Nama : Muhamad Hanif Hafizan
NIM : 13123069

Faculty of Mechanical and Aerospace Engineering
Institut Teknologi Bandung

WF2202 Partial Differential Equation and Numerical Method

Take Home Exam

Given out: WED 4 JUN 2025, Due Date WED 18 JUN 2025 (17.00 WIB)

Please write this etical statement on the top of your answer sheet and sign:

"I hereby declare that all answers in the exam are from my independent work. I did not commit or facilitate any improper conduct during exam. If i am proven to be in violation, I am ready to accept the consequences in accordance with the applicable regulations"

(Muhamad Hanif Hafizan)

Nama	NIM
Mochamad Arkan Nugraha	13123007
Muhamad Hanif Hafizan	13123069

1. Create a flowchard and programming code to solve the system simultaneous linear algebraic equations, which is defined as the following:

$$
\begin{array}{lllllll}
a_{11}\,x_1 & + a_{12}\,x_2 & + \ldots + a_{1j}\,x_j & + \ldots + a_{1(n-1)}\,x_{(n-1)} & + a_{1n}\,x_n & = b_1 \\
a_{21}\,x_1 & + a_{22}\,x_2 & + \ldots + a_{2j}\,x_j & + \ldots + a_{2(n-1)}\,x_{(n-1)} & + a_{2n}\,x_n & = b_2 \\
 & \vdots & & \vdots & & \vdots \\
a_{i1}\,x_1 & + a_{i2}\,x_2 & + \ldots + a_{ij}\,x_j & + \ldots + a_{i(n-1)}\,x_{(n-1)} & + a_{in}\,x_n & = b_i \\
 & \vdots & & \vdots & & \vdots \\
a_{(n-1)1}\,x_1 & + a_{(n-1)2}\,x_2 & + \ldots + a_{(n-1)j}\,x_j & + \ldots + a_{(n-1)(n-1)}\,x_{(n-1)} & + a_{(n-1)n}\,x_n & = b_{(n-1)} \\
a_{n1}\,x_1 & + a_{n2}\,x_2 & + \ldots + a_{nj}\,x_j & + \ldots + a_{n(n-1)}\,x_{(n-1)} & + a_{nn}\,x_n & = b_n
\end{array}
\tag{E.1.1}
$$

where the a's are constant coefficients and the b's are constants, and x are unknown variables. Create a matlab programming code that can be used to solve the system simultaneous linear algebraic equations. Please include in your programming solution the following:

a) Create the solve the system simultaneous linear algebraic equations that will have inputs from the keyboards. Inputs are :

   a.1. the number of equations/number of unknown variables =n

   a.2. the a's constant coefficients and

   a.3. the b's constants.

b) There are two choice for the method:

   Choice A = Using Gauss Elimination

   Choice B = Using Gauss Seidel

c) Print the system simultaneous linear algebraic equations, and validate the code for the solution given below

   Number of equations = 4

   10 X1 - 2 X2 - X3 - X4 = 3

   -2X1 + 10 X2 - X3 - X4 = 15

   - X1 - X2 + 10 X3 - 2 X4 = 27

   - X1 - X2 - 2 X3 + 10X4 = -9

   The solutions are printed in the format as the following

   The answers after 7 iterations are the following (4 digits of accuracy):

   X1= 0.9999

   X2= 1.9999

   X3= 2.9999

   X4= -0.0002

   (for cases of n=7)

d) Use the validated code to solve the following system of equations using both Gauss Elimination and Gauss-Seidel Methods.

$$\text{Consider } \mathbf{Ax} = \mathbf{b} \text{ where } \mathbf{A} = \begin{bmatrix} 6 & 1 & 1 & 1 & 1 \\ 1 & 7 & 1 & 1 & 1 \\ 1 & 1 & 8 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 1 & 10 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} -10 \\ -6 \\ 0 \\ 8 \\ 18 \end{bmatrix}.$$

Theoretical Background:

Metode Eliminasi Gauss merupakan salah satu metode numerik yang digunakan untuk menyelesaikan sistem persamaan linear. Dalam metode ini, sistem persamaan linear diubah ke dalam bentuk matriks agar lebih mudah untuk dianalisis dan diselesaikan. Sistem persamaan linear tersebut direpresentasikan oleh dua buah matriks, yaitu:

- Matriks A yang memuat koefisien dari setiap variabel xi dalam sistem persamaan, dan
- Matriks b yang merupakan matriks kolom berisi nilai di sebelah kanan tanda sama dengan dari masing-masing persamaan.

Secara matematis, sistem tersebut dapat dituliskan dalam bentuk umum sebagai berikut:

$$Ax = b$$

Di mana:

- A adalah matriks berordo $n\ x\ n$ yang memuat koefisien,
- x adalah vektor kolom dari variabel-variabel tak diketahui,
- b adalah vektor kolom dari nilai-nilai hasil.

Contoh pada soal :

$$\begin{bmatrix} 10 & -2 & -1 & -1 \\ -1 & 10 & -1 & -1 \\ -1 & -1 & 10 & -2 \\ -1 & -1 & -1 & 10 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{bmatrix} 3 \\ 15 \\ 27 \\ -9 \end{bmatrix}$$

Dari bentuk tersebut, kita menggunakan operasi baris elementer (*Elementary Row Operation* / ERO) dan eliminasi Gauss maju (*Forward Gauss Elimination* atau disebut juga **Naïve Gauss Elimination**). Metode ini bertujuan untuk menyederhanakan matriks A menjadi bentuk matriks segitiga atas (*upper triangular matrix*) dengan cara mengeliminasi elemen-elemen yang tidak diperlukan di bawah diagonal utama.

$$\boldsymbol{a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \quad (1)}$$

Menjadi sebuah persamaan pivot dengan $\boldsymbol{a_{11}}$ sebagai koefisien pivot, namun perlu diperhatikan bahwa nilai $\boldsymbol{a_{11}}$ **tidak boleh nol**. Selanjutnya, kita perlu mengeliminasi elemen $\boldsymbol{a_{21}}$ agar menjadi nol. Oleh karena itu, kita harus mengubah persamaan dengan cara mengalikan baris pertama dengan $\boldsymbol{\frac{a_{21}}{a_{11}}}$ kemudian mengurangkannya dari baris ketiga.

Langkah ini dilakukan agar elemen pada posisi $\boldsymbol{a_{21}}$ hilang, dan proses eliminasi dapat dilanjutkan ke elemen-elemen di bawahnya hingga diperoleh bentuk matriks segitiga atas yang memudahkan penyelesaian sistem persamaan.

$$\boldsymbol{a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1 \ (2)}$$

Kita ketahui :

$$\boldsymbol{a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1} \ \ \mathbf{(3)}$$

Kita dapat menulis ulang dengan *substracting* persamaan (1) dari persamaan (2)

$$\boldsymbol{\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(\frac{a_{21}}{a_{11}}a_{1n-}a_{2n}\right)x_n = b_2\frac{a_{21}}{a_{11}}b_1} \ \ \mathbf{(4)}$$

$$\boldsymbol{a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2} \ \mathbf{(5)}$$

Ulangi proses ini sebanyak n kali (sesuai dengan n baris) untuk menghasilkan matriks segitiga atas dengan hasil. Sebagai contoh, matriks segitiga atas dari masalah harus sebagai berikut:

$$\begin{bmatrix} 10 & \mathbf{-2} & \mathbf{-1} & \mathbf{-1} \\ 0 & \boldsymbol{a'_{22}} & \boldsymbol{-a'_{23}} & \boldsymbol{a'_{24}} \\ 0 & 0 & \boldsymbol{a'_{33}} & \boldsymbol{a'_{34}} \\ 0 & 0 & 0 & \boldsymbol{a'_{44}} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{bmatrix} 3 \\ \boldsymbol{b'_2} \\ \boldsymbol{b'_3} \\ \boldsymbol{b'_4} \end{bmatrix}$$

Sekarang, seperti yang kita ketahui hasil x_4 dari matriks, kita dapat perlahan-lahan bekerja ke atas untuk menentukan x_3, x_2, dan x_1. Proses ini dapat direpresentasikan sebagai berikut:

$$x_i = \frac{b_i^{i-1} - \sum_{j=i+1}^{n}. a_{ij}^{i-1}x_j}{a_{ii}^{i-1}}$$
$$i = n-1, n-2, \dots 1$$

Augmented Matrix metho lebih disukai untuk menyederhanakan proses analitis dalam menyelesaikan matriks

Metode Gauss-Seidel merupakan metode iteratif yang digunakan untuk mencari solusi dari sistem persamaan linear. Dalam metode ini, nilai awal dari setiap variabel x diasumsikan terlebih dahulu sebagai nilai tebakan awal (*initial guess*). Nilai tebakan tersebut kemudian digunakan untuk menghitung nilai-nilai x yang baru secara berulang-ulang hingga diperoleh hasil yang konvergen atau mendekati nilai yang benar.

$$\boldsymbol{a_{11} + a_{22} + \cdots + a_{nn} > \ a_{n1} \ + \ a_{n-1,2} \ + \cdots + a_{1n}}$$

Perhitungan dapat disederhanakan dengan notasi ini:

$$x_i = \frac{b_i^{i-1} - \sum_{j=1}^{n}. \boldsymbol{a_{ij}}x_j}{\boldsymbol{a_{ii}}}$$

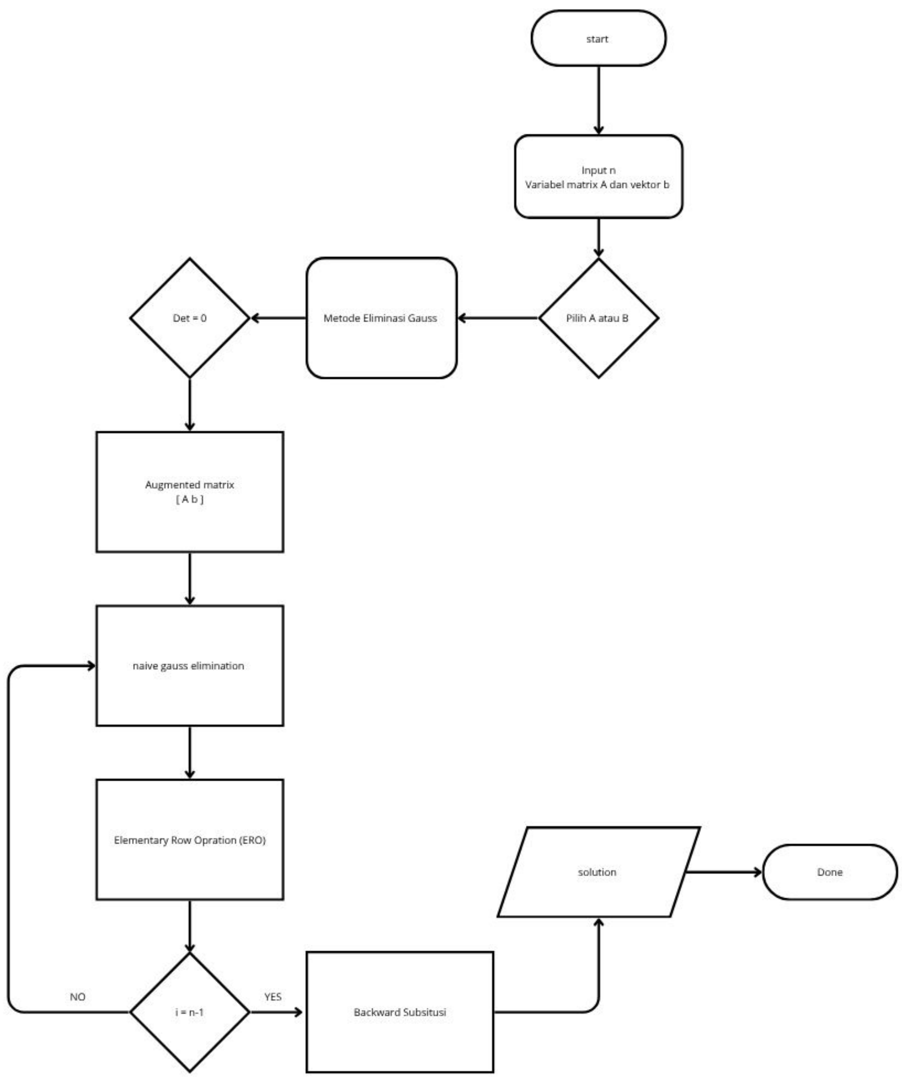
$$i = 1,2, \dots n \ and \ j = 1,2, \dots, n$$

Untuk proses iteratif terbatas, iterasi dapat dihentikan jika perubahan kesalahan lebih kecil dari kesalahan yang dapat ditoleransi yang ditentukan oleh pengguna. Yakni:

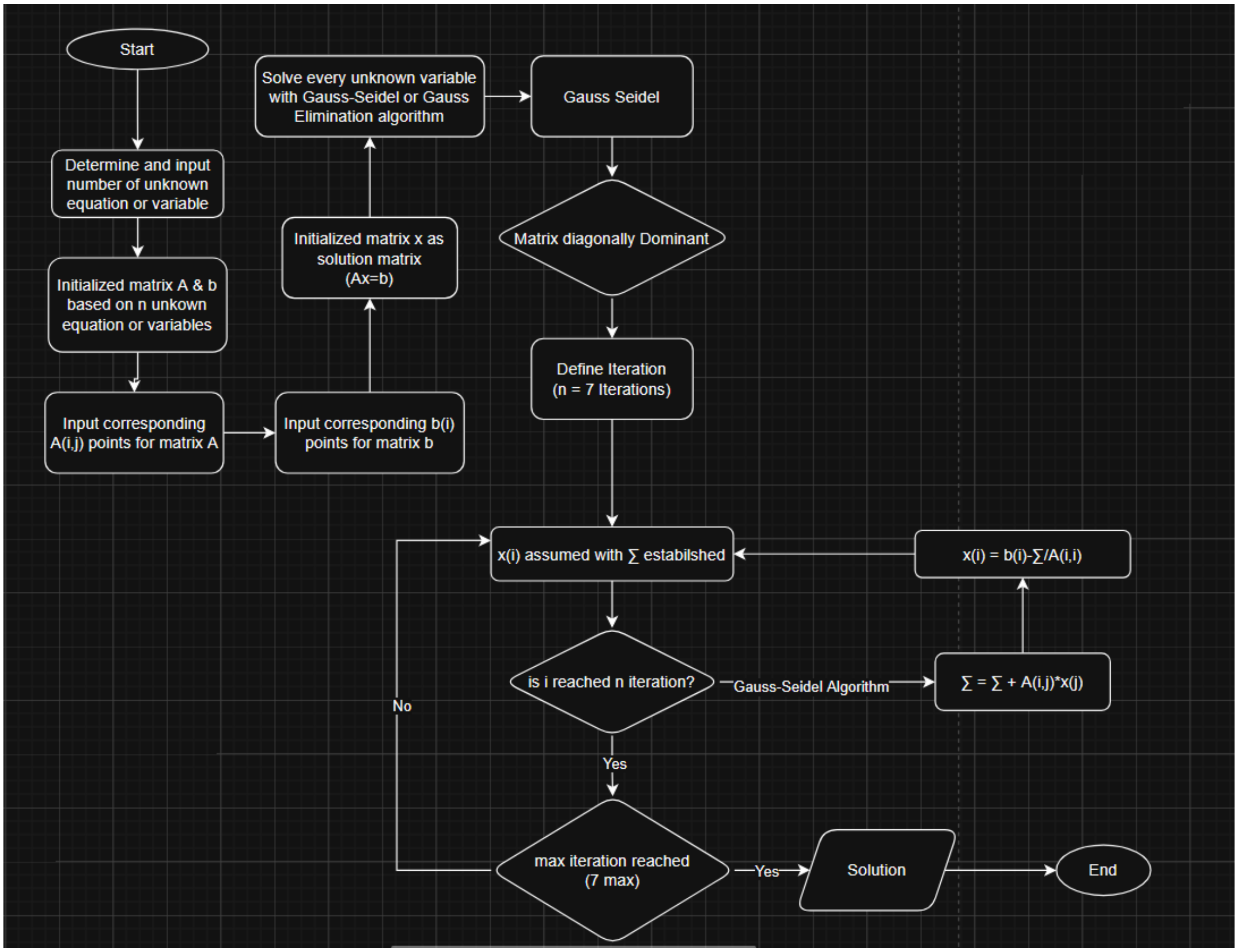
$$|\boldsymbol{\varepsilon}_{\alpha,i}| = \left| \frac{x_i^j - x_i^{j-1}}{x_i^j} \right| \times 100\%$$

Flowchart

Gauss Elimination



Gauss-Seidel



Source Code *Gauss Elimination* (1c & 1d)

`clear all; close all; clc`
`format short`
$A = \begin{bmatrix} 10 & -2 & -1 & -1; \\ -2 & 10 & -1 & -1; \\ -1 & -1 & 10 & -2; \\ -1 & -1 & -2 & 10; \end{bmatrix}$
$b = \begin{bmatrix} 3 \\ 15 \\ 27 \\ -9 \end{bmatrix}$
	Untuk membersihkan page
	Inisialisasi Matriks A dan vector b
	
	bagian 1. C (input dari user)

<table>
<tr><td>
<pre>
A = [6 1 1 1 1;
      1 7 1 1 1;
      1 1 8 1 1;
      1 1 1 9 1;
      1 1 1 1 10
     ];

b = [-10;-6;0;8;18];
</pre>
</td><td>Inisialisasi Matriks A dan vector b

bagian 1. D</td></tr>
<tr><td>
<pre>
maxerror = 1e-5;
disp("Matrix A")
disp(A);
disp("Matrix B")
disp(b);
</pre>
</td><td>Menampilkan nilai pada layar</td></tr>
<tr><td>
<pre>
n = length(b);
x = zeros(n,1); % Initial guess

% Define function
function [x,det] = gauss(A,b)
% Solves A*x = b by Gauss elimination and computes det(A).
% USAGE: [x,det] = gauss(A,b)
</pre>
</td><td>Fungsi yang digunakan untuk melakukan Eliminasi Gauss</td></tr>
<tr><td>
<pre>
for k = 1:n-1   % Elimination phase
   for i= k+1:n
       if A(i,k) ~= 0
            lambda = A(i,k)/A(k,k);
            A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
            b(i)= b(i) - lambda*b(k);
        end
    end
end
</pre>
</td><td>Mengubah matriks A menjadi segitiga atas (upper triangular).</td></tr>
<tr><td>
<pre>
if nargout == 2; det = prod(diag(A)); end

 for k = n:-1:1 % Back substitution phase
     b(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
 end
 x = b;
 end
</pre>
</td><td>Menghitung determinan matriks A (yang sudah jadi segitiga atas) dengan mengalikan semua elemen diagonal.
substitusi balik untuk mencari nilai x_n</td></tr>
</table>

```
x_Results = gauss(A,b);

% Display result rounded to 4 decimal places
fprintf('\nGauss Elimination Iterative Solution after 7 iterations:\n')
for i = 1:n
    val = floor(x_Results(i) * 10000) / 10000;
    fprintf('X%d = %.4f\n', i, val);
end
```

Menampilkan solusi $\boldsymbol{x_n}$ dibulatkan ke 4 angka di belakang koma

```
Gauss Elimination Iterative Solution after 7 iterations:
X1 = 1.0000
X2 = 2.0000
X3 = 3.0000
X4 = 0.0000
```

Hasil 1.C

Hasil Matrix X berbeda karena pembulatan MATLAB

```
Gauss Elimination Iterative Solution after 7 iterations:
X1 = -2.0000
X2 = -1.0001
X3 = 0.0000
X4 = 1.0000
X5 = 1.9999
```

Hasil 1.D

Source Code *Gauss-Seidel* (1c & 1d)

```
clear all; close all; clc
format short
```

Untuk membersihkan page

$$A = \begin{bmatrix} 10 & -2 & -1 & -1; \\ -2 & 10 & -1 & -1; \\ -1 & -1 & 10 & -2; \\ -1 & -1 & -2 & 10; \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 15 \\ 27 \\ -9 \end{bmatrix}$$

Inisialisasi Matriks A dan vector b

bagian 1.C (input dari user)

```
A = [ 6 1 1 1 1;
      1 7 1 1 1;
      1 1 8 1 1;
      1 1 1 9 1;
      1 1 1 1 10
    ];

b = [-10;-6;0;8;18];

n=5;
```

Inisialisasi Matriks A, vector b, dan n = 5

bagian 1. D

<pre>maxerror = 1e-5;
disp("Matrix A")
disp(A);
disp("Matrix B")
disp(b);</pre>
	Menampilkan nilai pada layar

<pre>n = length(b);
x = zeros(n,1); % Initial guess (initial 0,0,0,0,0)
x_prev_iteration = x;</pre>
	x adalah vektor solusi (asumsi dimulai dari nol semua)

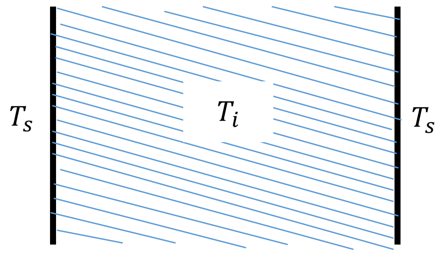
<pre>% Gauss-Seidel Formula
for k = 1:7
    for i = 1:n
        sum1 = A(i,1:i-1) * x(1:i-1);
        sum2 = A(i,i+1:n) * x_prev_iteration(i+1:n);
        x(i) = (b(i) - sum1 - sum2)/A(i,i);
    end
x_prev_iteration = x;
end</pre>
	Proses Iterasi Gauss-Seidel

<pre>x_Results = x;
% Display result rounded to 4 decimal places
fprintf('\nGauss-Seidel Iterative Solution after 7 iterations:\n');
for i = 1:n
    val = floor(x_Results(i) * 10000) / 10000;
    fprintf('X%d = %.4f\n', i, val);
end</pre>
	Menampilkan hasil akhir setelah 7 iterasi. Dan membulatkan ke bawah hingga 4 angka desimal.

<pre>Gauss-Seidel Iterative Solution after 7 iterations:
X1 = 0.9999
X2 = 1.9999
X3 = 2.9999
X4 = -0.0001</pre>
	Hasil 1.C
	Hasil X4 berbeda karena pembulatan MATLAB

<pre>Gauss-Seidel Iterative Solution after 7 iterations:
X1 = -2.0001
X2 = -1.0001
X3 = 0.0000
X4 = 1.0000
X5 = 2.0000</pre>
	Hasil 1.D

2. A wall 1 ft. thick and infinite in other directions, as shown in figure below, has an initial uniform temperature (T_i) of 100 ºF. The surface temperature (Ts) at the two sides are suddenly increased and maintained at 300 ºF . The wall is composed of nickel steel (40% Ni) with diffusivity of $\alpha = 0.1$ ft^2/hr.



The unsteady one-dimensional heat conduction equation in cartesian coordinates is written as follows :

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

a. Find analytical solution of the above PDE.
b. Derive the discrete equation of the PDE using Forward Time Central Space (FTCS) scheme.
c. Generate a numerical algorithm of the discretized equation.
d. Generate a numerical coding based the developed algorithm.
e. Compute temperature distribution within the wall as a function of time using the following sets of step sizes : $\Delta x = 0.05\ ft\ with\ various\ \ \Delta t$: $(i)\ \Delta t\ = 0.005\ hr$; (ii) $\Delta t = 0.01\ hr$ and $(iii)\ \Delta t\ = 0.05\ hr$
f. Make analysis of the results
g. Compare the numerical result with the analytical result.

Theoretical Background:

Heat equation adalah salah satu mekanisme perpindahan panas yang terjadi akibat perbedaan suhu dalam suatu material. Panas berpindah dari daerah bersuhu tinggi ke daerah bersuhu rendah melalui interaksi molekul tanpa perpindahan massa. Fenomena ini dijelaskan secara matematis oleh persamaan konduksi panas (*heat conduction equation*), yang juga dikenal sebagai persamaan difusi panas (*heat diffusion equation*).

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

Solusi analitik sangat dipengaruhi oleh kondisi awal dan kondisi batas.

Sebagai contoh :

- Batang panjang L
- Suhu awal: $T(x, 0) = f(x)$

- Batas suhu tetap (Dirichlet condition) :

$$T(0,t) = 0, T(L,t) = 0$$

Asumsikan :

$$T(x,t) = X(x) \cdot \tau(t)$$

Substitusikan ke dalam PDE :

$$X(x).\frac{d\tau}{dt} = \alpha.\tau(t).\frac{d^2X}{dx^2}$$

$$\frac{1}{\alpha\tau}.\frac{d\tau}{dt} = \frac{1}{X}.\frac{d^2X}{dx^2} = -\lambda$$

Diperoleh dua ODE:

- Untuk waktu:

$$\frac{d\tau}{dt} + \alpha\lambda\tau = 0$$

$$\tau(t) = Ae^{-\alpha\lambda t}$$

- Untuk posisi :

$$\frac{d^2X}{dx^2} + \lambda X = 0$$

Dengan syarat batas $X(0) = X(L) = 0$ solusi eigenfunction-nya:

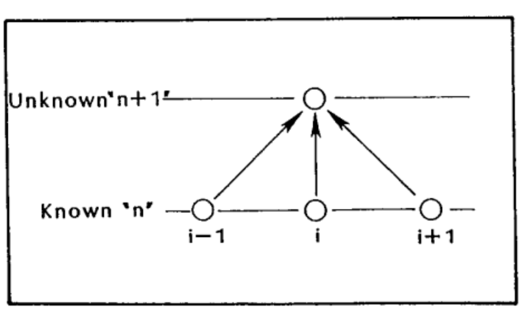
$$X_n(x) = sin\left(\frac{n\pi x}{L}\right), \lambda_n = \left(\frac{n\pi}{L}\right)^2$$

Maka Solusi umum :

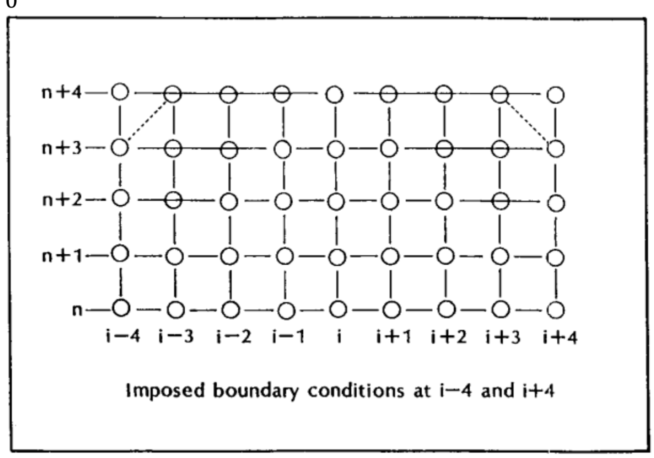
$$T(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{-\alpha\left(\frac{n\pi}{L}\right)^2 t}$$

Koefisien $B_n$ditentukan dari kondisi awal $T(x,0) = f(x)$ dengan deret Fourier:

$$B_n = \frac{2}{L}\int_0^L f(x) \sin\left(\frac{n\pi x}{L}\right) dx$$



Grid points for the explicit



Effect of the boundary conditions

Dengan demikian, PDE orde kedua telah digantikan oleh persamaan aljabar. Representasi grafis titik-titik grid dalam Persamaan

Formulasi untuk formulasi eksplisit. Menggunakan perkiraan selisih maju untuk turunan waktu dan selisih pusat untuk turunan ruang dalam Persamaan

PDE using Forward Time Central Space (FTCS)

- Forward difference untuk waktu:

$$\frac{\partial T}{\partial t} \approx \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

- Central difference untuk ruang:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

Substitusikan ke PDE:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha . \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

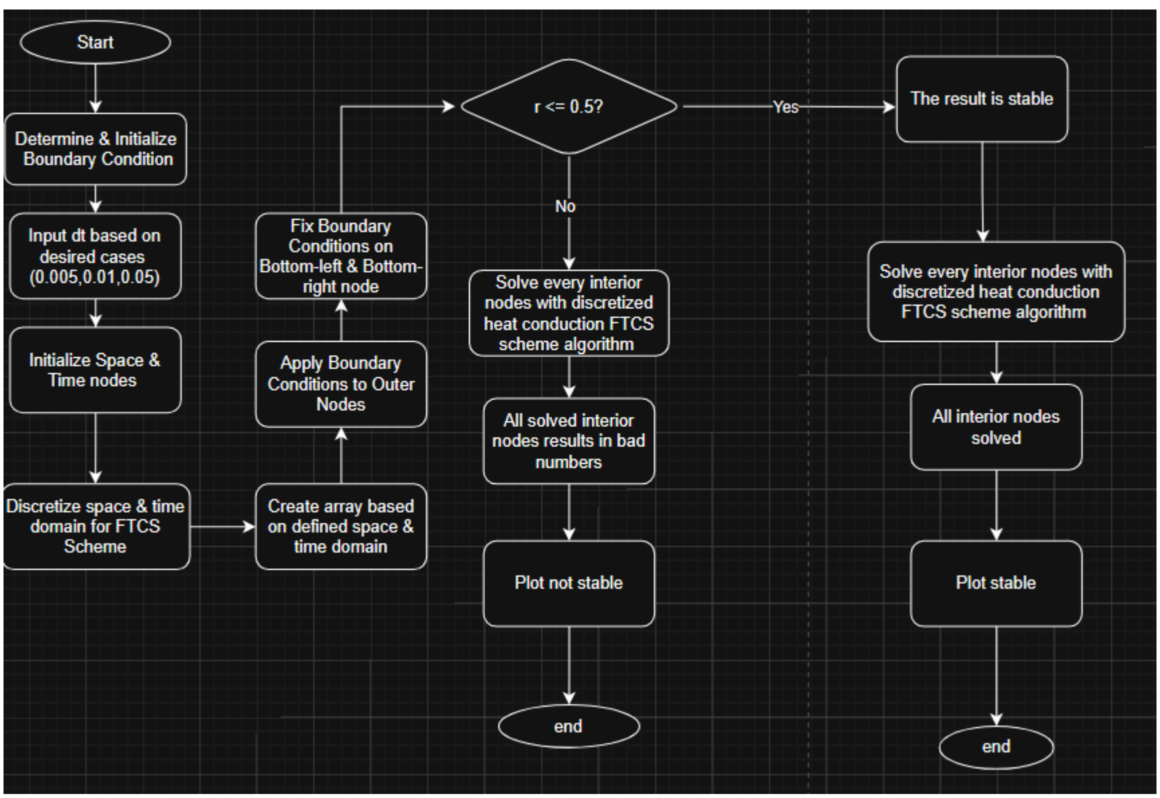
Maka bentuk *Forward Time Central Space* (FTCS):

$$T_i^{n+1} = T_i^n + r(T_{i+1}^n - 2T_i^n + T_{i-1}^n)$$

Dengan :

$$r = \frac{\alpha \Delta t}{\Delta x^2}$$

Flowchart:



Analysis:
2.a & 2b:

$$\alpha\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \;, \quad \mathbf{0 < x < 1}, \qquad \boldsymbol{t = o}, \qquad \boldsymbol{\alpha^2 = (0, 1)^2}$$

Dengan syarat batas:

$$\boldsymbol{u(0, t) = 300}, \qquad \boldsymbol{u(1, t) = 300}, \qquad \boldsymbol{t > 0}$$

Dan kondisi awal:

$$\boldsymbol{u(x, 0) = 100}, \qquad \boldsymbol{0 < x < 1}$$

Penyelesaian Analitik
Solusi Steady-State Pada kondisi tunak:

$$\frac{\partial u}{\partial t} = 0$$

$$\frac{\partial^2 u}{\partial x^2} = 0$$

Solusi umum:

$$\boldsymbol{u(x) = Ax + B}$$

Gunakan syarat batas:

- $\boldsymbol{u(0) = 300 \Rightarrow B = 300}$
- $\boldsymbol{u(1) = 300 \Rightarrow A + 300 = 300 \Rightarrow A = 0}$
- **Jadi:**

$$\boldsymbol{\theta(x) = 300}$$

Transformasi Fungsi:
Misalkan:

$$\boldsymbol{u(x, t) = \theta(x) + w(x, t) \Rightarrow w(x, t) = u(x, t) - \theta(x)}$$

$$\boldsymbol{w_t = u_t}, \qquad \boldsymbol{w_{xx} = u_{xx}}$$

Sehingga

$$\alpha^2\frac{\partial^2 u}{\partial x^2} = \frac{\partial w}{\partial t}$$

Dengan syarat:

- $\boldsymbol{w(0, t) = 0}$
- $\boldsymbol{w(1, t) = 0}$
- $\boldsymbol{w(x, 0) = 100 - 300 = -200}$

2. Analisis & Matlab Program

(a) Persamaan differensial parsial:

$$\alpha\, u_{xx} = u_t, \quad 0 < x < 1, \ t = 0, \ \alpha^2 = (0{,}1)^2$$

Boundary condition: $u(0,t) = 300$, $u(1,t) = 300$

Initial condition: $u(x,0) = 100$

Misal, $v(x)$: fungsi distribusi panas (tak bergantung t)

@ Steady-state $t \to \infty$, Perubahan panas konstan $v_t = 0 \Rightarrow V_{xx} = 0$

Maka, $v(0) = 300$ & $v(1) = 300$

$$\boxed{v(x) = \frac{(T_2 - T_1)}{L} \times T_1}$$

diperoleh: $v(x) = 300$

Misal, $u(x,t)$: temperatur barang @ posisi x & waktu t, $0 < x < 1$, $t > 0$

definisikan, $u(x,t) = v(x) + w(x,t)$, $w(x,t) \to$ fungsi transformasi

$\Rightarrow u_t = w_t$ dan $u_{xx} = v''(x) + w_{xx} = w_{xx}$

$u_{xx} = u_t \Rightarrow w_{xx} = w_t$

$$B.C: \ w(0,t) = u(0,t) - v(0) = 300 - 300 = 0$$
$$= w(1,t) = u(1,t) - v(1) = 300 - 300 = 0$$

$$I.C: \ w(x,0) = u(x,0) - v(x) =$$
$$= 100 - 300 = -200$$

$$\boxed{\begin{aligned} & w_{xx} = w_t \\ & B.C: w(0,t) = 0 \\ & \quad\quad w(1,t) = 0 \\ & I.C: w(x,0) = -200 \end{aligned}}$$

Metode Separation variable : $w(x,t) = X(x) \cdot T(t)$

$\alpha^2 \cdot w_{xx} = w_t$

$\Rightarrow \alpha^2 X''(x) \cdot T(t) = X(x) \cdot T'(t)$

$\Rightarrow \frac{X''}{X} = \frac{1}{\alpha^2} \frac{T'}{T} = -\lambda$

B.C :

$w(0,t) = X(0) \cdot T(t) = 0 \Rightarrow X(0) = 0$

$w(1,t) = X(1) \cdot T(t) = 0 \Rightarrow X(1) = 0$

(1) Eigenvalue problem : $X'' + \lambda X = 0$, $X(0)=0$, $X(1)=0$

(2) ODE : $T' + \lambda T = 0$

$\cdot\ X'' + \lambda X = 0$, $X(0) = 0$ $X(1) = 0$

Eigenvalue problem

Nilai Eigen : $\lambda_n = n^2 \cdot \pi^2$, $n = 1,2,3$

Fungsi Eigen : $X_n(x) = B_n \cdot \sin(n\pi x)$

$\cdot\ T' + \alpha^2 \cdot \lambda \cdot T = 0$ dengan $\lambda = n^2 \cdot \pi^2$

$\Rightarrow T' + n^2 \pi^2 \alpha^2 \cdot T = 0$

$\hookrightarrow T' + n^2 \pi^2 \alpha^2 \cdot T = 0 \rightsquigarrow \frac{\partial T}{\partial t} = n^2 \pi^2 \alpha^2 \cdot T$

$\Rightarrow T_n(t) = D_n \cdot e^{-n^2 \pi^2 \alpha^2 t}$

$\Rightarrow \int \frac{\partial T}{T} = \int -n^2 \pi^2 \alpha^2 \, \partial t$

$\Rightarrow \ln T = -n^2 \pi^2 \alpha^2 \cdot t + K$

$\Rightarrow T = D \cdot e^{-n^2 \pi^2 \alpha^2 \cdot t}$

I.C : $w(x,0) = -200$

$\Rightarrow \sum_{n=1}^{\infty} C_n \cdot \sin(n\pi x) = -200$

$C_n = 2 \cdot \int_0^1 -200 \cdot \sin(n\pi x) \, \partial x$

$= \left[\frac{400}{n\pi} \cdot \cos(n\pi x) \right]_0^1$

$\Rightarrow C_n = \frac{400}{n\pi} \cdot \cos(n\pi) - \frac{400}{n\pi}$

$= \frac{400}{n\pi} (-1)^n - \frac{400}{n\pi}$

$C_n = \begin{cases} -\frac{800}{n\pi}, & n \text{ ganjil} \\ 0, & n \text{ genap} \end{cases}$

diperoleh: $W(x,t) = \sum_{n=1}^{\infty} C_n \cdot \sin(n\pi x) \cdot e^{-n^2 \pi^2 \alpha^2 t}$

$$= \sum_{n \text{ ganjil}}^{\infty} -\frac{800}{n\pi} \cdot \sin(n\pi x) \cdot e^{-n^2 \pi^2 \alpha^2 \cdot t}$$

∴ PDE Solution (analytical solution) $\rightarrow u(x,t) = v(x) + w(x,t)$

$$\hookrightarrow \boxed{u(x,t) = 300 + \sum_{n \text{ odd}}^{\infty} -\frac{800}{n\pi} \cdot \sin(n\pi x) \cdot e^{-n^2 \pi^2 (0.1)^2 \cdot t}}$$

(b) PDE:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

discretize time & space:

- $\Delta t \rightarrow$ time step size $\rightarrow n = \Delta t$
- $\Delta x \rightarrow$ spatial step size (space) $\rightarrow i = \Delta x$
- $T_i^n \rightarrow$ Temperature @ position i & time m

FTCS Scheme:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2 \cdot T_i^n + T_{i-1}^n}{(\Delta x)^2}$$

Solve for T_i^{n+1}:

$$T_i^{n+1} = T_i^n + \alpha \cdot \frac{\Delta t}{(\Delta x)^2} \left(T_{i+1}^n - 2 T_i^n + T_{i-1}^n \right)$$

Define stability parameter: (r)

FTCS discrete equation untuk that equation

$$r = \alpha \cdot \frac{\Delta t}{(\Delta x)^2} \longrightarrow T_i^{n+1} = T_i^n + r\left(T_{i+1}^n - 2T_i^n + T_{i-1}^n\right)$$

$$T_i^{n+1} = r \cdot T_{i+1}^n + (1-2r) T_i^n + r \cdot T_{i-1}^n$$

$\boxed{\text{Metode ini stabil untuk } r \leq \tfrac{1}{2} \text{ atau } r \leq 0.5.}$

2c. & 2d. Numerical Algorithm for discretized equation & Numerical Coding:

Source Code Matlab:

<pre>% Input (known data)
L = 1.0;   % wall thickness in ft
alpha = 0.1;   % Diffusivity constant
t_final = 1 ;   % Assume 1 Hour time
Ti = 100.0;   % initial temperature
Ts = 300.0;   % boundary temperature
dx = 0.05;   % Spatial step
dt = input('Enter time step(dt): '); % Time Step
fprintf('Time step(dt): %d\n', dt);
Nx = L/dx;   % number of spatial steps
r = alpha * dt / dx^2;
 	
N = L/dx + 1 ; % Space nodes
M = t_final/dt + 1 ; % Time nodes</pre>
	Program menerima masukan waktu langkah (dt) dari user.
	
	alpha digunakan dalam perhitungan numerik kestabilan.
	
	r adalah bilangan penting untuk kestabilan metode eksplisit FTCS. Jika r > 0.5, solusi bisa tidak stabil.
	
	N dan M menyatakan jumlah grid dalam ruang dan waktu.

<pre>% Zero vectors of x & t
x = zeros(N, 1);
t = zeros(M, 1);
 	
% Discretize space & time domain
for i = 1:N
    x(i) = 0 + (i - 1)*dx; % step space domain
end
 	
for n = 1:M
    t(n) = 0 + (n - 1)*dt;   % step time domain
end</pre>
	Membuat vektor posisi x dan waktu t

<pre>% Outer Nodes
T_numerical = zeros(M,N);
T_numerical(:,1) = 300; %Left Boundary Condition
T_numerical(:,N) = 300; %Right Boundary Condition
T_numerical(1,2:N-1) = 100; %Initial Condition
 	
% Fix the corners by averaging overlapping BCs
T_numerical(1,1)   = (300+100) / 2;   % Bottom-left
T_numerical(1,N)   = (300+100) / 2; % Bottom-right</pre>
	Kondisi awal: tengah = 100°F, batas kiri dan kanan = 300°F.
	
	Titik pojok awal disesuaikan dengan rata-rata

<table>
<tr>
<td><pre>
% Interior Nodes
for n= 1:M-1
    for i=2:N-1
        T_numerical(n+1, i) = r*T_numerical(n,i+1) + (1-2*r)*T_numerical(n,i) + r*T_numerical(n,i-1);
    end
end

% Plotting
x = linspace(0, L, N);
figure;
hold on;
for k = 1:size(T_numerical, 1)
    plot(x, T_numerical(k, :));
end
xlabel('Position (ft)');
ylabel('Temperature (F)');
title("Steady Heat Condunction in a Wall with time step (dt): " + num2str(dt) + "hr (Gauss-Seidel)")
grid on;
</pre></td>
<td>Menghitung suhu pada titik interior dari waktu ke waktu menggunakan skema eksplisit FTCS

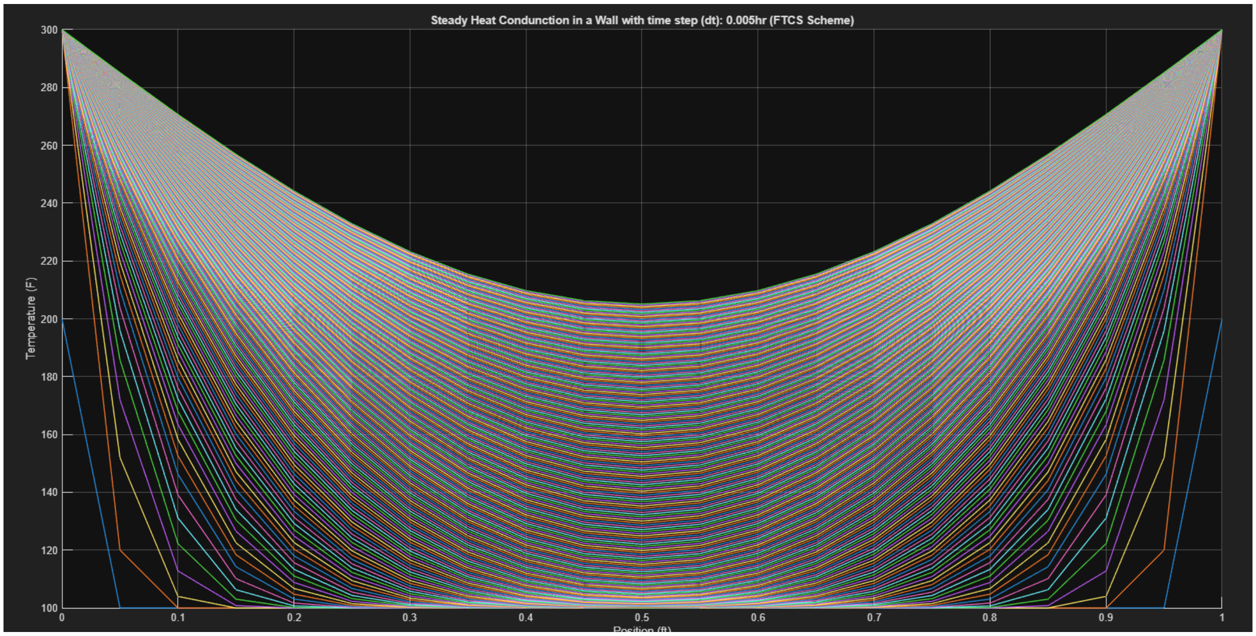
Menampilkan perubahan distribusi suhu sepanjang waktu.</td>
</tr>
<tr>
<td><pre>
% Comparison time (e.g., 1hr)
t_final = 1;
x = x';

% Compute analytical solution at t_final
n_terms = 100;   % number of odd terms
n_vec = (1:2:2*n_terms)';   % odd integers: 1,3,5,...
C_n = -800 ./ (pi * n_vec);   % Coefficients
sin_terms = sin(n_vec * pi * x' / L); % sin(n*pi*x)
exp_terms = exp(-alpha * (n_vec * pi / L).^2 * t_final); % exp(-α(nπ/L)^2*t)
transient_sum = sum(C_n .* sin_terms .* exp_terms, 1);
T_analytical = Ts + transient_sum';   % Add 300 to get full solution
</pre></td>
<td>Berdasarkan deret Fourier untuk solusi analitik dari persamaan difusi panas dengan syarat batas tetap.

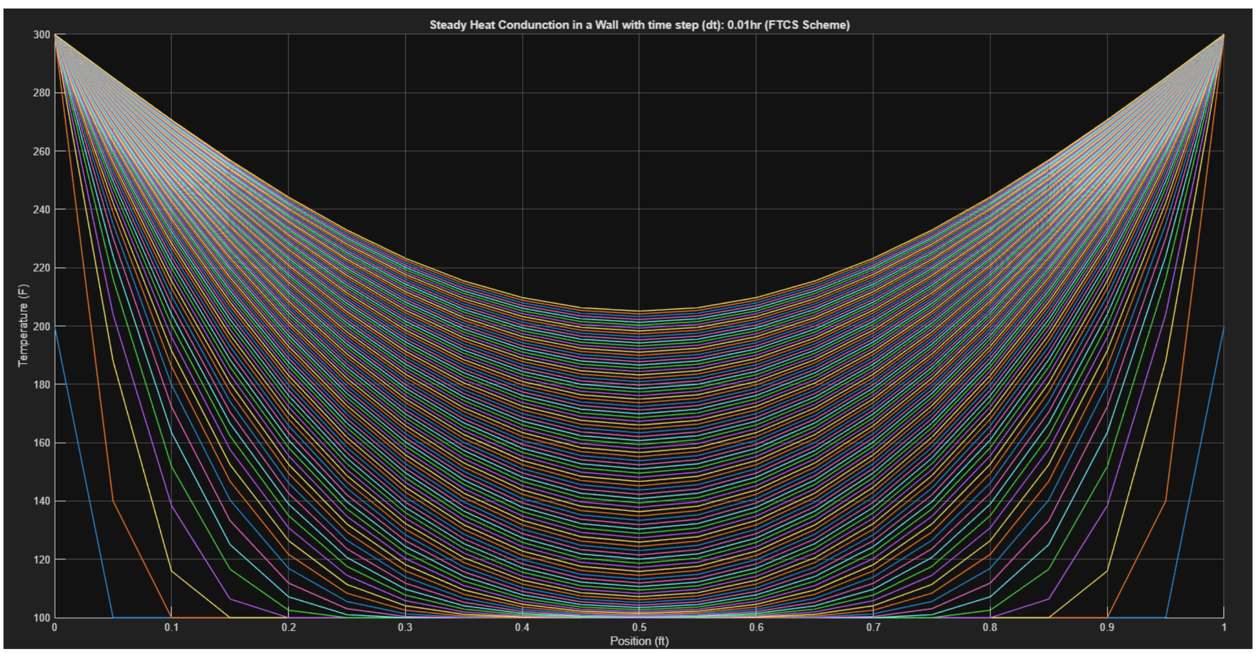
Diperoleh solusi eksak pada waktu akhir
$t = 1$</td>
</tr>
<tr>
<td><pre>
% Final Time Comparison Plot
n_idx = round(t_final / dt) + 1; % Index in time for final step

figure;
plot(x, T_numerical(n_idx, :), 'r-', 'LineWidth', 3);
hold on;
plot(x, T_analytical, 'k--', 'LineWidth', 3);
xlabel('Position (ft)');
ylabel('Temperature (°F)');
title(sprintf('Final Time Comparison at t = %.2f hr', t_final));
legend('Numerical (FTCS)', 'Analytical');
grid on;
</pre></td>
<td>Menampilkan grafik suhu hasil simulasi numerik dan solusi analitik untuk memverifikasi akurasi.</td>
</tr>
</table>

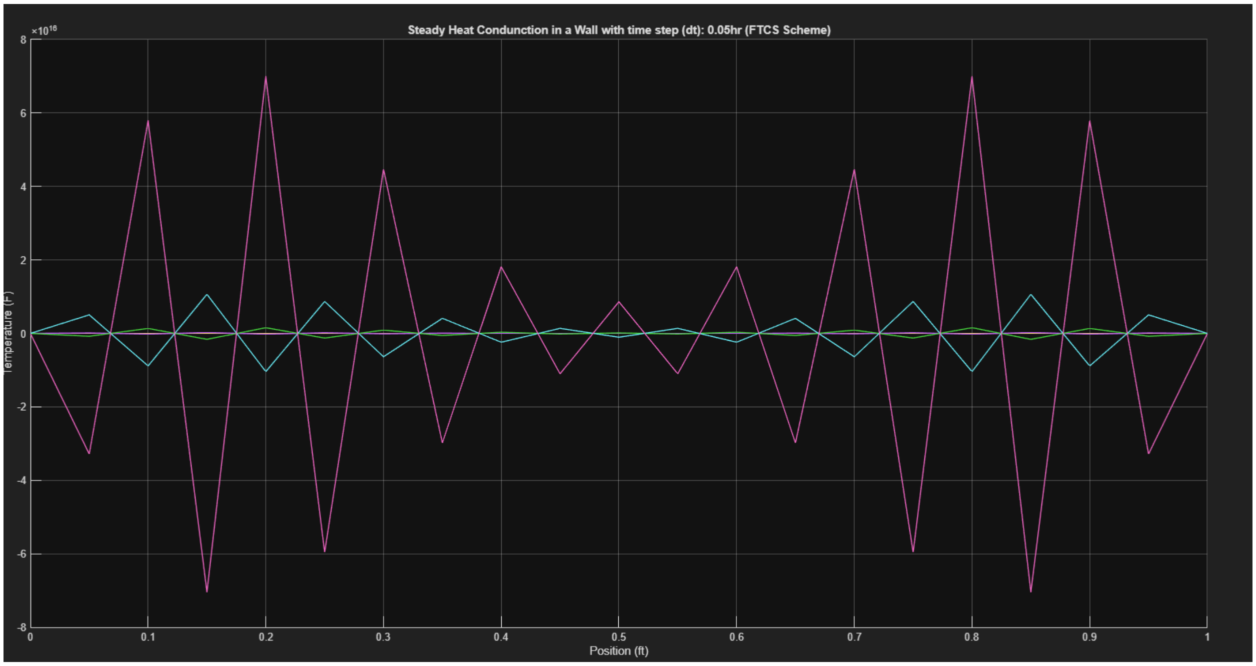
2e. Compute temperature distribution within the wall as a function of time with 3 Cases (0.005, 0.01,0.05)



Case 1 (Stable)



Case 2 (Stable)



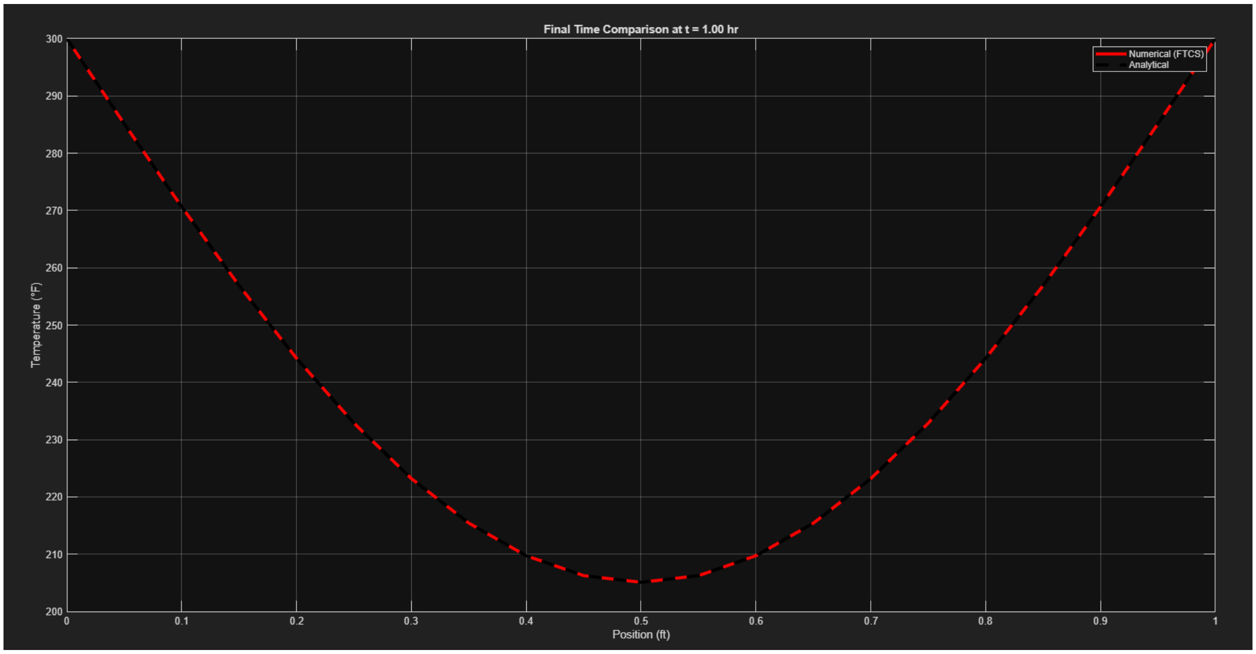
Case 3 (Not Stable)

2f . Analysis of the results

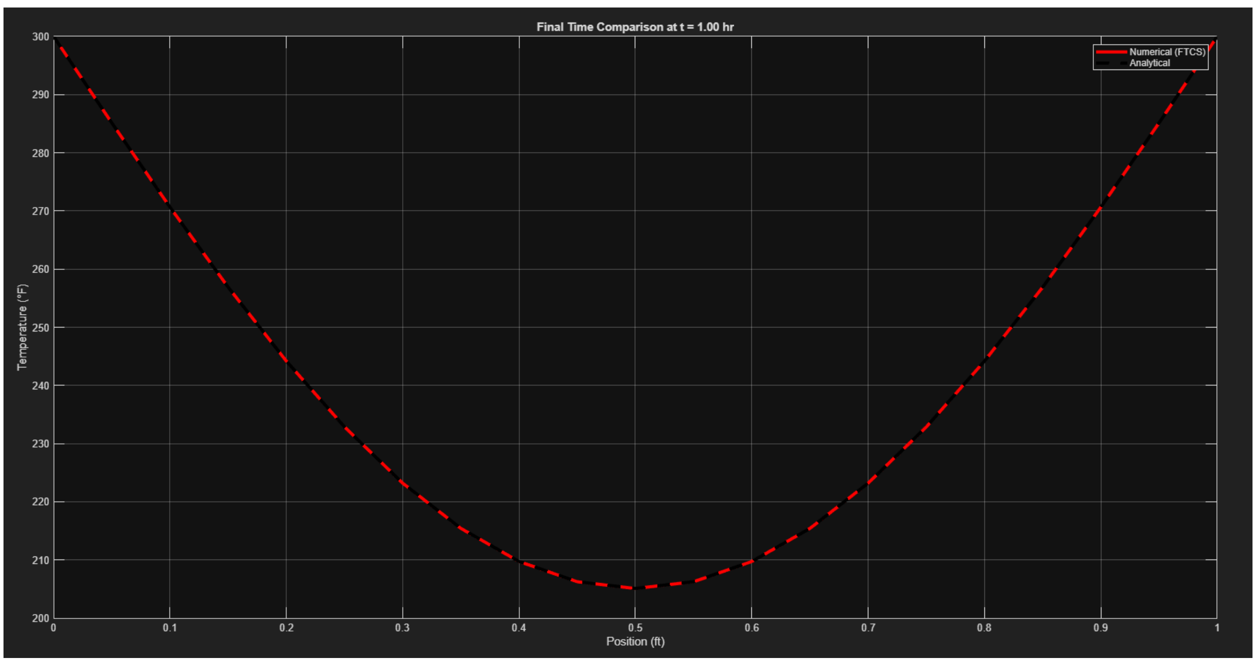
- **Case (I)** memiliki r = $(0.1)\frac{0.005}{0.0025} = 0.2$, sehingga menghasilkan solusi yang **stabil**, maka plotnya dapat divisualisasi
- **Case (II)** memiliki r = $(0.1)\frac{0.01}{0.0025} = 0.4$, sehingga menghasilkan solusi yang **stabil**, maka plotnya dapat divisualisasi
- **Case (III)** memiliki r = $(0.1)\frac{0.05}{0.0025} = 2.0$, sehingga menghasilkan solusi yang tidak **stabil**, menyebabkan distribusi temperature berosilasi linear, sehingga plotnya tidak dapat divisualisasi atau hasil numeriknya tidak bermakna.

2g. Compare the numerical result with the analytical result

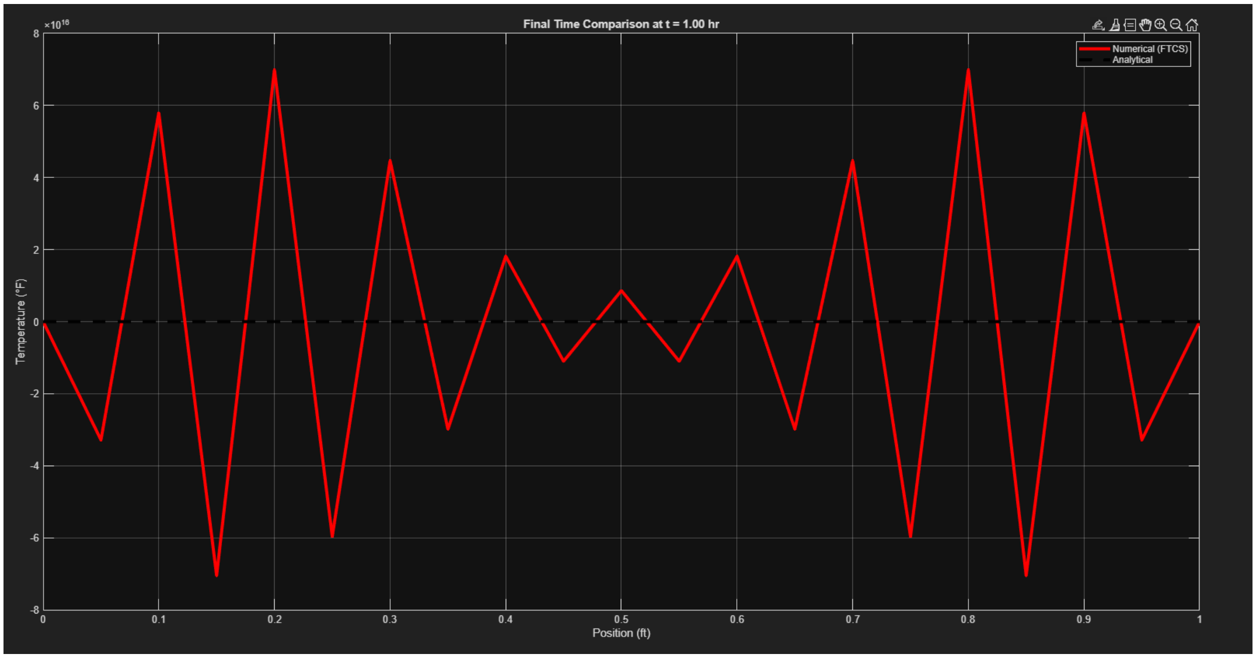
- Untuk kasus stabil (i dan ii), kurva hasil numerik sangat cocok (overlapped dengan hasil numerik) dengan kurva solusi analitik. Hal ini memvalidasi bahwa solusi numerik yang didapat dari 2a akurat.
- Akan ada sedikit perbedaan kecil antara keduanya yang akan semakin kecil jika dx dan dt diperkecil (selama r-nya tetap stabil).
- Untuk kasus tidak stabil (iii), perbandingan tidak dapat dilakukan karena hasil numeriknya tidak bermakna.



Perbandingan Hasil Numerical & Analytical Case 1 (dt = 0.005)



Perbandingan Hasil Numerical & Analytical Case 2 (dt=0.01)



Perbandingan Hasil Numerical & Analytical Case 3 (dt=0.05)

3. Use Gauss-Seidel Method and the code that you have developed in question no. 1 to solve for the temperature of the steady state heated plate in figure P.3.1. Do the iteration with the error to ε_s = 1%. The temperatures on all four sides are stated by the numbers on the part of your NIM. For example, NIM 13623020 will give these boundary condition. **Plot your results of temperature distribution on the plate**.

NIM =

1	3	6	2	3	0	2	0

136ºC

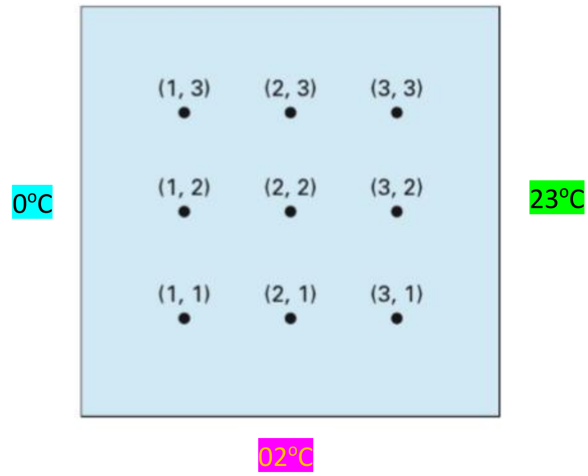


Figure P.3.1. Heated Plate

Theoretical Background:

Finite-difference Elliptical Equation merupakan salah satu metode penyelesaian persamaan diferensial yang berubah dengan menggunakan *central difference* method untuk daerah permukaan. Kita dapat mengubah masalah tersebut menjadi persamaan Laplace:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Dari situ kita bisa memperkirakan bahwa :

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} + O(\Delta x^2) \;, \frac{\partial^2 T}{\partial y^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta y^2} + O(\Delta y^2)$$

Untuk bagian persegi, $\Delta x = \Delta y$, jadi :

$$T_{i+1,j} + T_{i-1,j} + T_{J+1,i} + T_{j-1,i} - 4T_{i,j} = 0$$

Persamaan perbedaan Laplacian :

$$T_{21} + T_{01} + T_{12} + T_{10} - 4T_{11} = 0$$

Kita tahu bahwa untuk setiap sudut harus memiliki nilai konstan :

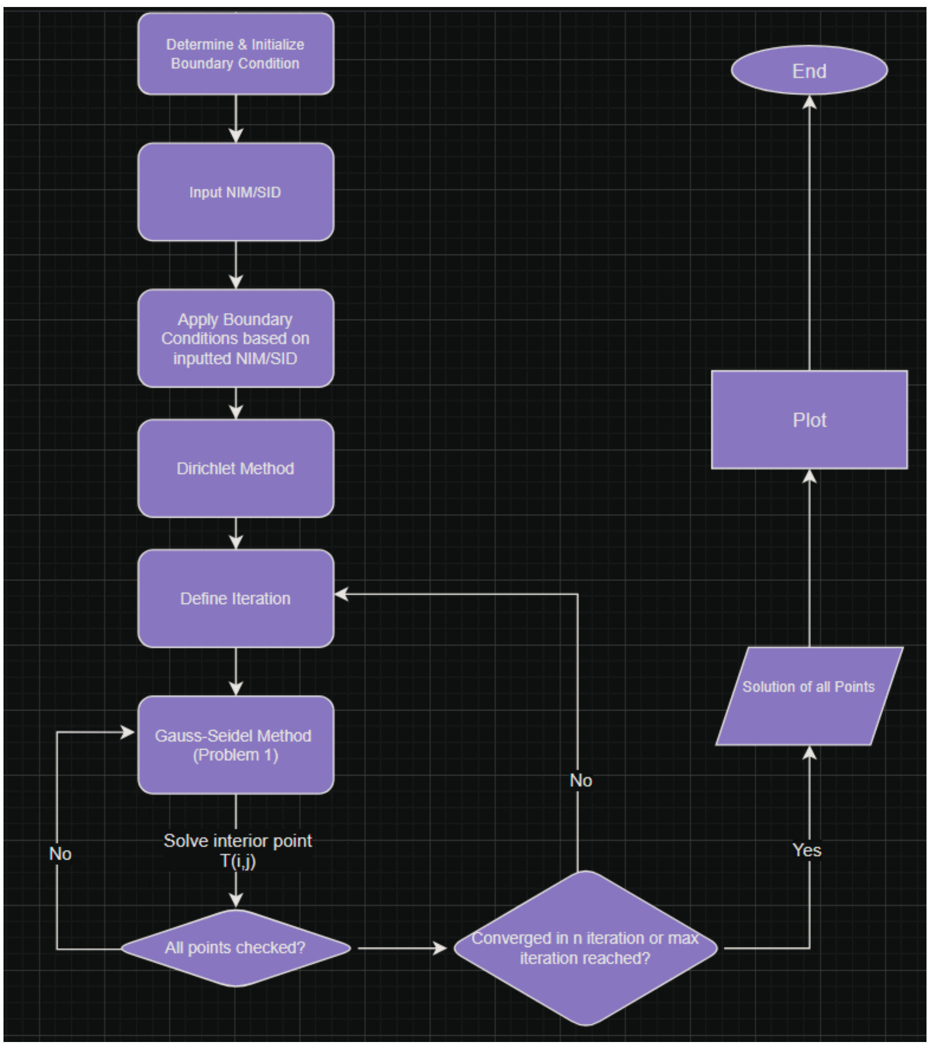
$$T_{0,k} = 1℃, \qquad T_{k,0} = 5℃, \qquad T_{n+1,k} = 21℃, \qquad T_{k,n+1} = 136℃$$

kita bisa menemukan persamaan :

$$
\begin{array}{cccccccccl}
4T_{11} & -T_{21} & & -T_{12} & & & & & & = T_{0,k} + T_{k,0} \\
-T_{11} & +4T_{21} & -T_{31} & & -T_{22} & & & & & = T_{k,0} \\
 & -T_{21} & +4T_{31} & & & -T_{32} & & & & = T_{n+1,k} + T_{k,0} \\
-T_{11} & & & 4T_{12} & -T_{22} & & -T_{13} & & & = T_{0,k} \\
 & -T_{21} & & -T_{12} & +4T_{22} & -T_{32} & & -T_{23} & & = 0 \\
 & & -T_{31} & & -T_{22} & -4T_{32} & & & -T_{33} & = T_{n+1,k} \\
 & & & -T_{12} & & & +4T_{13} & -T_{23} & & = T_{0,k} + T_{k,n+1} \\
 & & & & -T_{22} & & -T_{13} & +4T_{34} & -T_{33} & = T_{k,n+1} \\
 & & & & & -T_{32} & & -T_{23} & +4T_{33} & = T_{n+1,k} + T_{k,n+1}
\end{array}
$$

Persamaan ini diselesaikan dengan menggunakan metode *Gauss-Seidel* atau Eliminasi Gauss. Pada kasus ini, Metode *Gauss-Seidel* khusus untuk sistem persamaan linear disebut juga metode *Liebman*, sedangkan penyelesaian matriks seperti Eliminasi Gauss disebut metode *Dirichlet*.

Flowchart:



Source Code:

`clear all; close all ; clc`
 	
`% Input SID/NIM`
`NIM = input("Input NIM/SID: ");`
`fprintf('SID/NIM Number: %d\n', NIM);`
 	
`NIM = num2str(NIM);  % Convert to array of string;`
	Input NIM dari user.

`% Discretization of Geometry`
`Length = 5;`
`Width = 5;`
	Definisikan Geometri pelat logam (5x5) .

`elX = 10;% number of elements in x direction`
`elY = 10; % number of elements in y direction`
`Nx = elX + 1; % number of nodes in x direction`
`Ny = elY + 1; % number of nodes in y direction`
 	
`dx = linspace(0, Length, Nx);`
`dy = linspace(0, Width, Ny);`
	Inisialisasi jumlah element x dan y
	Dan jumlah Nodes di x dan y.
	
	Semakin banyak nodesnya hasil plot nya akan lebih smooth.
	
	Nx= total nodes arah x .
	Ny= total nodes arah y.
	
	dx = Total space step berdasarkan nndx.
	dy = Total time step berdasarlam nndy.

`% Boundary Condition & Initial Condition` `T = zeros(Nx, Ny); % Initialize solution for T`	Inisialiasi matrix(T) 0 sebanyak total Nx & Ny (Nx x Ny). Contoh : (10x10) atau (5x5) karena MATLAB dimulai dari 1 arraynya

`% Boundary Conditions from NIM`
`T_top = str2double(NIM(1:3));`
`T_right = str2double(NIM(4:5));`
`T_bottom = str2double(NIM(6:7));`
`T_left = str2double(NIM(8));`
Output:
`T_top = 131, T_right = 23, T_bottom = 6, T_left = 9`
	Menentukan boundary condition berdasarkan NIM yang diinput ke 4 sisi pelat logam.
	
	Contoh : 13123069

<table>
<tr>
<td>

```
% Apply boundary conditions
T(1, :) = T_top;          % Top side
T(Nx, :) = T_bottom;       % Bottom side
T(:, 1) = T_left;         % Left side
T(:, Ny) = T_right;        % Right side
```

</td>
<td>Meaplikasikan boundary condition ke 4 sisi pleat berdasarkan hasil pembagian NIM.</td>
</tr>
<tr>
<td>

```
% Fix the corners by averaging overlapping BCs
T(1,1)    = (T_top + T_left) / 2;       % Top-left
T(1,Ny)    = (T_top + T_right) / 2;      % Top-right
T(Nx,1)    = (T_bottom + T_left) / 2;    % Bottom-left
T(Nx,Ny)    = (T_bottom + T_right) / 2;  % Bottom-right
```

Output:

```
Top_side = 70
Bottom_side = 77
Left_side = 7.500000e+00
Right_side = 1.450000e+01
```

</td>
<td>Memperbaiki masalah Boundary Conditions yang overlapped di MATLAB. Dengan menghitung average dari total BC yang overlapped.

Contoh : 13123***</td>
</tr>
<tr>
<td>

```
% Initial Matrix T
disp(T);
```

Output:
dilampirkan dibawah tabel source code

</td>
<td>Initial condition Matrix T</td>
</tr>
<tr>
<td>

```
% Numerical procedure
% Convergence criteria
epsilon_s = 1; % convergence tolerance in %
max_iter = 10000; % maximum iteration

% Relaxition Paramter for faster calculation
omega = 1; % Relaxation Parameter (1 for Gauss-Seidel)
```

</td>
<td>Dari soal, diketahui convergence tolerance nya sebesar $\varepsilon_s = 1\%$. Maximum iterasi didefinisikan agar algoritma bisa melakukan iterasi berulang-ulang hingga convergence sampai atau iterasi maks tercapai

Relaksasi Parameter (ω) sebesar 1 untuk Gauss-Seidel</td>
</tr>
</table>

Algoritma Gauss-Seidel (dari Problem 1)

```
% Gauss-Seidel Iteration
for iter = 1:max_iter
    max_error = 0;
    for i = 2:N-1
        for j = 2:N-1
            prev_iter = T(i,j);
            T(i,j) = (1 - omega)*T(i,j) + omega * 0.25 * (T(i+1,j) + T(i-1,j) + T(i,j+1) + T(i,j-1));
            error = abs((T(i,j) - prev_iter)/T(i,j)) * 100;
            if error > max_error
                max_error = error;
            end
        end
    end

    if max_error < epsilon_s
        fprintf('Converged in %d iterations with max error = %.4f%%\n', iter, max_error);
        break
    end
end
```

Hasil Output Source Code: (13123069)

```
70.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000   77.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 9.0000         0         0         0         0         0         0         0         0         0   23.0000
 7.5000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000   14.5000
```

Matriks Kondisi Awal

```
% Display final interior values
fprintf('Final Temperature at Interior Points (%.f x %.f):\n',elX-1,elY-1);
disp(T(2:Nx-1, 2:Ny-1));
```

```
Converged in 31 iterations with max error = 0.9687%
Final Temperature at Interior Points (9 x 9):
   68.3780   90.8769  100.5463  104.9046  106.4657  105.8887  102.6836   94.6648   75.1959
   42.7858   63.8182   75.6937   81.9183   84.3719   83.6701   79.3855   69.9106   52.1700
   30.1804   46.2720   56.9249   63.1654   65.8781   65.4200   61.5727   53.6066   40.6403
   22.9494   34.5940   43.0879   48.4926   51.0842   51.0162   48.2265   42.5179   33.8588
   18.3278   26.5210   32.8877   37.2128   39.5042   39.8114   38.1611   34.6016   29.3507
   15.1336   20.7077   25.2496   28.5166   30.4324   31.0147   30.3445   28.5838   26.0091
   12.7532   16.3003   19.3330   21.6423   23.1418   23.8535   23.9067   23.5536   23.1556
   10.7750   12.6921   14.4780   15.9330   16.9750   17.6376   18.0893   18.6962   20.0968
    8.7797    9.3926   10.1631   10.8539   11.3921   11.8051   12.2441   13.1189   15.5539
```

Interior Matriks Kondisi Akhir

```
% Full Grid Display
fprintf('Full Temperature Grid (%.f x %.f):\n',elX,elY);
disp(T);
```

```
Full Temperature Grid (10 x 10):
   70.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000  131.0000   77.0000
    9.0000   68.3780   90.8769  100.5463  104.9046  106.4657  105.8887  102.6836   94.6648   75.1959   23.0000
    9.0000   42.7858   63.8182   75.6937   81.9183   84.3719   83.6701   79.3855   69.9106   52.1700   23.0000
    9.0000   30.1804   46.2720   56.9249   63.1654   65.8781   65.4200   61.5727   53.6066   40.6403   23.0000
    9.0000   22.9494   34.5940   43.0879   48.4926   51.0842   51.0162   48.2265   42.5179   33.8588   23.0000
    9.0000   18.3278   26.5210   32.8877   37.2128   39.5042   39.8114   38.1611   34.6016   29.3507   23.0000
    9.0000   15.1336   20.7077   25.2496   28.5166   30.4324   31.0147   30.3445   28.5838   26.0091   23.0000
    9.0000   12.7532   16.3003   19.3330   21.6423   23.1418   23.8535   23.9067   23.5536   23.1556   23.0000
    9.0000   10.7750   12.6921   14.4780   15.9330   16.9750   17.6376   18.0893   18.6962   20.0968   23.0000
    9.0000    8.7797    9.3926   10.1631   10.8539   11.3921   11.8051   12.2441   13.1189   15.5539   23.0000
    7.5000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000    6.0000   14.5000
```

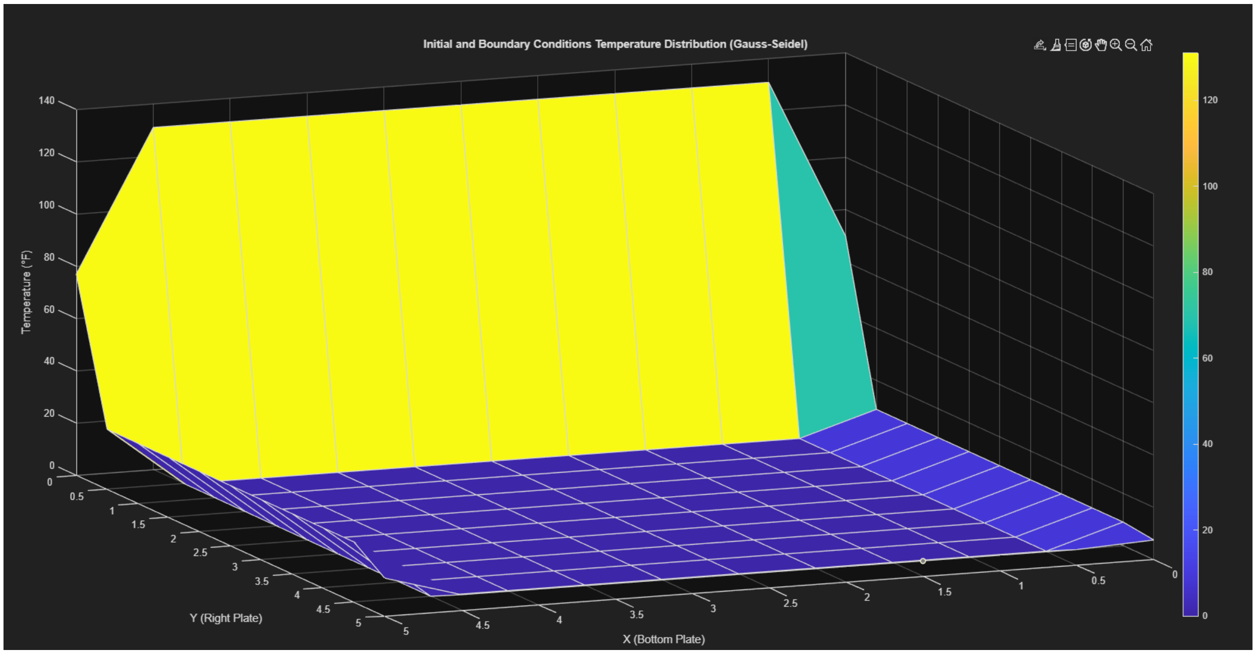
Matriks Full Grid

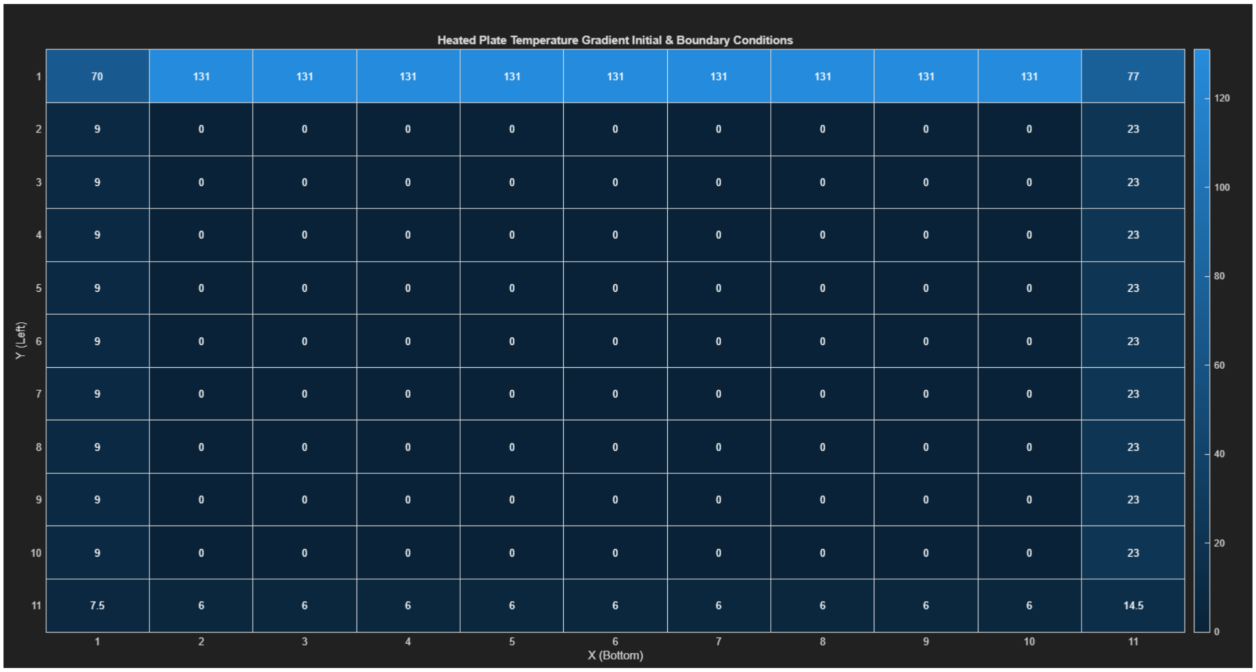
Hasil Plot:

```
% PLot initial and boundary conditions
figure;
[x,y] = meshgrid(dx,dy);
surf(x,y,T);
colorbar;
xlabel("X (Bottom Plate)"); ylabel("Y (Right Plate)"); zlabel("Temperature (°F)");
title('Initial and Boundary Conditions Temperature Distribution (Gauss-Seidel)');
view(45,30);

% Heatmap IC & BC  Plots
figure;
h = heatmap(T);
h.Title = 'Heated Plate Temperature Gradient Initial & Boundary Conditions';
h.XLabel = 'X (Bottom)';
h.YLabel = 'Y (Left)';
```

Plot code matlab Kondisi Awal



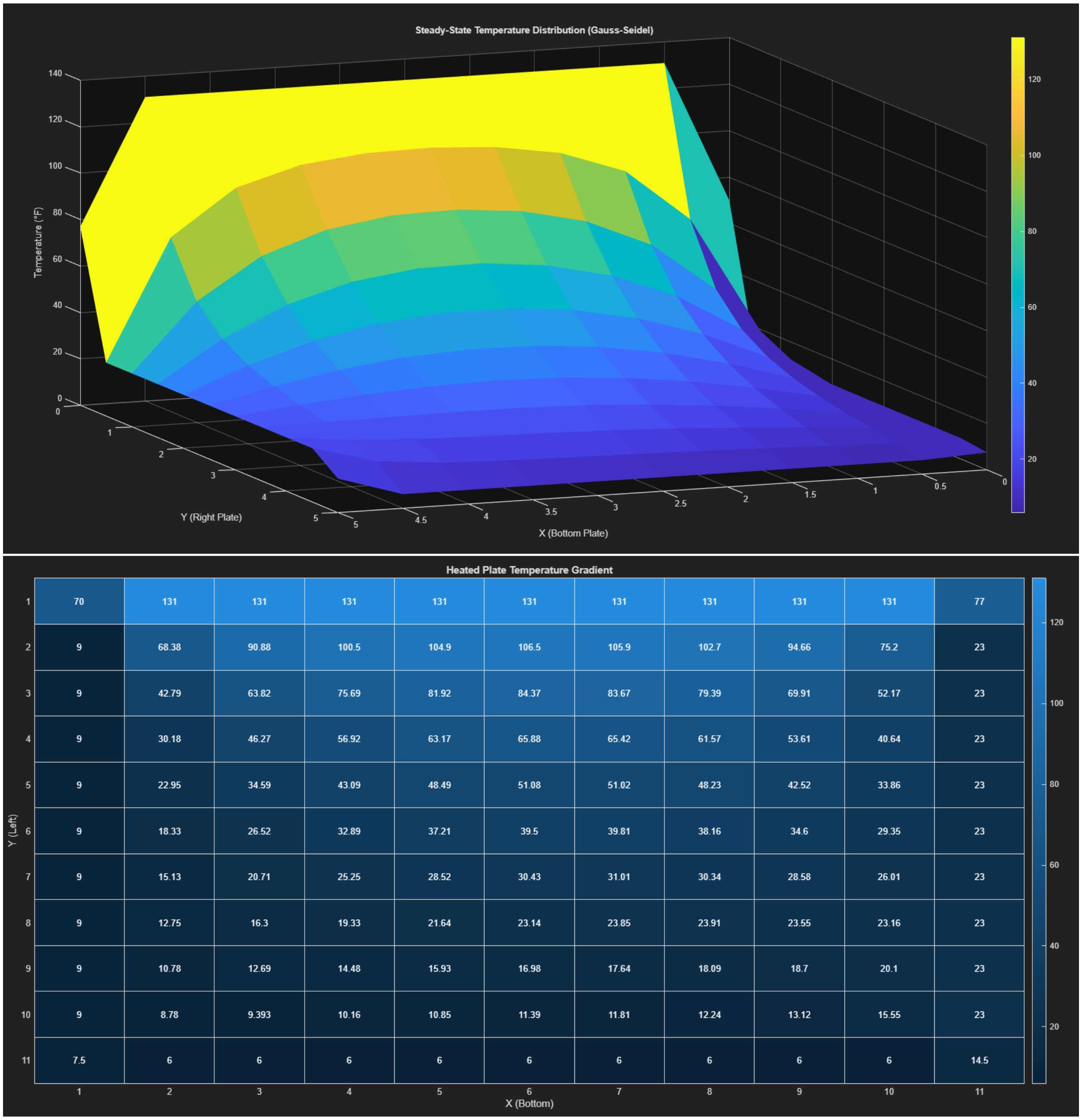


Kondisi Awal

```
% Plotting the result
[X, Y] = meshgrid(dx, dy);
figure;
surf(X, Y, T, 'EdgeColor', 'none');
xlabel('X (Bottom Plate)');
ylabel('Y (Right Plate)');
zlabel('Temperature (°F)');
title('Steady-State Temperature Distribution (Gauss-Seidel)');
colorbar;
view(45,30);


% Heatmap Plots
figure;
h = heatmap(T);
h.Title = 'Heated Plate Temperature Gradient';
h.XLabel = 'X (Bottom)';
h.YLabel = 'Y (Left)';
```

Plot code matlab Kondisi Akhir



Kondisi akhir

4. Deflection of aluminum beam that is clamped at one end and given the load as shown in figure P.4.1 can be calculated using the Bernoulli-Euler theory as follows:

$$\frac{M}{EI} = \frac{y''}{[1+(y')^2]^{3/2}} \qquad \text{(E.4.1)}$$

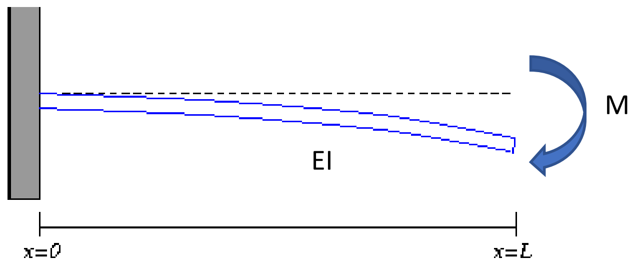


Figure P.4.1.

Equation (E.4.1) can be solved numerically by writing it into:

$$y'' = \frac{M}{EI}[1 + (y')^2]^{3/2} \qquad \text{(E.4.2)}$$

and then, equation (E.4.2) can be written into two equations:

$$y' = z \qquad \text{(E.4.3)}$$

$$z' = \frac{M}{EI}[1 + z^2]^{3/2} \qquad \text{(E.4.4)}$$

with the boundary condition y (0) = z (0) = 0

Data of beams are : E= Five last numbers of your NIM kg/mm^2 , I=1500 mm^4 , L=200 mm, M= Four last numbers of your NIM mm kg. For Example: NIM 13621021, E = 21021 kg/mm^2 and M=1021 mm kg.

a) Write the formula of Runge-Kutta with c/p coefficient using Modified Euler for second order R-K method and sketch the graphic to get y_{m+1} from y_m and z_{m+1} from z_m

b) Using the boundary condition $y_0 = y(0) = 0$ dan $z_0 = z(0) = 0$, determine y_1 and z_1 if h=20 mm. (use the 2nd order R-K method).

c) Using the same boundary conditions in b), determine the y and z with the 4th order Runge-Kutta R-K Method

d) Plot your results in b) and c) in the same graph

Theoretical Background:

Runge-Kutta secara umum merupakan solusi Persamaan Diferensial Biasa yang mencapai akurasi dari aproksimasi Deret Taylor tanpa memerlukan perhitungan turunan lebih tinggi tambahan. Metode ini didefinisikan dengan persamaan berikut (Persamaan 38) :

$$\boldsymbol{y_{i+1}} = \boldsymbol{y_i} + \phi(xi, yi, h)h$$

$\phi(xi, yi, h)h$ adalah *slope function gain* dari setiap interval, yang dapat didefinisikan sebagai :

$$\phi = a_{1\boldsymbol{k_1}} + a_2\boldsymbol{k_2} + \cdots + a_n\boldsymbol{k_n}$$

a adalah kontan dan k adalah:

$$\boldsymbol{k_1} = \boldsymbol{f}(x_i, \boldsymbol{y_i})$$
$$\boldsymbol{k_2} = \boldsymbol{f}(x_i + p_1h, \boldsymbol{y_i} + q_{11}k_1h)$$
$$\vdots$$
$$\boldsymbol{k_n} = f\left(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + \cdots + q_{n-1,n-1}k_{n-1}h\right)$$

p dan q sebagai konstan. Dalam kasus ini, metode untuk solve nya adalah ***2nd Order Range-Kutta***.
, bentuk persamaan yang digunakan akan menjadi:

$$\boldsymbol{y_{i+1}} = \boldsymbol{y_i} + (a_1\boldsymbol{k_1} + a_2\boldsymbol{k_2})h$$
$$\boldsymbol{k_1} = \boldsymbol{f}(x_i, \boldsymbol{y_i})$$
$$\boldsymbol{k_2} = \boldsymbol{f}(x_i + p_1h, \boldsymbol{y_i} + q_{11}k_1h)$$

Dengan menggunakan ***Euler's improved method***, kita bisa temukan konstan nya seperti dibawah ini:

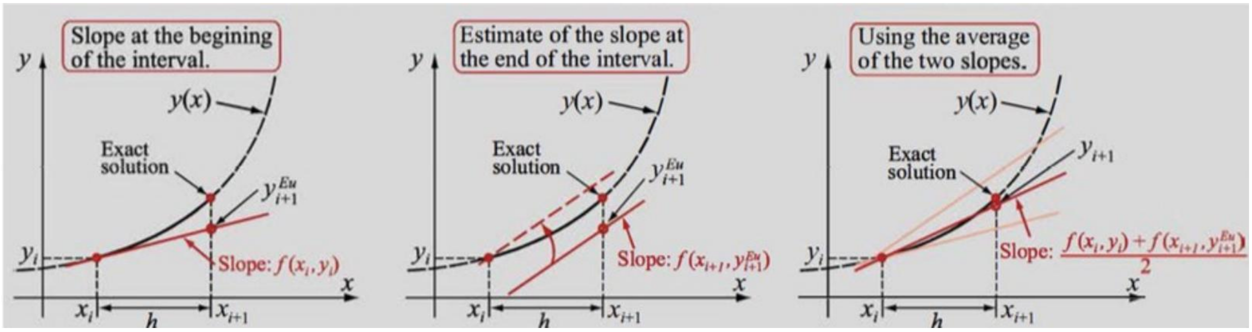
$$a_1 = \frac{1}{2}; a_2 = \frac{1}{2}; p_1 = 1; q_1 = 1$$

Sehingga bisa mengubah menjadi:

$$\boldsymbol{y_{i+1}} = \boldsymbol{y_i} + (\frac{1}{2}\boldsymbol{k_1} + \frac{1}{2}\boldsymbol{k_2})h$$

$$x_{i+1} = \mathbf{x}_i + h$$
$$\boldsymbol{y_{i+1}^{Eul}} = \boldsymbol{y_i} + \boldsymbol{k_1}h$$

Lalu sketch grafik dalam axis satu-dimensi:



Kita akan mendapatkan:

$$\mathbf{y} = \begin{bmatrix} y \\ z \end{bmatrix}, \qquad \mathbf{y}' = \mathbf{f}(x, \mathbf{y}) = \begin{bmatrix} y'(x,y,z) \\ z'(x,y,z) \end{bmatrix} = \begin{bmatrix} z \\ \frac{M}{EI}[1+z^2]^{3/2} \end{bmatrix}$$

Sehingga mendapatkan persamaan:

$$\mathbf{y}_{i+1}^{\mathrm{Eul}} = \begin{bmatrix} y_i \\ z_i \end{bmatrix} + \mathbf{k}_1 h = \begin{bmatrix} y_i + z_i h \\ z_i + \frac{M}{EI}\left[1 + z_i^2\right]^{3/2} h \end{bmatrix}$$

$$\mathbf{k}_2 = \begin{bmatrix} y'\left(x_{i+1}, y_{i+1}^{\mathrm{Eul}}, z_{i+1}^{\mathrm{Eul}}\right) \\ z'\left(x_{i+1}, y_{i+1}^{\mathrm{Eul}}, z_{i+1}^{\mathrm{Eul}}\right) \end{bmatrix} = \begin{bmatrix} z_{i+1}^{\mathrm{Eul}} \\ \frac{M}{EI}\left[1 + \left(z_{i+1}^{\mathrm{Eul}}\right)^2\right]^{3/2} \end{bmatrix}$$

$$\begin{bmatrix} y_{i+1} \\ z_{i+1} \end{bmatrix} = \begin{bmatrix} y_i \\ z_i \end{bmatrix} + \frac{h}{2}\begin{bmatrix} z_i + z_{i+1}^{\mathrm{Eul}} \\ \frac{M}{EI}\left[1 + z_i^2\right]^{3/2} + \frac{M}{EI}\left[1 + \left(z_{i+1}^{\mathrm{Eul}}\right)^2\right]^{3/2} \end{bmatrix} = \begin{bmatrix} y_i + \frac{1}{2}\left(z_i + z_{i+1}^{\mathrm{Eul}}\right)h \\ z_i + \frac{M}{2EI}\left[\left[1 + z_i^2\right]^{3/2} + \left[1 + \left(z_{i+1}^{\mathrm{Eul}}\right)^2\right]^{3/2}\right]h \end{bmatrix}$$

Untuk h=20mm, didapat hasil berikut:

$$z_1^{\mathrm{Eul}} = z_0 + \frac{M}{EI}[1 + z_0^2]^{\frac{3}{2}} h = 0 + \frac{M}{EI}[1 + 0^2]^{\frac{3}{2}} h = \frac{M}{EI} h$$

Classical form of Range-Kutta 4^th^ Order bisa di kembangkan dari ***2^nd^ Order*** menjadi berikut:

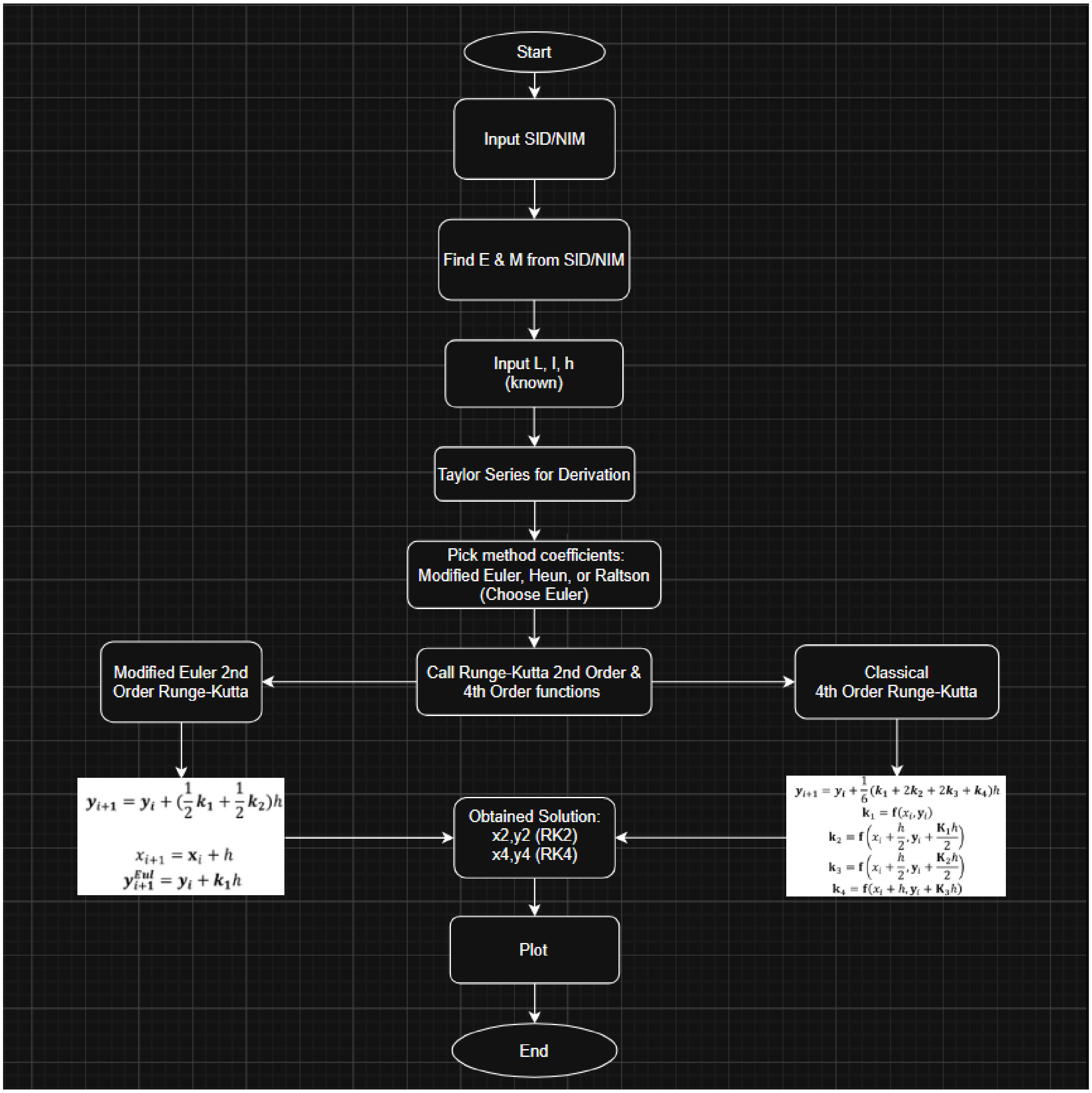
$$\boldsymbol{y_{i+1}} = \boldsymbol{y_i} + \frac{1}{6}(\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4})h$$
$$\mathbf{k}_1 = \mathbf{f}(x_i, \mathbf{y}_i)$$
$$\mathbf{k}_2 = \mathbf{f}\left(x_i + \frac{h}{2}, \mathbf{y}_i + \frac{\mathbf{K}_1 h}{2}\right)$$
$$\mathbf{k}_3 = \mathbf{f}\left(x_i + \frac{h}{2}, \mathbf{y}_i + \frac{\mathbf{K}_2 h}{2}\right)$$
$$\mathbf{k}_4 = \mathbf{f}(x_i + h, \mathbf{y}_i + \mathbf{K}_3 h)$$

Dari persamaan di atas, kita bisa menentukan:

$$\boldsymbol{y_{i+1}} = \boldsymbol{y_i} + \frac{1}{6}(\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4})h$$
$$\boldsymbol{z_{i+1}} = \boldsymbol{z_i} + \frac{1}{6}(\boldsymbol{\alpha_1} + 2\boldsymbol{\alpha_2} + 2\boldsymbol{\alpha_3} + \boldsymbol{\alpha_4})h$$
$$k_1 = \frac{M}{EI}\left[1 + z_i^2\right]^{3/2}$$
$$k_2 = \frac{M}{EI}\left[1 + \left(z_i + \frac{k_1 h}{2}\right)^2\right]^{3/2}$$
$$k_3 = \frac{M}{EI}\left[1 + \left(z_i + \frac{k_2 h}{2}\right)^2\right]^{3/2}$$
$$k_4 = \frac{M}{EI}[1 + (z_i + k_3 h)^2]^{3/2}$$
$$\alpha_1 = z_i$$
$$\alpha_2 = z\left(x_i + \frac{h}{2}, y_i + \frac{z_i h}{2}\right)$$

$$\alpha_3 = z\left(x_i + \frac{h}{2}, y_i + \frac{k_2 h}{2}\right)$$
$$\alpha_4 = z\left(x_i + \frac{h}{2}, y_i + k_3 h\right)$$

Flowchart:



Start
Input SID/NIM
Find E & M from SID/NIM
Input L, I, h
(known)
Taylor Series for Derivation
Pick method coefficients:
Modified Euler, Heun, or Raltson
(Choose Euler)
Modified Euler 2nd Order Runge-Kutta
Call Runge-Kutta 2nd Order & 4th Order functions
Classical 4th Order Runge-Kutta
$y_{i+1} = y_i + (\frac{1}{2}k_1 + \frac{1}{2}k_2)h$
$x_{i+1} = x_i + h$
$y_{i+1}^{Eul} = y_i + k_1 h$
Obtained Solution:
x2,y2 (RK2)
x4,y4 (RK4)
$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$
$k_1 = f(x_i, y_i)$
$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{K_1 h}{2}\right)$
$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{K_2 h}{2}\right)$
$k_4 = f(x_i + h, y_i + K_3 h)$
Plot
End

Source Code

```
clear, clc, clf ;
```

Membersihkan page

```
% SID/NIM input
NIM = input('Enter the SID/NIM number: ');
fprintf('SID/NIM Number: %d\n', NIM);
```

User memasukkan NIM sebagai angka.

```
NIM = num2str(NIM);   % convert to NIM array
```

NIM diubah menjadi array of strings agar bisa diekstrak karakter tertentu.

```
% Initialization / Known Information

% Declared E & M by extract it from NIM array
global M E I;
E = str2double(NIM(4:8));
M = str2double(NIM(5:8));
```

Ekstraksi digit ke-4 sampai 8 untuk Modulus lastisitas (E) kemudian diubah ke double.

Ekstraksi digit ke-5 sampai 8 untuk momen (M) kemudian diubah ke double.

```
% Contants
I = 1500;
L = 200; % mm
```

Input konstan I (inertia) dan L (panjang batang dalam mm).

```
% Boundary Condition
x0 = 0;
y0 = [0,0];
z0 = 0;
h = 20;  % step
```

Inisialisasi Boundary Condition

```
% Declare Functions
% ODE System for Beam Deflection
function dydx = beamODE(x, y)
    % Access global constants: Moment, Modulus, Inertia
    global M E I;

    % Extract slope (dy/dx) from the second element of y vector
    z = y(1, 2);                          % z = dy/dx

    % Compute derivatives
    dy1 = z;                              % dy/dx = z
    dy2 = M / (E * I) * (1 + z^2)^(1.5);  % d²y/dx² = M/(EI)*(1 + (dy/dx)^2)^(3/2)

    % Return derivative vector [dy/dx, d²y/dx²]
    dydx = [dy1, dy2];
end
```

Deklarasi fungsi beamODE

Untuk mendefinisikan sistem ODE dalam fungsi.

Lalu parameter y akan menghasilkan variable berikut:

dy1 adalah hasil turunan pertama

	dy2 adalah hasil turunan kedua yaitu persamaan beam deflection Bernoulli-Euler dy/dx = adalah hasil kedua derivative/turunan Sehingga dy/dx adalah output dari fungsi n4ODE

<pre>% Modified Euler Runge-Kutta 2nd Order
function [xs, ys] = ModifiedEulerRK2(x0, y0, L, h)
    % Initialize step count based on total length and step size
    n = floor(L / h);
    order = length(y0);   % Number of equations in the system (should be 2)
 	
    % Initialize arrays for x values and solution y values
    xs = zeros(n+1, 1);   % x values
    ys = zeros(n+1, order);   % y values (each row: [y, dy/dx])
 	
    % Set initial values
    xs(1) = x0;
    ys(1, :) = y0;
 	
    % Initialize variables for iteration
    i = 1;
    x = x0;
    y = y0;
 	
    % Begin stepping through the domain until x reaches L
    while x < L
        i = i + 1;   % Increment index
        h = min(h, L - x);   % Adjust final step size if overshooting
 	
        % Calculate slopes at beginning and end of the step
        k1 = beamODE(x, y);
        k2 = beamODE(x + h, y + k1 * h);
 	
        % Update values using Modified Euler method
        y = y + 0.5 * (k1 + k2) * h;
        x = x + h;
 	
        % Store current step results
        xs(i) = x;
        ys(i, :) = y;
    end
end</pre>
	Deklarasi fungsi yang melakukan algoritma Modified Euler Runge-Kutta 2nd Order.
	
	Output fungsi tersebut disimpan di [xs,ys] sebagai solusi x dan y dari hasil algoritma.

<pre>% Classical Runge-Kutta 4th Order
function [xs, ys] = ClassicalRK4(x0, y0, L, h)
    % Initialize step count and number of variables in the system
    n = floor(L / h);
    order = length(y0);
 	
    % Initialize arrays
    xs = zeros(n+1, 1);
    ys = zeros(n+1, order);
 	
    % Set initial values
    xs(1) = x0;
    ys(1, :) = y0;
 	
    % Initialize variables
    i = 1;
    x = x0;
    y = y0;
 	
    % Main iteration loop
    while x < L
        i = i + 1;   % Increment index
        h = min(h, L - x); % Adjust step size if close to boundary
 	
        % Compute intermediate slopes
        k1 = beamODE(x, y);
        k2 = beamODE(x + h/2, y + k1 * h/2);
        k3 = beamODE(x + h/2, y + k2 * h/2);
        k4 = beamODE(x + h,   y + k3 * h);
 	
        % Update the solution using weighted average of slopes
        y = y + (k1 + 2*k2 + 2*k3 + k4) * h / 6;
        x = x + h;
 	
        % Store results
        xs(i) = x;
        ys(i, :) = y;
    end
end</pre>
	Deklarasi fungsi yang melakukan algoritma Classical Runge-Kutta 4[th] Order.
	
	Output fungsi tersebut disimpan di [xs,ys] sebagai solusi x dan y dari hasil algoritma.

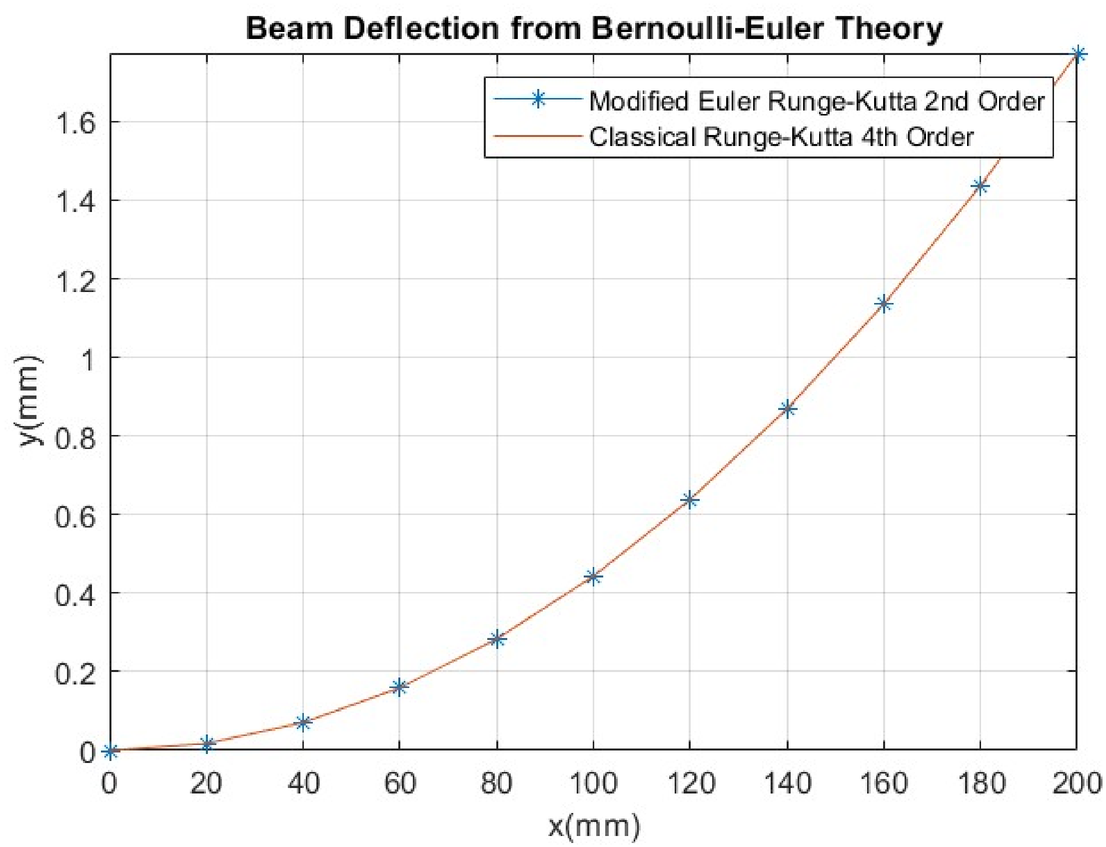
<pre>% Call Function
[x2,y2]=ModifiedEulerRK2(x0,y0,L,h);
[x4,y4]=ClassicalRK4(x0,y0,L,h);
 	
plot(x2,y2(:,1),'-*')
hold on
grid on
plot(x4,y4(:,1))
xlim([0 L])
ylim([0 max(y4(:,1))])
legend(["Modified Euler Runge-Kutta 2nd Order" "Classical Runge-Kutta 4th Order"])
xlabel("x(mm)")
ylabel("y(mm)")
title("Beam Deflection from Bernoulli-Euler Theory")</pre>
	Kemudian panggil fungsi ModifiedEulerRK2 dengan memberikan argument x0 & yo dari boundary conditions, konstan L dan h diketahui dari soal. Kemudian menyimpan hasil output di variable x2 & y2.
	
	Setelah itu, panggil fungsi ClassicalRK4

<table>
<tr>
<td>
<pre>
% Display results in table format
fprintf('\n%-10s %-20s %-20s %-20s %-20s\n', 'x (mm)', 'y_RK2 (mm)', 'dy/dx_RK2', 'y_RK4 (mm)', 'dy/dx_RK4');
fprintf(repmat('-', 1, 90)); fprintf('\n');
for i = 1:length(x2)
    fprintf('%-10.2f %-20.6f %-20.6f %-20.6f %-20.6f\n', ...
        x2(i), y2(i,1), y2(i,2), y4(i,1), y4(i,2));
end
</pre>
</td>
<td>dengan memberikan argument x0 & yo dari boundary conditions dan konstan L dan h diketahui dari soal. Kemudian menyimpan hasil output di variable x4 & y4.
Tampilkan hasil RK2 dan RK4 berupa variable x2,y2 dan x4,y4 dalam tabel di command window.</td>
</tr>
</table>

Hasil

Contoh dengan NIM : 13123069

```
Enter the SID/NIM number: 13123069
SID/NIM Number: 13123069
>>
```



Output RK2 dan RK4:

x (mm)	y_RK2 (mm)	dy/dx_RK2	y_RK4 (mm)	dy/dx_RK4
0.00	0.000000	0.000000	0.000000	0.000000
20.00	0.017738	0.001774	0.017738	0.001774
40.00	0.070953	0.003548	0.070953	0.003548
60.00	0.159644	0.005322	0.159644	0.005322
80.00	0.283813	0.007095	0.283813	0.007095
100.00	0.443461	0.008869	0.443461	0.008869
120.00	0.638589	0.010643	0.638589	0.010643
140.00	0.869199	0.012418	0.869200	0.012418
160.00	1.135294	0.014192	1.135295	0.014192
180.00	1.436876	0.015966	1.436877	0.015966
200.00	1.773947	0.017741	1.773949	0.017741