \(kC_A\) rate.

Species balance equation:

\[
\frac{\partial c_A}{\partial t} + \bar{V} \frac{\partial c_A}{\partial x} = D \frac{\partial^2 c_A}{\partial x^2} - (kC_A) a
\]

where \(a\) is the specific surface area per unit volume of the tube.

or \[
\frac{\partial c_A}{\partial t} + \bar{V} \frac{\partial c_A}{\partial x} = D \frac{\partial^2 c_A}{\partial x^2} - k'C_A;
\]
IC: \( t = 0 \) \( C_A = C_{A,i} \),

BC  \( 0^+, X = 0; C_A = C_{A,in}, X = L, \ \nabla C_A = 0 \Rightarrow \frac{\partial C_A}{\partial x} = 0 \)

\[
\frac{C_{A,i+1}^{t+1} - C_{A,i}^t}{\Delta t} = \frac{1}{2} D \left( \frac{C_{A,i+1}^{t+1} - 2C_{A,i}^t + C_{A,i-1}^t}{h^2} \right) + D \left( \frac{C_{A,i+1}^{t+1} - 2C_{A,i}^t + C_{A,i-1}^t}{h^2} \right) - \bar{V} \left( \frac{C_{A,i+1}^{t+1} - C_{A,i-1}^t}{2h} \right) - \left( k'C_{A,i}^{t+1} + k'C_{A,i}^t \right)
\]

Note: This is the extra term.

Re-arrange,

\[
\left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i}^{t+1} - \left( \frac{1}{\Delta t} + \frac{D}{h^2} + \frac{k'}{2} \right) C_{A,i}^{t+1} + \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^{t+1} = - \left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i-1}^{t+1} - \left( \frac{1}{\Delta t} - \frac{D}{h^2} - \frac{k'}{2} \right) C_{A,i}^t - \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^t
\]

\[ i = 1, N \]

BCs:
\[
\begin{align*}
  &i = 0, \quad C_{A,0}^t = C_{A,0}^{t+1} = C_{A,in} \quad \text{(for all time steps)} \\
  &i = N, \quad C_{A,N+1}^{t+1} = C_{A,N-1}^{t+1} \quad \text{and} \quad C_{A,N+1}^t = C_{A,N-1}^t
\end{align*}
\]

Prepare tridiagonal matrix:

\[
a_i = \left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right), \quad b_i = -\left( \frac{1}{\Delta t} + \frac{D}{h^2} + \frac{k'}{2} \right), \quad c_i = \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right), \quad d_i
\]

\[
= -\left( \frac{D}{2h^2} + \frac{\bar{V}}{4h} \right) C_{A,i-1}^t - \left( \frac{1}{\Delta t} - \frac{D}{h^2} - \frac{k'}{2} \right) C_{A,i}^t - \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,i+1}^t
\]

Apply BCs.

\[
d_1 = -\left( \frac{D}{h^2} + \frac{\bar{V}}{2h} \right) C_{A,in} - \left( \frac{1}{\Delta t} - \frac{D}{h^2} - \frac{k'}{2} \right) C_{A,1}^t - \left( \frac{D}{2h^2} - \frac{\bar{V}}{4h} \right) C_{A,2}^t : i = 1 \quad \text{(1st row)}
\]

(Note that this step is the same as \( d_1 = d_1 - a_1 C_{A,in} \))

\[
a_N = a_N + c_N
\]

\[
d_N = -\frac{D}{h^2} C_{A,N-1}^t - \left( \frac{1}{\Delta t} - \frac{D}{h^2} - \frac{k'}{2} \right) C_{A,N}^t \quad i = N \quad \text{(last row)}
\]

\[ A y^{(t+1)} = d(t); \ A \text{ is the tridiagonal matrix.} \]
Proceed as before with initial condition $C_A(i = 1, N) = C_{A,i}$. Evaluate $d^{(t)}$, solve for $y^{(t+1)}$, evaluate $d^{(t+1)}$, solve for $y^{(t+2)}$ ... etc. Subroutine Tridiagonal $(N, a, b, c, d, y)$ is called at every time step. Plot qualitatively for the same conditions as before, use $k = 0.1 \text{ s}^{-1}$.

The axial concentration profiles for different $t_s$ will be the similar as before; however increase in concentration with $t$ will be less than before because of the reactive/adsorptive wall.
Example: Consider a spherical \( d_p = 0.1 \text{ cm} \) steel pellet \( (k = 40 \text{ W/m} \cdot \text{K}, \rho = 8000 \text{ kg/m}^3, C_p = 400 \text{ J/kg} \cdot \text{K}) \). The initial temperature of the pellet is 300 K. It is immersed in a large oil tank at 400 K. The corrective heat transfer coefficient, \( h_f \) at the sphere surface is 3000 \( W/m^2 K \). Solve for unsteady-state temperature profiles in the sphere. Choose \( \Delta t = 1 \text{ s} \). Determine the average temperature in the sphere and the rate of which the surface temperature decreases at \( t = 20 \text{ s} \).

* Recall: There is a discontinuity at \( i = 0 (r = ih) \)

However, the grad \( \nabla T = 0 \) at \( r = 0 \). Therefore, \( \frac{1}{r} \frac{\partial T}{\partial r} \approx \frac{\partial^2 r}{\partial r^2} \)

![Diagram of heat transfer](image)

Energy balance equation:

\[
\rho C_p \frac{\partial T}{\partial t} = k \nabla^2 T + S_{\Delta} \quad (\alpha = \frac{k}{\rho C_p})
\]

\[
\frac{\partial T}{\partial t} = \frac{\alpha}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial r}{\partial r} \right) \quad r_p > r \geq 0
\]

IC: \( t = 0 \quad T = 300 K(T_o) \) for all \( r_p \geq r \geq 0 \)

BC: \( 0^+ \quad r = 0 \quad \nabla T = 0 \quad \text{or} \quad \frac{\partial T}{\partial r} = 0 \)

\[
r = r_p \quad - k \frac{\partial T}{\partial r} = h(T - T_f)
\]

or \( k \frac{\partial T}{\partial r} = h(T_f - T) \)
The equation is re-written as \( \frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial r^2} + \frac{2a \partial T}{r \partial r} \)

\[ \frac{T^{t+1}_i - T^t_i}{\Delta t} = \frac{1}{2} \left[ \alpha \left( \frac{T^{t+1}_{i+1} - 2T^t_i + T^{t+1}_{i-1}}{h^2} \right)^{t+1} + \alpha \left( \frac{T^{t+1}_{i+1} - 2T^t_i + T^{t+1}_{i-1}}{h^2} \right)^t + \frac{2a}{r_i} \left( \frac{T^{t+1}_{i+1} - T^{t+1}_{i-1}}{2h} \right)^{t+1} \right] + \frac{2a}{r_i} \left( \frac{T^{t+1}_{i+1} - T^{t+1}_{i-1}}{2h} \right)^t \]

Arrange: \( \frac{(\alpha/2h^2) T^{t+1}_{i-1} - (\alpha/2r_i h) T^t_i + (1/\Delta t + \alpha/h^2) T^{t+1}_i + (\alpha/2h^2 + \alpha/2r_i h) T^{t+1}_{i+1} = (\alpha/2r_i h - \alpha/2h^2) T^t_{i-1} + (\alpha/2h^2 + \alpha/2r_i h) T^t_{i+1}} {i = 1, N} \)

Apply Crank-Nicholson discretization scheme:

\[ \frac{T^{t+1}_i - T^t_i}{\Delta t} = \frac{3a}{2} \left[ \left( \frac{T^{t+1}_{i-1} - 2T^t_i + T^{t+1}_{i+1}}{h^2} \right)^{t+1} + \left( \frac{T^{t+1}_{i-1} - 2T^t_i + T^{t+1}_{i+1}}{h^2} \right)^t \right] : i = 0 \]

\[ \left( \frac{3a}{2h^2} \right) T^{t+1}_{i-1} - \left( \frac{3a}{2h^2} + \frac{1}{\Delta t} \right) T^t_i + \left( \frac{3a}{2h^2} \right) T^{t+1}_i = - \left( \frac{3a}{2h^2} \right) T^t_{i-1} + \left( \frac{3a}{2h^2} - \frac{1}{\Delta t} \right) T^t_i - \left( \frac{3a}{2h^2} \right) T^t_{i+1} ; i = 0 \]

Prepare tridiagonal matrix:

\[
\begin{align*}
a_0 &= \frac{3a}{2h^2}, & b_0 &= -(\frac{3a}{h^2} + \frac{1}{\Delta t}), & c_0 &= \frac{3a}{2h^2} \\
d_0 &= -\left(\frac{3a}{2h^2}\right) T^t_i + \left(\frac{3a}{h^2} - \frac{1}{\Delta t}\right) T^t_i - \left(\frac{3a}{2h^2}\right) T^t_i \\
a_i &= \left(\frac{\alpha}{2h^2} - \frac{\alpha}{2r_i h}\right), & b_i &= -\left(\frac{\alpha}{h^2} + \frac{1}{\Delta t}\right), & c_i &= \left(\frac{\alpha}{2h^2} + \frac{\alpha}{2r_i h}\right), & d_i &= \left(\frac{\alpha}{2r_i h} - \frac{\alpha}{2h^2}\right) T^t_i + \left(\frac{\alpha}{h^2} - \frac{1}{\Delta t}\right) T^t_i - \left(\frac{\alpha}{2h^2} + \frac{\alpha}{2r_i h}\right) T^t_{i+1} \\
i &= 1, N
\end{align*}
\]

Discretize BCs:

\( 1 \) \( r = 0 \frac{\partial T}{\partial r} = 0 \Rightarrow i = 0; \quad T_{-1} = T_{+1} \)

\( 2 \) \( r = r_p \quad k \frac{\partial T}{\partial r} = h_f (T_f - T) \quad i = N; \)
\[ \frac{T_{N+1} - T_{N-1}}{2h} = \frac{h_f}{k} (T_f - T_N) \]

or, \[ T_{N+1} = T_{N-1} + \frac{2h_f h}{k} (T_f - T_N) \]

Apply BC

\[ d_0 = \left( \frac{3a}{h^2} - \frac{1}{\Delta t} \right) T_0^t - \left( \frac{3a}{h^2} \right) T_1^t \] \quad i = 0

\[ a_N = a_N + c_N \]

\[ d_N = -\frac{a}{h^2} T_{N-1}^t + \left( \frac{a}{h^2} - \frac{1}{\Delta t} \right) T_N^t + \left( \frac{a}{2h^2} - \frac{a}{2r_i h} \right) \frac{2h_f h}{k} T_N^t - \left( \frac{a}{2h^2} + \frac{a}{2r_i h} \right) \frac{2h_f h}{k} T_f \]

Call Tridiagonal(N+1, a, b, c, d, y)

Solve \( Ay^{(i+1)} = d^{(i)} \), \( i = 0, N \) with initial condition: \( T = T_{in} = 300 \)

\( Ay^{(1)} = d^{(0)} \) \( \Rightarrow Ay^{(2)} = d^{(1)} \) \( \Rightarrow Ay^{(3)} = d^{(2)} \), etc.

Once the functional values or \( y(t, r) \) or \( y_i^t \), \( i = 0, N \) are computed, the **average temperature** of the sphere at any time, \( t \) can be calculated as the volume-average quantity (why?), as follows:

\[ 3 \sum r_i^2 \Delta r / R^3, \quad i = 0, N \] \( \text{and} \ r_i = ih \) (assuming the constant or average thermophysical properties i.e., \( \rho, C_p \) of the material). You can use the Simpson’s 1/3rd or Trapezoidal rule of integration.

Similarly, the **rate of decrease of the surface temperature**, viz. \( \frac{\partial T}{\partial t} @ r = R \) at any time, \( t \) can be calculated using CDS as \( \frac{y(t_{i+1}) - y(t_{i-1})}{2\Delta t} \) or BDS as \( \frac{4y(t_{i+1}) - 3y(t_i) - y(t_{i-2})}{2\Delta t} \) or FDS as \( \frac{-4y(t_{i-1}) + 3y(t_i) + y(t_{i+2})}{2\Delta t} \), depending upon the **time**, \( t \). Therefore, you must use FDS at the initial time, \( t = 0 \), etc.
**Example:** Consider the catalytic oxidation of $SO_2$ into $SO_3$ over a spherical Pt-dispersed porous carbon catalyst ($\text{radius} = r_p$). The atmospheric concentration of $SO_2$ is 1%. The rate of reaction is $r = k (= 0.1) C_A \text{ mole/s} - \text{m}^3$. The film mass transfer coefficient is $k_m (\text{m/s})$. Determine the time-development of concentration profiles within the catalyst. The pore diffusion coefficient of $SO_2$ is estimated to be $D \text{ m}^2/\text{s}$.

\[
\frac{\partial C_A}{\partial t} = \frac{D}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial C_A}{\partial r} \right) - k C_A \tag{1}
\]

$t = 0 \quad C_A = C_{Ai} \ll C_b \quad \text{for all } r_p \geq r \geq 0$

\[
0^+ \quad \nabla C_A = 0 \quad @ \quad r = 0
\]

\[
-D \frac{\partial C_A}{\partial r} = k_m (C_A - C_b) \quad @ \quad r = r_p
\]

Before you solve, two clarifications follow:

1. The steady-state solution of eq(1) is the same as that of the equation solved in the previous lectures for BVP. Therefore, one way of checking the present code for this unsteady-state 1D parabolic equation is to compare its SS solution with that from the BVP-code. In the latter lectures you will see that very often it is better to artificially introduce the transient term (viz. $\frac{\partial T}{\partial t}, \frac{\partial C}{\partial t}, \frac{\partial V}{\partial t}$) in the BVPs and solve the entire time-profiles, even when one is asked to solve the SS solution only. This strategy is often followed for solving elliptic PDE.

2. You should also compare this mass transport example with the previous example on heat transfer. But for the non-homogenous term ($k C_A$), two equations are identically the same, including the IC and BCs.

Recall:
\[
\frac{\partial \tau}{\partial t} = \frac{\alpha}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial \tau}{\partial r} \right) \quad \frac{\partial C_A}{\partial t} = \frac{D}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial C_A}{\partial r} \right)
\]

\begin{align*}
IC: & \quad t = 0 \quad T = T_o \\
BC 1: & \quad r = 0 \quad \frac{\partial \tau}{\partial r} = 0 \\
2: & \quad r = r_p \quad -k \frac{\partial \tau}{\partial r} = h_f(T - T_f) \quad -D \frac{\partial C_A}{\partial r} = k_m(C_A - C_b)
\end{align*}

All it means is that their non-dimensionalized forms are identically the same:

\[
\frac{\partial \theta}{\partial \tau} = \frac{1}{\xi^2} \frac{\partial}{\partial \xi} \left( \xi^2 \frac{\partial \theta}{\partial \xi} \right)
\]

where,

\[
\begin{align*}
\theta &= \frac{T - T_o}{T_f - T_o}, \quad \frac{C_A - C_Ai}{C_{A,b} - C_{A,i}} \\
\xi &= \frac{r}{r_p} \\
\tau &= \frac{t}{t_c}; \quad t_c = \frac{r_p^2}{\alpha}, \quad \frac{r_p^2}{D} \\
\end{align*}
\]

\begin{align*}
IC. & \quad \tau = 0 \quad \theta = 0 \\
BC 1. & \quad \xi = 0 \quad \frac{\partial \theta}{\partial \xi} = 0 \\
2. & \quad \xi = 1 \quad -\frac{\partial \theta}{\partial \xi} = A(\theta - 1), \text{where } A = Nu \text{ or Sh}
\end{align*}

Therefore, it is clear that their (non-dimensional) solutions will also be the same.

It also follows that one should non-dimensionalize the transport equations, as a good practice, before solving the equations. Apart from learning about transport phenomena from the analogy between heat, mass and momentum, there is a possibility that the non-dimensional forms of the conservation equations & the respective bcs being the same, the non-dimensionalized solutions will also be the same. In such cases or similar cases, the programming code written for one problem will be the same or similar, requiring small modifications.

Let us continue with the non-dimensionalized form of the present mass transport problem:

\[
\frac{\partial \theta}{\partial \tau} = \frac{1}{\xi^2} \frac{\partial}{\partial \xi} \left( \xi^2 \frac{\partial \theta}{\partial \xi} \right) - B\theta - C, \quad \text{where } B = kt_c, \quad C = kt_c \frac{C_{A,i}}{C_{A,b} - C_{A,i}}
\]

Discretize:
\[
\frac{\theta_i^{t+1} - \theta_i^t}{\Delta \tau} = \frac{1}{2} \left[ \left( \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{\Delta \xi^2} \right)^{t+1} + \left( \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{\Delta \xi^2} \right)^t + \frac{2}{\xi_i} \left( \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta \xi} \right)^{t+1} \right] + \frac{2}{\xi_i} \left( \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta \xi} \right)^t - B(\theta_i^{t+1} + \theta_i^t) - C
\]

\[i = 0, N\]

@\(i = 0\) there is a discontinuity in the equation. Therefore,

\[
\frac{\theta_0^{t+1} - \theta_0^t}{\Delta \tau} = \frac{3}{2} \left[ \left( \frac{\theta_{1} - 2\theta_0 + \theta_{-1}}{\Delta \xi^2} \right)^{t+1} + \left( \frac{\theta_{1} - 2\theta_0 + \theta_{-1}}{\Delta \xi^2} \right)^t + \frac{B}{2} \left( \theta_0^{t+1} + \theta_0^t \right) \right] - C \quad \text{for } i = 0
\]

Apply BCs

\[
\begin{cases}
  i = 0 & \theta_{-1} = \theta_1 \\
  i = N & -\frac{\theta_{N+1} - \theta_{N-1}}{2\Delta \xi} = A(\theta_N - 1) \\
  \text{or} & \theta_{N+1} = \theta_{N-1} - 2A\Delta \xi (\theta_N - 1)
\end{cases}
\]

\[i = 0:\]

\[
\left( \frac{1}{\Delta \tau} + \frac{3}{\Delta \xi^2} - \frac{B}{2} \right) \theta_0^{t+1} - \frac{3}{\Delta \xi^2} \theta_1^{t+1} = \left( \frac{1}{\Delta \tau} - \frac{3}{\Delta \xi^2} + \frac{B}{2} \right) \theta_0^t + \frac{3}{2\Delta \xi^2} \theta_1^t - C
\]

\[i = N:\]

\[
\left( \frac{1}{\Delta \xi^2} \right) \theta_{N-1}^{t+1} - \left( \frac{1}{\Delta \xi^2} + \frac{B}{2} + \frac{1}{\Delta \tau} \right) \theta_N^{t+1} = -\left( \frac{1}{\Delta \xi^2} \right) \theta_{N-1}^t - \left( \frac{1}{\Delta \tau} - \frac{1}{\Delta \xi^2} - \frac{B}{2} \right) \theta_N^t + C
\]

\[i = 1, N - 1:\]

\[
\left( \frac{1}{2\Delta \xi^2} - \frac{1}{2\Delta \xi \xi} \right) \theta_{i-1}^{t+1} - \left( \frac{1}{\Delta \tau} + \frac{1}{\Delta \xi^2} - \frac{B}{2} \right) \theta_i^{t+1} + \left( \frac{1}{2\Delta \xi^2} - \frac{1}{2\Delta \xi \xi} \right) \theta_{i+1}^{t+1}
\]

\[
= \left( -\frac{1}{2\Delta \xi^2} + \frac{1}{2\Delta \xi \xi} \right) \theta_{i-1}^t - \left( \frac{1}{\Delta \tau} - \frac{1}{\Delta \xi^2} - \frac{B}{2} \right) \theta_i^t + \left( -\frac{1}{2\Delta \xi^2} - \frac{1}{2\Delta \xi \xi} \right) \theta_{i+1}^t + C
\]

\[d\]

Solve \( A\theta^{t+1} = d^t, i = 0, N \) with \( lC \, \theta = 0 \) @ \( \tau = 0 \)

(same as before, \( A \equiv \) Tridiagonal matrix)

Here, we skipped the steps for preparing tridiagonal matrix and directly substituted discretized BCs into the discretized equations! As an exercise, prepare tridiagonal matrix and see if you get the same discretized equations post substitution of BCs, as before.

**Quiz III**
Lecture #24-25
Elliptic PDE (Method of Lines)

The model equations may be recognized by the following examples:

\[ (1) \, k \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + S = 0; \quad S \equiv I^2 R/\psi \cdot \text{cal}/s - m^3 \]

\[ (2) \, k \left( \frac{\partial^2 T}{\partial x^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial T}{\partial r} \right) \right) + S = 0 \]

--- SS 2D temperature profiles in a (1) rectangular plate and (2) cylindrical wire because of uniform heating

\[ (3) \, V_x \frac{\partial c}{\partial x} = D \frac{\partial^2 c}{\partial x^2} + \frac{D}{r} \left( r \frac{\partial c}{\partial r} \right) - kC \]

---- SS 2D concentration distributions of a solute in the reactive flow in a tube

\[ (4) \, V_x \frac{\partial \tau}{\partial x} = \alpha \frac{\partial^2 \tau}{\partial x^2} + \alpha \frac{\partial^2 \tau}{\partial y^2} \]

- SS 2D temperature distributions in a flow through rectangular channel.

\[ (5) \, \rho V_x \frac{\partial v_x}{\partial x} = \mu \left( \frac{\partial^2 v_x}{\partial x^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial v_x}{\partial r} \right) \right) - \frac{dp}{dx} \]

- SS 2D velocity profiles of an incompressible NF in a pressure-driven horizontal flow.

By now, you must have realized that we are referring to an elliptic PDE which describes a SS 2D (no time-dependent term) heat/mass/momentum transport. Why is the equation called elliptic? Because one has to solve the entire 2D space. Considering that there is the SS consideration, there is no ‘marching’ on time as such and the updating of the solutions from the previous time-step, as we earlier discussed for the parabolic PDEs. The solution in this case must be sought one-time only under SS conditions. You will see later that solving elliptic equations is computational extensive requiring iterations.

- The best way of understanding the numerical technique for solving an elliptic PDE is to directly take the example (1) above.
Ex: A rectangular plate \((L \times w \times h)\) fabricated from stainless steel \((k = 40 \text{ W/m} - \text{k}, \rho = 8000 \text{ kg/m}^3, C_p = 400 \text{ J/kg} \cdot \text{K})\) is uniformly heated using an electric power source \(100 \text{ W}\). The top and bottom ends are insulated, whereas the side surfaces are exposed to atmosphere \((T_a = 30^\circ C, h = 100 \text{ W/m}^2 \cdot \text{k})\). Determine the SS temperature profiles in the plate.

2D energy balance over \(\Delta x \Delta y h \cdot CV\) under SS:

\[
\rho C_p \left( \frac{\partial T}{\partial t} + V \cdot \nabla T \right) = k \nabla^2 T + S; \quad W/m^3
\]

\[
SS: \quad 0, \text{solid}
\]

\[
k \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + S = 0; \quad S = 100 \text{ W/m}^3 \text{ of the plate volume}
\]

BCs: \(x = 0 \quad \text{for } w > y > 0; \quad -k \frac{\partial T}{\partial x} = -h_f (T - T_a)
\]

\(= L \quad \text{for } w > y > 0; \quad -k \frac{\partial T}{\partial x} = h_f (T - T_a)
\]

\(y = 0 \quad \text{and } w \text{ for } L > x > 0, \quad -k \frac{\partial T}{\partial y} = 0\) (insulation)
Computational molecule:

\[ \Delta x = \frac{L}{N} \quad \Delta y = \frac{W}{M} \]

\[ i = 0 \ldots N \}
\[ j = 0 \ldots M \]

Discretize equations over \((i,j)\). Note that it is a 2D problem. Therefore, one will have to discretize in both directions \((x,y)\) using steps \(\Delta x\) and \(\Delta y\), respectively. They need not be equal.

\[
k \left[ \frac{\partial^2 T}{\partial x^2} \right]_{i,j} + \left[ \frac{\partial^2 T}{\partial y^2} \right]_{i,j} + S_{i,j} = 0
\]

\[
\left( \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} \right) + \left( \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2} \right) + \frac{S_{i,j}}{k} = 0
\]

(As expected, when discretizing \(\frac{\partial^2 T}{\partial x^2}, \ j \) is constant, and when discretizing \(\frac{\partial^2 T}{\partial y^2}, \ i \) is constant)

There are two ways to arrange the discretized terms to

\[ A\vec{y} = \vec{b} \ form: \]

\[
(1) \frac{1}{\Delta x^2} \left( T_{i-1,j} - 2T_{i,j} + T_{i+1,j} \right) = -\frac{1}{\Delta y^2} \left( T_{i,j-1} - 2T_{i,j} + T_{i,j+1} \right) - \frac{S_{i,j}}{k}
\]

\[
(2) \frac{1}{\Delta y^2} \left( T_{i,j-1} - 2T_{i,j} + T_{i,j+1} \right) = -\frac{1}{\Delta x^2} \left( T_{i-1,j} - 2T_{i,j} + T_{i+1,j} \right) - \frac{S_{i,j}}{k}
\]

**Notes:**

1. Although the linear algebraic equations have taken the form of \(A\vec{y} = \vec{b}\) in both cases, \(\vec{b}\) or RHS terms are not known. Therefore, in principle \(\vec{v}\) or \(\vec{T}\) cannot be solved by inverting the 'matrix' formed on LHS, the way we did in the previous cases. In other words, there is no way one can march in \(x\) or \(y\) direction and solve the unknown variables, because marching in either direction \((i\ or\ j)\) creates unknown variables in the other direction \((j\ or\ i)\). In fact, one has to solve the 'entire space' at one time (without marching). This is the problem in solving an elliptic PDE!

2. The only way to solve an elliptic PDE or \(A\vec{y} = \vec{b}\), where \(A\) is the tridiagonal matrix, is by making guess for all variables \((T_{i,j})\) to start with, so that \(\vec{b}\) (RHS term) is known. Then, \(\vec{y}\) (which is similar to \(\vec{b}\)) can be solved. Next, compare the newly calculated values with the guess values and iterate till there is the convergence:
\[ Ay^{k+1} = b^k \]

\[ b^0 = \text{initial guess for } (T_{ij}) \]

\[ k = \text{# of iterations} \]

(3) Either of the two discretized schemes (1) & (2) can be used to solve \( T_{ij} \). In general, one should sweep the discretized set of equations in the direction the expected solution (functional value) is lesser stiff than in the other direction. In the present example, \( w \ll L \) and the y-ends are insulated. One should use scheme (1). The convergence will be relatively faster.

\[
\frac{1}{\Delta x^2} \left(T_{i-1,j} - 2T_{i,j} + T_{i+1,j}\right) = -\frac{1}{\Delta y^2} \left(T_{i,j-1} - 2T_{i,j} + T_{i,j+1}\right)^g - \frac{S_{ij}}{k}
\]

Prepare tridiagonal matrix:

\[
\begin{align*}
a(i,j) &= \frac{1}{\Delta x^2} ; b(i,j) = -\frac{2}{\Delta x^2} , \quad c(i,j) = \frac{1}{\Delta x^2} \\
d(i,j) &= -\frac{1}{\Delta y^2} \left(T_{i,j-1} - 2T_{i,j} + T_{i,j+1}\right)^g - \frac{S_{ij}}{k}
\end{align*}
\]

The superscript "g" stands for the guess values. Once the initial guess is made, one can now march along 'j' direction, solving \( T_{ij} \) at every 'jth' step as \( A\bar{y}_j = b \), where \( A \) is the tridiagonal matrix containing the discretized 'y' along 'i' direction for a fixed \( j \), and \( \bar{b} \) is known from the guess.

Discretize BCs

\[
\begin{align*}
(i = 0), \quad & j: \quad -k \frac{T_{1,j} - T_{-1,j}}{2\Delta x} = -h_f (T_{0,j} - T_a) \\
\text{or} \quad & T_{-1,j} = T_{1,j} - \frac{2h_f \Delta x}{k} (T_{0,j} - T_a) \\
(i = N), \quad & j: \quad T_{N+1,j} = T_{N-1,j} - \frac{2h_f \Delta x}{k} (T_{N,j} - T_a)
\end{align*}
\]

\((j = 0), \quad i: \quad T_{i,-1} = T_{i,1}\)

\((j = M), \quad i: \quad T_{i,M+1} = T_{i,M-1}\)

Apply BCs

\[ j = 0 \]
\begin{align*}
\begin{aligned}
c(0,0) &= c(0,0) + a(0,0) \\
b(0,0) &= b(0,0) - \frac{2h_f \Delta x}{k} a(0,0) \\
d(0,0) &= \frac{2}{\Delta y^2} (T_{0,0} - T_{0,1})^g - \frac{S_{0,0}}{k} - \frac{2h_f \Delta x}{k} T_a \cdot a(0,0) \\
d(i,0) &= \frac{2}{\Delta y^2} (T_{i,0} - T_{i,1})^g - \frac{S_{i,0}}{k}, \quad i = 1, N - 1
\end{aligned}
\end{align*}

\begin{align*}
\begin{aligned}
a(N, 0) &= a(N, 0) + c(N, 0) \\
b(N, 0) &= b(N, 0) - \frac{2h_f \Delta x}{k} c(N, 0) \\
d(N, 0) &= \frac{2}{\Delta y^2} (T_{N,0} - T_{N,1})^g - \frac{S_{N,0}}{k} - \frac{2h_f \Delta x}{k} T_a \cdot a(N, 0)
\end{aligned}
\end{align*}

\begin{align*}
\begin{aligned}
c(0, j) &= c(0, j) + a(0, j) \\
b(0, j) &= b(0, j) - \frac{2h_f \Delta x}{k} a(0, j) \\
d(0, j) &= d(0, j) - \left(\frac{2h_f \Delta x}{k}\right) T_a a(0, j)
\end{aligned}
\end{align*}

\[
\Rightarrow \text{No coefficients will change for the rows, } i = 1, N - 1, \text{ because all of these correspond to interior nodes:}
\]

\[
\begin{tikzpicture}
    \node at (0,0) {0,0};
    \node at (4,0) {N};
    \node at (0,4) {0};
    \node at (4,4) {N - 1};
    \node at (2,4) {M - 1};
    \node at (4,2) {M};
    \draw [->] (0,0) -- (4,0);
    \draw [->] (0,0) -- (0,4);
    \draw [dashed] (0,0) -- (4,4);
\end{tikzpicture}
\]
\[
\begin{align*}
\alpha(N,j) &= \alpha(N,j) + c(N,j) \\
\beta(N,j) &= \beta(N,j) - \left(\frac{2h_f \Delta x}{k}\right) c(N,j) \\
\gamma(N,j) &= \gamma(N,j) - \left(\frac{2h_f \Delta x}{k}\right) T \alpha c(N,j)
\end{align*}
\]

\[
\text{Tridiag}(N + 1, \alpha, \beta, \gamma, \delta, T_{i,j}) \quad (i = 0, N)
\]

\[
\begin{align*}
\delta(0,j) &= \delta(0,j) + \alpha(0,j) \\
\beta(0,j) &= \beta(0,j) - \frac{2h_f \Delta x}{k} \alpha(0,j) \\
\gamma(0,j) &= \frac{2}{\Delta y^2} \left(-T_{0,M-1} + T_{0,M}\right) - \frac{S_{0,M}}{k} - \left(\frac{2h_f \Delta x}{k}\right) T \alpha \delta(0,j)
\end{align*}
\]

\[
\begin{align*}
\beta(i,j) &= \beta(0,j) - \frac{2h_f \Delta x}{k} \alpha(0,j) \\
\gamma(i,j) &= \frac{2}{\Delta y^2} \left(-T_{i,M-1} + T_{i,M}\right) - \frac{S_{i,M}}{k} - \left(\frac{2h_f \Delta x}{k}\right) T \alpha \gamma(i,j)
\end{align*}
\]

\[
\text{Tridiag}(N + 1, \alpha, \beta, \gamma, \delta, T_{i,j}) \quad (i = 0, N)
\]

You have, therefore, called Tridiagonal subroutine ‘M + 1’ times as you marched along ‘j’ direction. At the end, entire ‘x – y’ grids have been solved for \( T_{i,j} \) (\( i = 0, N; j = 0, M \)). Compare solved values with the guess values and keep iterating till there is a convergence.

\[\Rightarrow\] Solving an elliptic PDE is indeed computational extensive because there is a convergence issue using the guesses.

\[\Rightarrow\] There is another twist in solving an elliptic PDE. Do you update your guess values of \( \bar{b} \) (RHS terms) at ‘j’ with the ones solved at ‘j – 1’ as you march in y-direction, or do you wait till you have solved till ‘M’ (the last boundary)? This question should remind you of the G-S and Jacobi iterations. Choice is yours. A fast convergence is the criterion. Also, the relaxation factor, ‘w’ can be used to update the functional values in either case: \( T_{i,j} = wT'_{i,j} + (1 - w)T_{i,j} \), where \( T' \) and \( T \) are the new and old values, respectively.
Ex 2: Consider the fully developed SS flow \((Re = 200)\) of an incompressible NF in a long horizontal tube \((L, D)\). The inlet temperature of the liquid is \(T_o\). The heat is supplied to the flowing liquid at constant flux \(q_w (W/m^2)\) through the tube walls. Determine the 2D \((r, x)\) SS temperature profiles in the tube.

Energy balance over \('2\pi r\Delta r\Delta x' CV:\n\rho C_p V_x \nabla T = k \nabla^2 T + \frac{q_w}{L} \quad (J or cal/s - m^3)\n
\(V(r) \frac{\partial T}{\partial x} = \alpha \left( \frac{\partial^2 T}{\partial x^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial T}{\partial r} \right) \right) \quad \alpha = \frac{k}{\rho c} ;\n
BC. \quad X = 0 \quad T(r) = T_o \quad R \geq r \geq 0 \nX = L \quad + \frac{\partial T}{\partial x} = 0 \quad \text{(long tube approximation)} \n\frac{\partial T}{\partial r} = 0 \quad \text{(symmetric BC)} \n\frac{\partial T}{\partial r} = -q_w \quad (W/m^2) \quad L \geq z \geq 0 \nor \quad \frac{\partial T}{\partial r} = q_w \n
Let us non-dimensionalize the equation & BC
\(\theta = \frac{T}{T_o}, \quad z = \frac{x}{L_c} \quad \text{where} \quad L_c = \frac{R P_e}{C_{p, \text{radial}}} \quad \frac{U_{\text{max}} R}{\alpha}, \quad \xi = \frac{r}{R} \n\frac{U_{\text{max}} (1 - \xi^2) \frac{\partial \theta}{\partial z}}{L_c} = \alpha \left( \frac{1}{L_c^2} \frac{\partial^2 \theta}{\partial z^2} + \frac{1}{R^2} \frac{\partial^2 \theta}{\partial \xi^2} + \frac{1}{R^2} \frac{\partial \theta}{\partial \xi} \right)
or

\[
(1 - \xi^2) \frac{\partial \theta}{\partial z} = \frac{1}{Pe^2} \frac{\partial^2 \theta}{\partial \xi^2} + \frac{1}{\xi} \frac{\partial \theta}{\partial \xi}
\]

BCs:

\[
\begin{align*}
z &= 0 & \theta &= 1 & + \frac{\partial \theta}{\partial z} &= 0 \\
\xi &= 0 & \frac{\partial \theta}{\partial \xi} &= 0 \\
\xi &= 1 & \frac{\partial \theta}{\partial \xi} &= \left(\frac{q_{wR}}{k_{T_0}}\right) = H(\text{constant}) & \text{for } L/L_c > z > 0
\end{align*}
\]

Before solving, let us consider another example on mass transport.

Ex. Consider the fully developed SS flow \((Re = 200)\) of an incompressible NF in a long horizontal tube \((L,D)\). The inlet concentration of the species A in the liquid is \(C_{Ao}\). The species are catalytically destroyed at the tube walls by the zero\(^{th}\) order chemical reaction \((k \text{ mole/s m}^3)\). Determine the 2D \((r,X)SS\) concentration profiles in the tube.

\[
\begin{align*}
\text{Species balance over } & (2\pi r \Delta r \Delta X) C, V. \\
V_X \frac{\partial C_A}{\partial X} &= D \left( \frac{\partial^2 C_A}{\partial X^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial C_A}{\partial r} \right) \right) \text{ moles/s m}^3 \\
U_{max} \left(1 - \frac{r^2}{R^2}\right) \frac{\partial C_A}{\partial X} &= D \left( \frac{\partial^2 C_A}{\partial X^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial C_A}{\partial r} \right) \right) \\
&= D \left( \frac{\partial^2 C_A}{\partial X^2} + \frac{\partial^2 C_A}{\partial r^2} + \frac{1}{r} \frac{\partial C_A}{\partial r} \right)
\end{align*}
\]

BCs. \(X = 0\) \(C_A = C_{Ao}, X = L\) \(\frac{\partial C_A}{\partial x} = 0 \ (R > r > 0)\)

\[
\begin{align*}
r &= 0 \ & \frac{\partial C_A}{\partial r} &= 0 \ & r &= R \ & \ -D \frac{\partial C_A}{\partial r} &= k \ (L > X > 0)
\end{align*}
\]

Let us non-dimensionalize the equation & BCs:
\[ \theta = \frac{C_A}{C_{Ao}}, \quad z = \frac{x}{L_c}, \quad \xi = \frac{r}{R}, \quad L_c = R P_e, \quad \text{where } P_e(\text{radial}) = \frac{U_{max} R}{D} \]

(Note that this 'Pe' is based on mass transport)

\[
(1 - \xi^2) \frac{\partial \theta}{\partial x} = \frac{1}{P_e^2} \frac{\partial^2 \theta}{\partial \xi^2} + \frac{1}{\xi} \frac{\partial \theta}{\partial \xi}
\]

BC. \( z = 0, \ \theta = 1 \); \( z = \frac{L_c}{L}, \ \frac{\partial \theta}{\partial z} = 0 \)

\[ \xi = 0, \ \frac{\partial \theta}{\partial \xi} = 0 \]; \( \xi = 1 \) \(-\frac{\partial \theta}{\partial \xi} = \left( \frac{kR}{D C_{Ao}} \right) = M' \) (constant)

⇒ It is clear that the non-dimensionalized equations and BCs of this (mass transport) and previous (heat transport) examples are the same. Therefore, you need to solve only one of the two. The dimensionless solutions will be the same. This situation should remind you of the recommendation that one should non-dimensionalize the equation before solving it. Several such analogous heat, mass, and momentum transport scenarios exist in chemical engineering applications.

⇒ In most cases, radial \( P_e(\text{mass or heat}) \) number in laminar flow regime is of the order of 10 or higher. In such cases, the axial diffusion term can be neglected. In other words,

\[ \alpha \frac{\partial^2 T}{\partial X^2} \ll V_X \frac{\partial T}{\partial X} \quad \text{or} \quad D \frac{\partial^2 C_A}{\partial X^2} \ll V_X \frac{\partial C_A}{\partial X} \] (See BSL book)

If you neglect the axial diffusion term, the 2D elliptic PDE is modified/simplified to 1D parabolic PDE:

\[ V_X \frac{\partial T}{\partial X} = \frac{\alpha}{r \frac{\partial}{\partial r}} \left( r \frac{\partial T}{\partial r} \right) \quad \text{or} \quad V_X \frac{\partial C_A}{\partial X} = \frac{D}{r \frac{\partial}{\partial r}} \left( r \frac{\partial C_A}{\partial r} \right) \]

This simplified PDE can be solved using the Crank-Nicholson technique described in the preceding lecture. In other words, one can numerically march in ‘X’ direction and solve in ‘r’ direction using Thomas Algorithm for the tridiagonal matrix built from the discretized ‘r’ terms. For now, let us revert to the original (full) non-dimensionalized elliptic PDE:

\[
(1 - \xi^2) \frac{\partial \theta}{\partial z} = \frac{1}{P_e^2} \frac{\partial^2 \theta}{\partial \xi^2} + \frac{2}{\xi} \frac{\partial^2 \theta}{\partial \xi^2}
\]

At \( \xi = 0 \), there is a discontinuity and you are solving an approximation:

\[
(1 - \xi^2) \frac{\partial \theta}{\partial z} = \frac{1}{P_e^2} \frac{\partial^2 \theta}{\partial \xi^2} + \frac{2}{\xi} \frac{\partial^2 \theta}{\partial \xi^2}
\]
Discretize the main conservation equation:

\[
(1 - \xi_j^2) \frac{\theta_{l+1,j} - \theta_{l-1,j}}{2\Delta z} = \frac{1}{p_e^2} \frac{\theta_{l+1,j} - 2\theta_{l,j} + \theta_{l-1,j}}{\Delta z^2} + \frac{\theta_{l,j+1} - 2\theta_{l,j} + \theta_{l,j-1}}{\Delta \xi^2} + \frac{1}{\xi_j} \frac{\theta_{l+1,j} - \theta_{l-1,j}}{2\Delta \xi}
\]

or

\[
\left( \frac{(1-\xi_j^2)}{2\Delta z} + \frac{1}{p_e^2 \Delta z^2} \right) \theta_{l-1,j} - \frac{2\theta_{l,j}}{p_e^2 \Delta z^2} - \left( \frac{(1-\xi_j^2)}{2\Delta z} - \frac{1}{p_e^2 \Delta z^2} \right) \theta_{l+1,j} = \left( \frac{1}{2 \Delta \xi_j} - \frac{1}{\Delta \xi^2} \right) \theta_{l,j-1} + \frac{2\theta_{l,j}}{\Delta \xi^2} - \left( \frac{1}{2 \Delta \xi_j} + \frac{1}{\Delta \xi^2} \right) \theta_{l,j+1}
\]

\[
\text{Discretize the approximated conservation equation at the line of symmetry:}
\]

\[
\left( \frac{(1-\xi_j^2)}{2\Delta z} + \frac{1}{p_e^2 \Delta z^2} \right) \theta_{l-1,0} - \frac{2\theta_{l,0}}{p_e^2 \Delta z^2} - \left( \frac{(1-\xi_j^2)}{2\Delta z} - \frac{1}{p_e^2 \Delta z^2} \right) \theta_{l+1,0} = -\frac{2\theta_{l-1,0}}{\Delta \xi^2} + \frac{4\theta_{l,0}}{\Delta \xi^2} - \frac{2\theta_{l+1,0}}{\Delta \xi^2}
\]

\[
i = 1, N \quad j = 0 \text{ (note } \xi_0 = 0) \]

\[
\Delta z = 1/N, \quad \Delta \xi = \frac{L/L_c}{M}
\]

Note that RHS terms for all rows contain guess values for all \( \theta_{l,j} \) to begin with

Discretize BCs:

\[
i = 0, j = 0, M; \quad \theta_{0,j} = 1; \quad i = N, j = 0, M; \quad \theta_{N+1,j} = \theta_{N-1,j} \]
\[
j = 0, i = 1, N; \quad \theta_{i,-1} = \theta_{i,1}; \quad j = M, i = 0, N; \quad \theta_{i,M+1} = \theta_{i,M-1} - 2M' \Delta \xi
\]

\[
\text{1st row of Tridiagonal matrix,}
\]

\[
i = 1 \quad \theta_{1,0} - \frac{2\theta_{1,0}}{p_e^2 \Delta z^2} - \left( \frac{1 - \xi_0^2}{2\Delta z} - \frac{1}{p_e^2 \Delta z^2} \right) \theta_{2,0} = -\frac{4\theta_{1,0}}{\Delta \xi^2} + \frac{4\theta_{1,0}}{\Delta \xi^2} - \left( \frac{1 - \xi_0^2}{2\Delta z} + \frac{1}{p_e^2 \Delta z^2} \right) \theta_{0,0}
\]

\[
= 1
\]
middle rows,
\[ \begin{align*}
i = 2, N - 1: \quad & \left( \frac{1 - \xi_0^2}{2\Delta z} + \frac{1}{p_e^2\Delta z^2} \right) \theta_{i-1,0} - \frac{2\theta_{i,0}}{p_e^2\Delta z^2} - \left( \frac{1 - \xi_0^2}{2\Delta z} - \frac{1}{p_e^2\Delta z^2} \right) \theta_{i+1,0} \\
&= -\frac{4\theta_{i,1}}{\Delta \xi^2} + \frac{4\theta_{i,0}}{\Delta \xi^2}
\end{align*} \]

last row, \[ i = N: \quad \frac{2}{p_e^2\Delta z^2} \theta_{N-1,0} - \frac{2\theta_{N,0}}{p_e^2\Delta z^2} = -\frac{4\theta_{N,1}}{\Delta \xi^2} + \frac{4\theta_{N,0}}{\Delta \xi^2} \]

You have a tridiagonal matrix \( A \) in \( A\theta_{i,0}^{k+1} = b_{i,0}^{k} \) to invert

\[ j = 1, M - 1 \] (Visit the main conservation equation)

1st row,
\[ i = 1: \quad -\frac{2\theta_{1,j}}{p_e^2\Delta z^2} - \left( \frac{1 - \xi_j^2}{2\Delta z} - \frac{1}{p_e^2\Delta z^2} \right) \theta_{2,j} = -\left( \frac{1 - \xi_j^2}{2\Delta z} + \frac{1}{p_e^2\Delta z^2} \right) \theta_{0,j} \]

\[ \begin{align*}
&\left( \frac{1}{2\Delta \xi_j} - \frac{1}{\Delta \xi^2} \right) \theta_{1,j-1} + \frac{2\theta_{1,j}}{\Delta \xi^2} - \left( \frac{1}{2\Delta \xi_j} + \frac{1}{\Delta \xi^2} \right) \theta_{1,j+1}
\end{align*} \]

middle rows:
\[ i = 2, N - 1 \]: These are middle grids unaffected by BCs.

last row
\[ i = N \]
\[ \frac{2}{p_e^2\Delta z^2} \theta_{N-1,j} - \frac{2\theta_{N,j}}{p_e^2\Delta z^2} = \left( \frac{1}{2\Delta \xi_j} - \frac{1}{\Delta \xi^2} \right) \theta_{N,j-1} + \frac{2\theta_{N,j}}{\Delta \xi^2} - \left( \frac{1}{2\Delta \xi_j} + \frac{1}{\Delta \xi^2} \right) \theta_{N,j+1} \]

Again, \( A\theta_{i,j}^{k+1} = b_{i,j}^{k} \); \( i = 1, N \)

\[ j = M \]

1st row,
\[ i = 1: \quad -\frac{2\theta_{1,M}}{p_e^2\Delta z^2} - \left( \frac{1 - \xi_M^2}{2\Delta z} - \frac{1}{p_e^2\Delta z^2} \right) \theta_{2,M} = -\left( \frac{1 - \xi_M^2}{2\Delta z} + \frac{1}{p_e^2\Delta z^2} \right) \theta_{0,M} \]

\[ -\frac{2}{\Delta \xi^2} \theta_{1,M-1} + \frac{2\theta_{1,M}}{\Delta \xi^2} + 2M'\Delta \xi \left( \frac{1}{2\Delta \xi_M} + \frac{1}{\Delta \xi^2} \right) \]

middle rows, LHS (same as that of eq1), unaffected by BC =
\[ i = 2, N - 1 \]
\[ -\frac{2}{\Delta \xi^2} \theta_{i,M-1} + \frac{2\theta_{i,M}}{\Delta \xi^2} + 2M'\Delta \xi \left( \frac{1}{2\Delta \xi_M} + \frac{1}{\Delta \xi^2} \right) \]
last row,

\[ i = N \quad \frac{2\theta_{N-1,M}}{P_e^2 \Delta z^2} - \frac{2\theta_{N,M}}{P_e^2 \Delta z^2} = \frac{2}{\Delta \xi^2} \theta_{N,M-1} + \frac{2\theta_{N,M}}{\Delta \xi^2} + 2M'\Delta \xi \left( \frac{1}{2\Delta \xi \xi_M} + \frac{1}{\Delta \xi^2} \right) \]

Again, \[ A\theta_{i,M}^{k+1} = b_{i,M}^k \]

You have called the Thomas Algorithm (M+1) times to invert the tridiagonal \( A \) matrix to solve \( \theta_{i,j}, \quad i = 1, N \), \( j = 0, M \) using the guess values on the RHS of the tridiagonal matrix.

You must have noted that we did not prepare the tridiagonal matrix, and instead directly substituted the BCs in the discretized equations! As an exercise, prepare the tridiagonal matrix by defining individuals elements of rows for every ‘j’ viz \( a(i,j), b(i,j) \) ……, substitute BCs and see if you get the same equations for different ‘j’s.’

Now, this is the time to compare the solved \( \theta_{i,j} \) values with the guess values you made at the beginning of the iterations before starting the iterations.

⇒ From the programming point of view, you should be able to choose reasonable values of all variables including, \( \text{D, } V_{\text{max}}, \text{M and } P_e, \text{mass transport}, \) or \( \text{H and } P_e, \text{heat transport} \).

⇒ As earlier noted, solving elliptic PDE is computational extensive, requiring guess values, iterations and convergence. Very often, one artificially inserts a transient term \( \left( \frac{\partial C_A}{\partial t} \text{ or } \rho C_p \frac{\partial T}{\partial t} \text{ or } \rho \frac{\partial V}{\partial t} \right) \) and seeks SS solutions, which is the focus of the last two lectures.
Lecture #26-27

Time – dependent 2D parabolic PDE: ADI Method

We are solving, for examples:

\[
\frac{\partial C_A}{\partial t} + V \cdot \nabla C_A = D \nabla^2 C_A + (-r_A) ; \quad C_A(t, r, x)
\]

or

\[
\rho C_p \left( \frac{\partial T}{\partial t} + V \cdot \nabla T \right) = k \nabla^2 T + (-r_A)(\Delta H) ; \quad T(t, r, x)
\]

**Note:** SS solution must be the same as that of the converged solution of the analogous 2D elliptic PDE \( \frac{\partial C_A}{\partial t} = \frac{\partial T}{\partial t} = 0 \) discussed in the previous lectures.

Let us take a general case of the time-dependent 2D PDE:

\[
\frac{\partial \phi}{\partial t} = \phi_{XX} + \phi_X + \phi_{YY} + \phi_Y \quad ; \phi(t, x, y)
\]

with necessary IC and BCs.

Apply Crank-Nicholson method/scheme to discretize \( \phi_{XX}, \phi_{YY}, \phi_X, and \phi_Y \) terms, in the similar fashion solved the time-dependent 1D PDE on \( \phi(t, x) \):

\[
\left( \frac{\phi^{t+1} - \phi^t}{\Delta t} \right)_{i, j} = \frac{1}{2} \left[ \left( \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta X^2} \right)^{t+1} + \left( \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta X} \right)^{t+1} \right] + \left( \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta Y^2} \right)^{t+1} + \left( \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta Y} \right)^{t+1}
\]

- This way the X-derivatives have been discretized implicitly on time, whereas the Y-derivatives have been discretized explicitly.
- Alternatively, one can discretize Y-derivatives implicitly on time, whereas X-derivatives can be discretized explicitly.
- Re-arranging the terms as \( A \phi_x^{t+1} = \phi_y^t \) or \( A \phi_y^{t+1} = \phi_x^t \) from either scheme, it is clear that ‘A’ will be a tridiagonal matrix, and one can proceed on time-step by solving \( \phi \) on X-Y plane:
\[
(\frac{1}{2\Delta X^2} - \frac{1}{4\Delta X}) \phi_{i-1,j}^{t+1} - \left( \frac{1}{\Delta t} + \frac{1}{2\Delta X^2} \right) \phi_{i,j}^{t+1} + \left( \frac{1}{2\Delta X^2} + \frac{1}{4\Delta X} \right) \phi_{i+1,j}^{t+1} = \left( \frac{1}{2\Delta Y} - \frac{1}{2\Delta Y^2} \right) \phi_{i,j-1}^t - \left( \frac{1}{\Delta t} - \frac{1}{2\Delta Y^2} \right) \phi_{i,j}^t - \left( \frac{1}{2\Delta Y} - \frac{1}{2\Delta Y^2} \right) \phi_{i,j+1}^t
\]

- This way you are ‘marching’ in ‘j’ direction and ‘sweeping’ in ‘i’ direction.

By the second scheme, if you ‘march’ in ‘i’ direction and ‘sweep’ in ‘j’ direction, you will get the following equation:

\[
\left( \frac{1}{4\Delta Y} - \frac{1}{2\Delta Y^2} \right) \phi_{i,j-1}^{t+1} + \left( \frac{1}{\Delta t} + \frac{1}{2\Delta Y^2} \right) \phi_{i,j}^{t+1} - \left( \frac{1}{4\Delta Y} + \frac{1}{2\Delta Y^2} \right) \phi_{i,j+1}^{t+1} = \left( \frac{1}{2\Delta X^2} - \frac{1}{4\Delta X} \right) \phi_{i-1,j}^t + \left( \frac{1}{\Delta t} - \frac{1}{2\Delta X^2} \right) \phi_{i,j}^t + \left( \frac{1}{2\Delta X^2} + \frac{1}{4\Delta X} \right) \phi_{i+1,j}^t
\]

- Both schemes will work and the Crank-Nicholson method will produce 2nd order accuracy.

- Alternate Direct Implicit (ADI) method is an improved method producing 4th order accuracy without any extra computational cost (for more, refer the book by Ferziger):

First discretize \( \phi_X \) and \( \phi_{XX} \) terms implicitly and \( \phi_Y \) and \( \phi_{YY} \) terms explicitly to solve \( \phi \) over half step \((\Delta t/2)\) and then discretize \( \phi_Y \) and \( \phi_{YY} \) terms implicitly and \( \phi_X \) and \( \phi_{XX} \) terms explicitly over the next half step:

Step 1:

\[
\left( \frac{\phi_{x}^{t+1/2} - \phi_x^t}{\Delta t/2} \right)_{i,j} = \frac{1}{2} \left[ (\phi_{XX} + \phi_X)^{t+1/2} + (\phi_{YY} + \phi_Y)^t \right]
\]

\[\text{implicit} \quad \text{explicit}\]

Step 2:

\[
\left( \frac{\phi_{y}^{t+1/2} - \phi_y^{t+1/2}}{\Delta t/2} \right)_{i,j} = \frac{1}{2} \left[ (\phi_{XX} + \phi_X)^{t+1/2} + (\phi_{YY} + \phi_Y)^{t+1} \right]
\]

\[\text{explicit} \quad \text{implicit}\]

or

\[
A\phi_{x}^{t+1/2} = \phi_y^t \\
\text{and} \quad B\phi_{y}^{t+1} = \phi_x^{t+1/2}
\]

The procedure can also be understood by the following illustration:
**Ex:** Consider 2D diffusion in a rectangular shaped solid porous carbon block. The block is initially soaked with moisture, say at concentration $C_o \text{ (moles/m}^3\text{)}$. At $t = 0^+$ all four sides of the block are exposed to dry air (moisture concentration $= C_i \ll C_o$). We are interested in calculating the unsteady-state concentration profiles of moisture within the block, i.e. $C(t,x,y) = \?$. Pore diffusion coefficient for moisture in solid is $D_{pore}$.

**Ans:**

A species balance over $\Delta x \Delta y \cdot CV$ will yield the following conservation equation:

$$\frac{\partial C_A}{\partial t} + \sqrt[4]{\nabla C_A} = D_{pore} \nabla^2 C_A + (\frac{\partial C_A}{\partial y} \big|_{y=0})$$

$$\frac{\partial C_A}{\partial t} = D_{pore} \left( \frac{\partial^2 C_A}{\partial x^2} + \frac{\partial^2 C_A}{\partial y^2} \right)$$

$t = 0 \quad C = C_o \quad \text{for } L \geq x \geq 0; w \geq y \geq 0$

$0^+ \quad C = C_i^* \quad \text{at } x = 0 \& L \text{ for } w > y > 0$

and $\text{at } y = 0 \& w \text{ for } L > x > 0$

($C_i^*$ is the solid phase moisture concentration at the surface of the block in equilibrium with $C_i$ in atmosphere).
Step 1: discretize 'i' implicitly and 'j' explicitly over \((t & t + \frac{1}{2})\)

\[
\frac{C_{i,j}^{t+\frac{1}{2}} - C_{i,j}^t}{\Delta t/2} = \frac{D_{\text{pore}}}{2} \left( \left( \frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta X^2} \right)^{t+\frac{1}{2}} + \left( \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta y^2} \right) \right)
\]

\((\Delta X = L/N, \Delta y = w/M)\)

Arrange:

\[
\frac{D_{\text{pore}}}{2\Delta X^2} C_{i-1,j}^{t+\frac{1}{2}} - \left( \frac{D_{\text{pore}}}{\Delta X^2} + \frac{2}{\Delta t} \right) C_{i,j}^{t+\frac{1}{2}} + \frac{D_{\text{pore}}}{2\Delta X^2} C_{i+1,j}^{t+\frac{1}{2}} = -\frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j-1}^t + \left( \frac{D_{\text{pore}}}{\Delta y^2} - \frac{2}{\Delta t} \right) C_{i,j}^t - \frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j+1}^t
\]

\[i = 1, N - 1\]

\[j = 1, M - 1\]

Step 2:

Discretize BCs: \(C_{o,j} = C_{N,j} = C^*; \ j = 0, M\)

\(C_{i,0} = C_{i,M} = C^*; \ i = 0, N\)

Prepare the tridiagonal matrix:

\[a(i,j) = \frac{D_{\text{pore}}}{2\Delta X^2}; \ b(i,j) = -\left( \frac{D_{\text{pore}}}{\Delta X^2} + \frac{2}{\Delta t} \right); \ c(i,j) = \frac{D_{\text{pore}}}{2\Delta X^2}; \ d(i,j) = -\frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j-1} + \]

\[\left( \frac{D_{\text{pore}}}{\Delta y^2} - \frac{2}{\Delta t} \right) C_{i,j}^t - \frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j+1}^t\]

\[i = 1, N - 1\]

\[j = 1, M - 1\]
Substitute BCs and the following coefficients will be modified:

\[
\begin{align*}
&d(1,j) = d(1,j) - a(1,j)C^* \\
&d(N-1,j) = d(N-1,j) - c(N-1,j)C^*
\end{align*}
\]

\[j = 1, M - 1\]

Convince yourself that no other coefficients will change. This was actually an easy problem, when all four boundary conditions were simple, i.e., functional values were prescribed. Problems are complicated when you have ‘flux’/gradient or mixed boundary conditions; for example,

\[-D \frac{\partial c}{\partial x} = k_m(C - C_{atm})\]

or

\[= \text{Flux (known)}\]

and/or \[-D \frac{\partial c}{\partial y} = kC \text{ or } k_m(C - C_{atm})\]

In such cases the other coefficients may also change. Revert to the previous step.

Tridiagonal \((N − 1, a, b, c, d, y(i,j))\) \(i = 1, N − 1\)

\[\Rightarrow \text{You will be calling the subroutine } 'M − 1' \text{ times as you 'march' along 'j' direction. Now, you have the values for } C_{i,j}^{t+1/2}, \quad i = 1, N − 1, \quad j = 1, M − 1\]

Step3:

Now march along ‘i’ direction and ‘sweep’ along ‘j’ direction to solve for \(C_{i,j}^{t+1}\) from \(C_{i,j}^{t+1/2}\).

\[
\frac{C_{i,j}^{t+1} - C_{i,j}^{t+1/2}}{\Delta t/2} = \frac{D_{\text{pore}}}{2} \left[ \left( \frac{C_{i,j-1}^{t+1} - 2C_{i,j}^{t+1} + C_{i,j+1}^{t+1}}{\Delta x^2} \right)^{t+1/2} + \left( \frac{C_{i,j-1}^{t+1} - 2C_{i,j}^{t+1} + C_{i,j+1}^{t+1}}{\Delta y^2} \right)^{t+1/2} \right]
\]

explicit

\[
\text{implicit}
\]

Arrange:

\[
\frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j-1}^{t+1} - \left( \frac{D_{\text{pore}}}{\Delta y^2} + \frac{2}{\Delta t} \right) C_{i,j}^{t+1} + \frac{D_{\text{pore}}}{2\Delta y^2} C_{i,j+1}^{t+1/2} = -\frac{D_{\text{pore}}}{2\Delta x^2} C_{i-1,j}^{t+1/2} + \frac{D_{\text{pore}}}{\Delta x^2} - \frac{2}{\Delta t} \right) C_{i,j}^{t+1/2} - \frac{D_{\text{pore}}}{2\Delta x^2} C_{i+1,j}^{t+1/2}; \quad j = 1, M - 1
\]

\[i = 1, N - 1\]
Step 4:
Discretize the BCs: Same as before.

Prepare the tridiagonal matrix:

\[ a(i, j) = \frac{D_{pore}}{2\Delta y^2} \quad b(i, j) = -\left(\frac{D_{pore}}{\Delta y^2} + \frac{2}{\Delta t}\right) \quad c(i, j) = \frac{D_{pore}}{2\Delta x^2} \quad d(i, j) = -\frac{D_{pore}}{2\Delta x^2} \quad C_{t+1/2}^{i-1,j} + \left(\frac{D_{pore}}{\Delta X^2} - \frac{2}{\Delta t}\right) C_{t,j}^{t+1/2} - \frac{D_{pore}}{2\Delta X^2} C_{t+1,j}^{t+1/2} \]

\[ \begin{array}{l}
i = 1, N - 1 \\
j = 1, M - 1
\end{array} \]

Substitute the discretized BCs, and only the following coefficients will be modified:

\[ d(i, 1) = d(i, 1) - a(i, 1) C^* \quad d(i, M - 1) = d(i, M - 1) - c(i, M - 1) C^* \]

\[ \{i = 1, N - 1\} \]

Tridiagonal\( (M - 1, a, b, c, d, y(i, j)) \)\( j = 1, M - 1 \)

⇒ You will be calling the subroutine 'N - 1' times as you 'march' along 't' direction. Thus, you have solved for \( C_{i,j}^{t+1} \), \( i = 1, N - 1 \)

Recap (solving 2nd order ODE and PDEs at a glance)

BVP: \( A\phi_i = b \) (Direct Method)

1D parabolic: \( A\phi_{i}^{t+1} = \phi_{i}^{t} \) (Crank-Nicholson Method)

2D Elliptic: \( A\phi_{i,j} = \phi_{j,i}^{guess} \) or \( A\phi_{j,i} = \phi_{i,j}^{guess} \) (Method of Lines)

2D parabolic: (ADI Method)

\[ \begin{array}{l}
A\phi_{i}^{t+1/2} = \phi_{j}^{t} \quad \text{or} \quad A\phi_{j}^{t+1/2} = \phi_{i}^{t} \\
\text{and} \quad B\phi_{j}^{t+1} = \phi_{i}^{t+1/2} \quad \text{or} \quad B\phi_{i}^{t+1} = \phi_{j}^{t+1/2}
\end{array} \]

where, A and B are the Tridiagonal matrices.
Example: Consider the SS flow of a liquid through a long tube. Reynolds number is 180. At time \( t = 0 \), a tracer is injected into the liquid at inlet to the tube. Diffusion coefficient of tracer in the liquid is \( D \text{ cm}^2/\text{s} \). Determine the time profiles of the tracer concentrations \((r,x)\) in the tube, ie. \( C(t,r,x) = ? \)

Soln.

\[
V(r) = U_{\text{max}} \left(1 - \frac{r^2}{R^2}\right)
\]

A species balance over \( 2\pi r \Delta r \Delta x' \) \( CV \) yields the following eqn

\[
\frac{\partial C}{\partial t} + V(r) \frac{\partial C}{\partial x} = D \frac{\partial^2 C}{\partial x^2} + \left( -r \frac{\partial C}{\partial r} \right)
\]

or

\[
\frac{\partial C}{\partial t} + V(r) \frac{\partial C}{\partial x} = D \left( \frac{\partial^2 C}{\partial x^2} + \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial C}{\partial r} \right) \right) \quad L > X > 0 \quad R > r > 0
\]

or

\[
\frac{\partial C}{\partial t} = \left( -V(r) \frac{\partial C}{\partial x} + D \frac{\partial^2 C}{\partial x^2} \right) + D \left( \frac{1}{r} \frac{\partial C}{\partial r} + \frac{\partial^2 C}{\partial r^2} \right)
\]

\[
t = 0 \quad C = 0 \quad L \geq x \geq 0 \quad \text{and} \quad R \geq r \geq 0
\]

\[
0^+ \quad C = C_o \quad @ \quad x = 0 \quad \frac{\partial C}{\partial x} = 0 \quad @ \quad X = L \quad \text{for all} \quad R > r > 0
\]

\[
(\text{long tube approximation})
\]

\[
\frac{\partial C}{\partial r} = 0 \quad @r = 0 \quad (\text{symmetric})
\]

\[
= 0 \quad @r = R \quad (\text{non-reactive walls}) \quad \text{for all} \quad L > X > 0
\]
Note: You will encounter discontinuity at $r = 0$ in the radial diffusion terms while discretizing. Considering $\nabla C = 0$, you will be solving the approximated equation instead.

\[
\frac{\partial C}{\partial t} = (-V(r) \frac{\partial C}{\partial x} + D \frac{\partial^2 C}{\partial x^2}) + 2D \frac{\partial^2 C}{\partial r^2} \ 	ext{at} \ r = 0 \ 	ext{for all} \ L > X > 0
\]

Discretize the approximated equation:

\[
\frac{C_{i,j}^{t+1/2} - C_{i,j}^t}{\Delta t/2} = \frac{1}{2} \left[ \left( -V_j \frac{C_{i+1,j} - C_{i-1,j}}{2\Delta X} + D \frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{4\Delta X^2} \right)^{t+1/2} \right.
\]
\[
+ 2D \left( \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta r^2} \right)^t \left. \right\} \begin{array}{c} \text{implicit} \\
\text{explicit} \end{array} ; \ j = 0 \begin{array}{c} \text{for} \ i = 1, N \end{array}
\]

Arrange:

\[
\left( \frac{V_j}{4\Delta X} + \frac{D}{2\Delta X^2} \right) C_{i-1,j}^{t+1/2} - \left( \frac{2}{\Delta t} + \frac{D}{\Delta X^2} \right) C_{i,j}^{t+1/2} - \left( \frac{V_j}{4\Delta X} - \frac{D}{2\Delta X^2} \right) C_{i+1,j}^{t+1/2}
\]
\[
= - \left( \frac{D}{\Delta r^2} \right) C_{i,j-1} - \left( \frac{2}{\Delta t} - \frac{2D}{\Delta r^2} \right) C_{i,j}^{t} - \left( \frac{D}{\Delta r^2} \right) C_{i,j+1}^{t+1/2} ; \ j = 0 \begin{array}{c} \text{for} \ i = 1, N \end{array}
\]
Similarly, discretize the main equation:
\[
\frac{C_{i,j}^{t+\frac{1}{2}} - C_{i,j}^t}{\Delta t/2} = \frac{1}{2} \left[ \left( \frac{-V_j}{2\Delta X} \frac{C_{i+1,j} - C_{i-1,j}}{2\Delta X} + D \frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta X^2} \right)_{\text{implicit}} + D \left( \frac{1}{r_j} \frac{C_{i,j+1} - C_{i,j-1}}{2\Delta r} + \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta r^2} \right)_{\text{explicit}} \right]_{j = 1, M} \]

Arrange,
\[
\left( \frac{V_j}{4\Delta X} + \frac{D}{2\Delta X^2} \right) C_{i-1,j}^{t+\frac{1}{2}} - \left( \frac{2}{\Delta t} + \frac{D}{\Delta X^2} \right) C_{i,j}^{t+\frac{1}{2}} - \left( \frac{V_j}{4\Delta X} - \frac{D}{2\Delta X^2} \right) C_{i+1,j}^{t+\frac{1}{2}} = \left( \frac{D}{4r_j\Delta r} - \frac{D}{2\Delta r^2} \right) C_{i,j-1}^t - \left( \frac{2}{\Delta t} - \frac{D}{\Delta r^2} \right) C_{i,j}^t - \left( \frac{D}{4r_j\Delta r} + \frac{D}{2\Delta r^2} \right) C_{i,j+1}^t; \quad j = 1, M \]

\{ \text{Note: } V_j = U_{\text{max}} \left( 1 - \frac{r_j^2}{R^2} \right); \quad \eta_j = j\Delta r \}

Discretize BCs & IC
\[
t = 0, \quad C(i, j) = 0 \quad \text{or a small number } C_{\text{int}} \ll C_o \quad (N \geq i \geq 0 \text{ and } M \geq j \geq 0)
\]
\[
\begin{align*}
0^+ C(0, j) &= C_o \\
\frac{C(N+1, j) - C(N-1, j)}{\Delta X} &= 0 \\
\text{or } C(N + 1, j) &= C(N - 1, j) \\
C(i, -1) &= C(i, +1) \\
C(i, M + 1) &= C(i, M - 1)
\end{align*}
\]

\{ \text{Note: } \Delta X, \Delta r \text{ are small enough } \}

Prepare tridiagonal matrix
\[
\begin{align*}
\text{for } j = 0: \quad &a(i, 0) = \left( \frac{V_0}{4\Delta X} + \frac{D}{2\Delta X^2} \right); \quad b(i, 0) = -\left( \frac{2}{\Delta t} + \frac{D}{\Delta X^2} \right); \quad c(i, 0) = -\left( \frac{V_0}{4\Delta X} - \frac{D}{2\Delta X^2} \right); \quad d(i, 0) = \\
&= -\left( \frac{D}{\Delta r^2} \right) C_{i,-1}^t - \left( \frac{2}{\Delta t} - \frac{2D}{\Delta r^2} \right) C_{i,0}^t - \left( \frac{D}{\Delta r^2} \right) C_{i,+1}^t; \quad i = 1, N
\end{align*}
\]

Substitute BCs,
\[
d(1,0) = -\left( \frac{2D}{\Delta r^2} \right) C_{i,1}^t - \left( \frac{2}{\Delta t} - \frac{2D}{\Delta r^2} \right) C_{i,0}^t - a(1,0)C_o \\
a(N,0) = a(N,0) + c(N,0)
\]

\[\text{End of Page 3}\]
\[ d(N, 0) = -\left(\frac{2D}{\Delta r^2}\right) C^t_{N,1} - \left(\frac{2}{\Delta t} - \frac{2D}{\Delta r^2}\right) C^t_{N,0} \]

Tridiagonal \((N, a, b, c, d, y^{t+1/2}(i, 0))\); \(i = 1, N\)

Preparation of tridiagonal matrix... continue...

\[
\begin{align*}
  a(i, j) &= \left(\frac{V_j}{4\Delta X} + \frac{D}{2\Delta X^2}\right) \quad b(i, j) = -\left(\frac{2}{\Delta t} + \frac{D}{\Delta X^2}\right) \quad c(i, j) = -\left(\frac{V_j}{4\Delta X} - \frac{D}{2\Delta X^2}\right) \quad d(i, j) \\
  &= \left(\frac{D}{4r_j\Delta r} - \frac{D}{2\Delta r^2}\right) C^t_{i,j-1} - \left(\frac{2}{\Delta t} - \frac{D}{\Delta r^2}\right) C^t_{i,j} \\
  &= \left(\frac{D}{4r_j\Delta r} + \frac{D}{2\Delta r^2}\right) C^t_{i,j+1} \quad j = 1, M \\
  &\quad \quad i = 1, N
\end{align*}
\]

Substitute BC \((j = 1, M - 1)\)

\[
\begin{align*}
  d(1, j) &= d(1, j) - a(1, j)C_o \\
  a(N, j) &= a(N, j) + c(N, j)
\end{align*}
\]

Tridiagonal \((N, a, b, c, d, y^{t+1/2}(i, j))\); \(i = 1, N\)

You have called the tridiagonal subroutine \(M - 1\) times for the interior nodes.

Substitute BC @ \(j = M\)

\[
\begin{align*}
  d(1, M) &= -\left(\frac{D}{\Delta r^2}\right) C^t_{1,M-1} - \left(\frac{2}{\Delta t} - \frac{D}{\Delta r^2}\right) C^t_{1,M} - a(1, N). C_o \\
  d(i, M) &= -\left(\frac{2}{\Delta r^2}\right) C^t_{i,M-1} - \left(\frac{2}{\Delta t} - \frac{D}{\Delta r^2}\right) C^t_{i,M} \\
  a(N, M) &= a(N, M) + c(N, M)
\end{align*}
\]

\[
\begin{align*}
  d(N, M) &= -\left(\frac{D}{\Delta r^2}\right) C^t_{N,M-1} - \left(\frac{2}{\Delta t} - \frac{D}{\Delta r^2}\right) C^t_{N,M} \quad \text{Tridiagonal} \quad (N, a, b, c, d, y^{t+1/2}(i, M)) \quad i = 1, N
\end{align*}
\]

(Therefore, you have called the Thomas Algorithm \((M + 1)\) times while sweeping \(j = 0, M\) rows)
For the last time in this course, let us write down complete eq^n

\[ Ay^{t+1/2} = dt \]

for \( j = 0, i = 1, N \)

\[
\begin{bmatrix}
-\left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & -\left( \frac{V_0}{4\Delta X} + \frac{D}{2\Delta X^2} \right) & \cdots & \cdots & \cdots \\
\vdots & \left( \frac{V_0}{4\Delta X} + \frac{D}{2\Delta X^2} \right) & -\left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & -\left( \frac{V_0}{4\Delta X} - \frac{D}{2\Delta X^2} \right) & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \left( \frac{D}{\Delta x^2} \right) \\
-\left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & -\left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & \cdots & \cdots & \cdots \\
\end{bmatrix}
\]

\[ \{ Y_i \}^{t+1/2} \]

\[
= \begin{cases}
d(1,0) \\
\vdots \\
d(i,0) \\
\vdots \\
\vdots \\
d(N,0)
\end{cases}
\]

Similarly, you can write a set of equations for \( j = 1, M - 1 \) and \( j = M \) using the coefficient as above:

\( j = 1, M - 1 \)

\[
\begin{bmatrix}
-\left( \frac{2}{\Delta t} + \frac{4}{\Delta x^2} \right) & -\left( \frac{V_j}{4\Delta X} + \frac{D}{2\Delta X^2} \right) & \cdots & \cdots & \cdots \\
\vdots & \left( \frac{V_j}{4\Delta X} + \frac{D}{2\Delta X^2} \right) & -\left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & -\left( \frac{V_j}{4\Delta X} - \frac{D}{2\Delta X^2} \right) & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \left( \frac{D}{\Delta x^2} \right) \\
-\left( \frac{2}{\Delta t} + \frac{4}{\Delta x^2} \right) & -\left( \frac{2}{\Delta t} + \frac{4}{\Delta x^2} \right) & \cdots & \cdots & \cdots \\
\end{bmatrix}
\]

\[ \{ Y_i \}^{t+1/2} \]

\[
= \begin{cases}
d(i,j) \\
\vdots \\
d(i,j) \\
\vdots \\
\vdots \\
d(N,j)
\end{cases}
\]
\[ j = M \]

\[
\begin{bmatrix}
- \left( \frac{2}{\Delta t} + \frac{4}{\Delta x^2} \right) & - \left( \frac{V_M}{4\Delta X} - \frac{D}{2\Delta X^2} \right) & \ldots & \ldots & \ldots \\
\vdots & \left( \frac{V_M}{4\Delta X} + \frac{D}{2\Delta X^2} \right) & - \left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & - \left( \frac{V_M}{4\Delta X} - \frac{D}{2\Delta X^2} \right) & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
- \left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & - \left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & \left( \frac{D}{\Delta X^2} \right) & - \left( \frac{2}{\Delta t} + \frac{D}{\Delta x^2} \right) & \ldots \\
\end{bmatrix}
\begin{bmatrix}
Y_i \\
\vdots \\
Y_i \\
\end{bmatrix}^{t+1/2}
\]

\[
\begin{aligned}
&d(1, M) \\
&\vdots \\
&d(i, M) \\
&\vdots \\
&d(N, M)
\end{aligned}
\]

\[
\begin{aligned}
\left( \begin{array}{c}
d(1, M) \\
d(i, M) \\
\vdots \\
d(N, M)
\end{array} \right)^t \\
\end{aligned}
\]

Now, sweep in 'j' direction and march in 'i' direction for \((t + 1)\) based on the \(y(i,j)\) values you determined at \((t + 1/2)\) step (above)

\[
Ay^{t+1}(i,j) = d^{t+1/2} i = 1,N \\
j = 0,M
\]

Discretize \(j = 0\)

\[
\frac{C_{i,j}^{t+1} - C_{i,j}^{t+1/2}}{\Delta t/2}
\]

\[
= \frac{1}{2} \left[ \left( -V_j \frac{C_{i+1,j} - C_{i-1,j}}{2\Delta X} \right) + D \frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta X^2} \right]^{t+1/2}
\]

\[
+ 2D \left( \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta r^2} \right)^{t+1} (i = 1,N)
\]

\[ \uparrow \quad \text{explicit} \]

\[ \uparrow \quad \text{implicit} \]
\[ j = 1, M \]

\[
\frac{C_{i,j}^{t+1} - C_{i,j}^{t+1/2}}{\Delta t/2} = \frac{1}{2} \left[ \left( -V_j \right) \frac{C_{i+1,j} - C_{i-1,j}}{2\Delta X} + D \frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta X^2} \right]^{t+1/2} + D \left( \frac{1}{r_j} \frac{C_{i,j+1} - C_{i,j-1}}{2\Delta r} + \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta r^2} \right)^{t+1} \quad (i = 1, N) \]

In this last lecture of the course, I leave it here for you to do the remaining part (arranging the terms and applying BCs for \( i = 1, (2 \cdots N - 1), \ N \) rows to invert the matrix) as you sweep in \( 'j' \) direction, as an exercise. The topic on ADI stops here.

**Note:**

1. The SS solution \( y(t,x,r) \) as \( t \to \infty \) of the time-dependent 2D parabolic equation must be the same as that of the corresponding 2D elliptic PDE you have learnt how to solve in the preceding lecture, using 'Method of Lines', i.e. for

\[
\frac{\partial c}{\partial t} + V \cdot \nabla C = D \nabla^2 C + (-r_A) ;
\]

\( C(t,x,r) \) as \( t \to \infty \) must be the same as that of

\[
V \cdot \nabla C = D \nabla^2 C + (-r_A) \Rightarrow C(x,r)
\]

2. A question arises. When asked to solve the elliptic (2D) PDE, should not or cannot we artificially insert the transient term \( \frac{\partial c}{\partial t} \) and seek the SS solution to the corresponding time-dependent 2D parabolic equation? Very often, yes. Recall that, solving elliptic PDE requires iterations and there is always a convergence issue. How many iterations? On the other hand, the parabolic equation does not require iterations, and you march on \( 't' \) axis solving \( y(x,r) \) at every time step without iterations. Therefore, more than often the ADI method is preferred over 'Method of Lines' for solving an elliptic (2D) PDE. Insert the transient term and solve till you have SS solution.

3. Before closing this chapter, let us answer how we address non-linearity in the differential term, for example, \( V \frac{\partial v}{\partial x} \) of the NS equation? The answer is simple. By iterations! Guess velocity fields (\( V_g \)). Discretize the derivative term as before. Solve for velocity fields as before. Iterate
till there is convergence. Alternatively, Taylor's series can also be used to approximate velocity fields by linearization, in which case guesses are required for the velocity gradients.

**End – Semester Exam**

[In a regular semester, these course materials are usually covered in 28 lectures of 1 h 15 min duration each, or 42 lectures of 50 min duration each. For record and due acknowledgement, most of these materials were part of my graduate level lectures that I audited (forgetting the name of the instructor) way back in 1995 at the University of Arizona, Tucson, USA. At Kanpur, I offer this course to graduate students only. To this end, you are welcome to send me any comments, or mistakes or errors you notice in the lectures, to my email id: vermanishith@gmail.com.]