



## DAFTAR ISI

<b>DAFTAR ISI .....</b>	<b>1</b>
<b>BAB 1 Perulangan dan Percabangan .....</b>	<b>5</b>
1.1. <i>The if Statement .....</i>	<i>5</i>
1.2. <i>For Loops .....</i>	<i>8</i>
1.3. <i>While Loops.....</i>	<i>8</i>
1.4. <i>While Loops dan For Loops.....</i>	<i>9</i>
<b>BAB 2 Import .....</b>	<b>11</b>
<b>BAB 3 Plot Grafik .....</b>	<b>13</b>
3.1. <i>Menambahkan Judul dan Label Sumbu .....</i>	<i>13</i>
3.2. <i>Plot Banyak Grafik.....</i>	<i>14</i>
3.3. <i>Menentukan Gaya Garis dan Warna pada Grafik.....</i>	<i>16</i>
<b>BAB 4 Operasi Matriks.....</b>	<b>21</b>
4.1. <i>Membentuk Vektor dengan Matriks .....</i>	<i>21</i>
4.1.1. <i>Menuliskan secara langsung.....</i>	<i>21</i>
4.1.2. <i>Menggunakan Colon Operator.....</i>	<i>22</i>
4.1.3. <i>Menggunakan Linspace .....</i>	<i>23</i>
4.2. <i>Membuat Vektor dengan Vektor .....</i>	<i>23</i>
4.3. <i>Membentuk Matriks Khusus dengan Syntax .....</i>	<i>24</i>
4.3.1. <i>Matriks Zeros .....</i>	<i>24</i>



4.3.2.	Matriks Ones.....	25
4.3.3.	Matriks Eye .....	25
4.3.4.	Matiks Digital.....	25
4.3.5.	Matriks Magic.....	26
4.3.6.	Matriks Random.....	27
4.4.	<i>Menentukan Ukuran Matriks</i> .....	27
4.5.	<i>Modifikasi Elemen-Elemen Matriks</i> .....	30
4.5.1.	Ekstraksi Matriks.....	30
4.5.2.	Membalik Elemen pada Matriks .....	31
4.5.3.	Membalik Elemen pada Matriks .....	32
4.5.4.	Menghapus Elemen pada Matriks.....	33
4.5.5.	Menambahkan Elemen pada Matriks.....	34
4.5.6.	Mengganti Elemen pada Matriks.....	34
4.5.7.	Mengubah Dimensi Matriks.....	35
4.6.	<i>Transpose Matriks</i> .....	35
4.7.	<i>Inverse Matriks</i> .....	36
4.8.	<i>Determinan</i> .....	37
4.9.	<i>Rank</i> .....	38
4.10.	<i>Operasi Aritmatika</i> .....	38
4.10.1.	Penjumlahan Matriks .....	39
4.10.2.	Pengurangan Matriks.....	39



4.10.3. Perkalian Matriks .....	40
4.10.4. Pembagian Matriks .....	40
4.11. Operasi Vektor.....	41
4.11.1. Dot Product.....	41
4.11.2. Cross Product .....	41
4.12. Eigenvalue dan Eigenvector.....	42
4.13. Penyelesaian Sistem Persamaan Linier .....	42
4.13.1. Mendefinisikan Sistem Persamaan Linier .....	42
4.13.2. Menyelesaikan Sistem Persamaan Linier .....	43
<b>BAB 5 Persamaan Diferensial (PD).....</b>	<b>45</b>
5.1. Solusi Analitis Persamaan Diferensial (PD).....	45
5.1.1. PD orde 1 .....	45
5.1.2. PD orde 2 dengan kondisi awal .....	46
5.1.3. Beberapa contoh PD.....	46
5.2. Solusi Numerik Persamaan Diferensial (PD).....	47
5.2.1. Metode euler .....	47
5.2.2. Metode runge-kutta orde 4.....	48
5.3. ODE23 and ODE45 Solver Matlab .....	49
5.3.1. ODE23 .....	50
5.3.2. ODE45 .....	50
5.4. Tabel Perbandingan Solusi Numerik .....	51



<b>BAB 6 Transformasi.....</b>	<b>52</b>
6.1. <i>Transformasi Fourier Kontinu .....</i>	<i>52</i>
6.2. <i>Transformasi Fourier Balik (Inverse) .....</i>	<i>57</i>
6.3. <i>Transformasi Laplace .....</i>	<i>58</i>
6.4. <i>Transformasi Laplace Balik (Inverse) .....</i>	<i>62</i>



## BAB 1 Perulangan dan Percabangan

Dalam bahasa pemrograman tentunya sudah tidak asing dengan istilah looping atau perulangan dan percabangan. Sama halnya seperti bahasa pemrograman yang lain, pada Matlab juga mengenal istilah tersebut dan berikut cara penggunaan masing-masingnya.

### 1.1. The if Statement

If statement digunakan untuk mengeksekusi **commands** berdasarkan **condition** yang diberikan terpenuhi atau tidak. Bentuk if statement:

```
if condition
    commands
end
```

*Condition* adalah boolean (true/false) dan *commands* adalah set MATLAB commands. Jika *condition* bernilai true, maka *commands* di dalam statement (sebelum *end*) akan dieksekusi; jika *condition* bernilai false, maka *commands* di dalam statement tidak akan dieksekusi.

#### Contoh 1 : Penggunaan dasar

Perhatikan if statement berikut:

```
x = 2;          % Input nilai x (ubah-ubah nilai ini)
if x >= 0       % Set kondisi
    val = x     % Mendefinisikan variabel val sama dengan x dan
men-display
end
```

```
val = 2
```

Saat program dijalankan, jika *x* lebih besar atau sama dengan 0, maka variabel *val* akan sama dengan *x*; jika *x* negatif, maka tidak ada hasil apapun. Coba dengan mengganti nilai *x*

#### Contoh 2 : else command

Untuk menjalankan *commands* jika if statement dalam kondisi false, dapat menggunakan else. Script berikut akan menjalankan *2nd set-of-commands* jika if dalam kondisi false.



```
if condition
    1st set-of-commands
else
    2nd set-of-commands
end
```

Contoh program berikut

```
x = -10;      % Input nilai x (ubah-ubah nilai ini)
if x >= 0
    y = x;
else
    y = -x;
end
y
```

y = 10

Jika  $x \geq 0$  maka  $y = x$ ; jika kondisi tersebut tidak dipenuhi maka  $y = -x$ .

### Contoh 3 : elseif command

elseif command digunakan untuk menjalankan beberapa command berdasarkan lebih dari satu kondisi.

```
if 1st condition
    1st set-of-commands
elseif 2nd condition
    2nd set-of-commands
....
elseif nth condition
    nth set-of-commands
else
    final set-of-commands
```



end

Ketika program dijalankan, saat if 1st condition bernilai true maka 1st set of commands akan dieksekusi; jika if 1st condition bernilai false dan hanya elseif 2nd condition yang bernilai true maka 2nd set of commands akan dieksekusi; begitupun selanjutnya. Contohnya dapat menggunakan fungsi piecewise berikut

$$y = f(x) = \begin{cases} -x, & \text{if } x < 0 \\ \pi, & \text{if } x = 0 \\ x, & \text{if } 0 < x \end{cases}$$

```
x = 2;          % Input nilai x (ubah-ubah nilai ini)
if x < 0
    y = -x;
elseif x > 0
    y = x;
else
    y = pi;
end
y
```

y = 10

Saat nilai x positif atau negatif maka akan mengembalikan nilai y yaitu mutlak dari x, saat kedua kondisi tersebut tidak dipenuhi maka y akan mengembalikan nilai pi

#### Contoh 4 : Operator logika

Kita bisa menggunakan operasi logika dengan **if statement**, contoh script berikut akan melakukan pengecekan apakah x bernilai positif atau negatif, jika memenuhi maka y akan sama dengan x, namun jika kondisi tersebut tidak terpenuhi maka y akan sama dengan pi

```
x = 0;
if (x < 0) || (x > 0) % Cek apakah x positif atau negatif
    y = x
else
    y = pi
end
```



$$y = 3.1416$$

Selain OR (||) kita juga menggunakan operasi logika lain seperti AND (&&), NOT (~)

## 1.2. For Loops

Script for loop yang bisa digunakan di MATLAB:

```
for index = index_vector  
    commands  
end
```

- command for menandakan dimulainya loop
- command end menandakan loop berakhir
- **commands** akan dieksekusi berulang sesuai dengan banyaknya loop yang didefinisikan
- index vektor dalam bentuk vektor merupakan definisi banyaknya loop
- index adalah variabel yang menyimpan banyaknya loop index vector

### Contoh: Menjumlahkan elemen-elemen vektor

Pada script di bawah akan dilakukan perhitungan penjumlahan semua elemen vektor [1 2 3 4 5 6]

```
jumlah = 0; % Mendefinisikan variabel nilai awal  
for i = 1:6 % i adalah vektor dari 1 sampai 6 dengan increment 1, i  
    dapat diubah i = [1 2 3 4 5 6]  
    jumlah = jumlah + i % perhitungan  
end % penanda looping berakhir
```

jumlah = 1

jumlah = 3

jumlah = 6

jumlah = 10

jumlah = 15

jumlah = 21

## 1.3. While Loops

Script while loop yang bisa digunakan di MATLAB:





```
while condition  
    commands  
end
```

Pada while loop, commands akan terus dijalankan saat condition bernilai true. Loop akan berakhir hanya jika condition bernilai false. Sehingga dalam while loop kita tidak perlu mendefinisikan banyaknya iterasi terlebih dahulu.

### Contoh : Penjumlahan

while loop berikut akan menghitung dari 0 sampai 4

```
n = 0; % Mendefinisikan variabel nilai awal  
while n < 4 % Set kondisi  
    n = n + 1 % Assign a new value to "n"  
end % Exit the "while" loop
```

n = 1

n = 2

n = 3

n = 4

Penjumlahan n+1 akan terus berjalan saat n kurang dari 4, saat n lebih dari atau sama dengan 4 maka looping akan berakhir.

## 1.4. While Loops dan For Loops

Loops berikut akan menjumlahkan semua elemen vektor:

Menggunakan for loop:

```
x = 1:10; % Mendefinisikan array vektor dari 1 sampai 10  
sum = 0; % Nilai awal  
for i = 1:length(x) % i dari 1 sampai 10  
    sum = sum + x(i) % menjumlahkan sum dengan x(i), x(i) mengakses  
    nilai x pada data ke i  
end
```

sum = 1

sum = 3



```
sum = 6  
  
sum = 10  
  
sum = 15  
  
sum = 21  
  
sum = 28  
  
sum = 36  
  
sum = 45  
  
sum = 55
```

Menggunakan while loop:

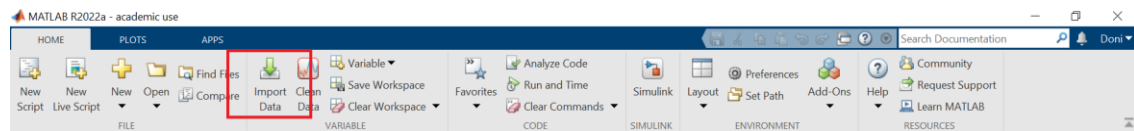
```
a = 1:10; % Mendefinisikan array vektor dari 1 sampai 10  
sum_a = 0; % Nilai awal  
k = 1; % Deklarasi k untuk menghitung  
while k <= length(a) % Set kondisi  
    sum_a = sum_a + a(k) % menjumlahkan sum_a dengan a(k), a(k) mengakses  
    nilai a pada data ke k  
    k = k + 1; % Increment k  
end
```

```
sum_a = 1  
  
sum_a = 3  
  
sum_a = 6  
  
sum_a = 10  
  
sum_a = 15  
  
sum_a = 21  
  
sum_a = 28  
  
sum_a = 36  
  
sum_a = 45  
  
sum_a = 55
```

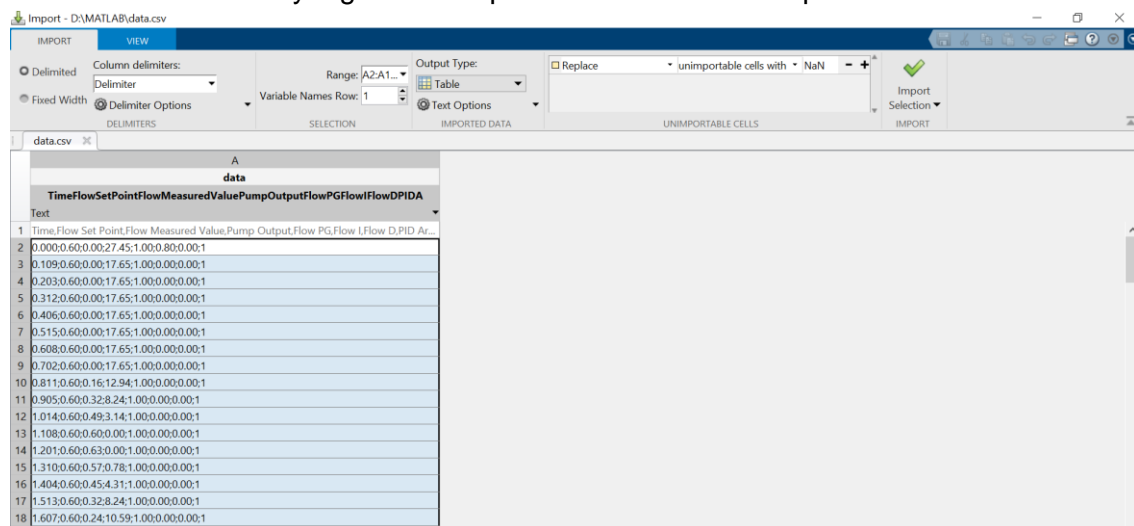
## BAB 2 Import

Matlab yang merupakan sebagai salah satu software pengolahan data terbaik tentunya harus dimanfaatkan dengan baik. Dan sebelum melakukan pengolahan dan analisis data maka data yang telah didapatkan sebelumnya haruslah di import terlebih dahulu, berikut beberapa cara untuk melakukan import data pada Matlab.

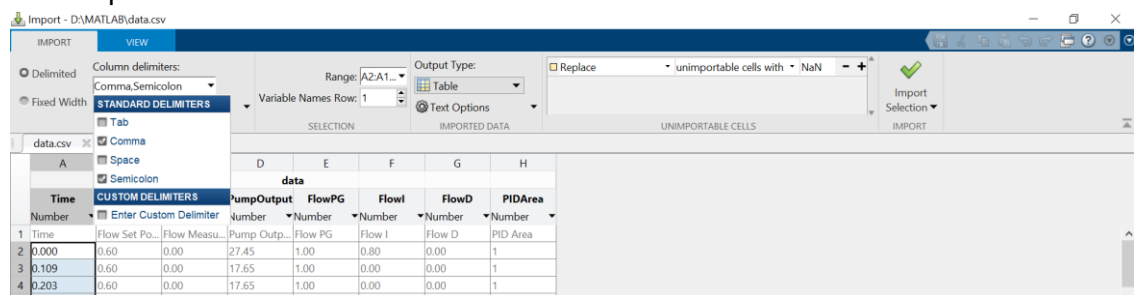
Cara pertama dapat dilakukan dengan cara menekan ikon import data pada bagian tab home



Setelah memilih data yang akan diimport akan muncul tab seperti berikut



Untuk memperbaiki data yang masih berantakan dapat dilakukan dengan memilih pada Column delimiters, column delimiters merupakan tanda pemisah antara satu data dengan data lainnya dan dapat dipilih berdasarkan tanda pemisah yang ada pada pilihan atau dapat di custom



Lalu untuk memberi nama pada setiap variabel data dapat dipilih pada Variable Names Row (artinya nama tiap variabel ada pada baris ke-) jika pada data sudah tertulis nama tiap variabelnya



Time	FlowSetPoint	FlowMeas...	PumpOutput	FlowPG	FlowI	FlowD	PIDArea
0.000	0.60	0.00	27.45	1.00	0.80	0.00	1
0.109	0.60	0.00	17.65	1.00	0.00	0.00	1
0.203	0.60	0.00	17.65	1.00	0.00	0.00	1
0.312	0.60	0.00	17.65	1.00	0.00	0.00	1

Untuk jenis data yang akan diimport nantinya dapat dipilih pada Output Type

Time	FlowSetPoint	FlowMeas...	PumpOutput	FlowPG	FlowI	FlowD	PIDArea
0.000	0.60	0.00	27.45	1.00	0.80	0.00	1
0.109	0.60	0.00	17.65	1.00	0.00	0.00	1
0.203	0.60	0.00	17.65	1.00	0.00	0.00	1
0.312	0.60	0.00	17.65	1.00	0.00	0.00	1
0.406	0.60	0.00	17.65	1.00	0.00	0.00	1

Dan setelah memilih data yang ingin diimport maka selanjutnya import data tersebut

Time	FlowSetPoint	FlowMeas...	PumpOutput	FlowPG	FlowI	FlowD	PIDArea
0.000	0.60	0.00	27.45	1.00	0.80	0.00	1

Data yang telah diimport dapat dilihat pada workspace

Name	Value
FlowMeasuredValue	153x1 double
Time	153x1 double

Cara kedua yaitu dengan menggunakan syntax yang telah disediakan oleh matlab, yaitu

```
A = importdata(filename)
```



## BAB 3 Plot Grafik

Plot Grafik merupakan salah satu fitur paling sering digunakan pada Matlab dikarenakan Matlab mempunyai kapabilitas yang kuat dalam segi pembuatan grafik dan cocok untuk digunakan para peneliti dan engineer. Ada beberapa perintah yang dapat digunakan untuk membuat plot grafik diantaranya

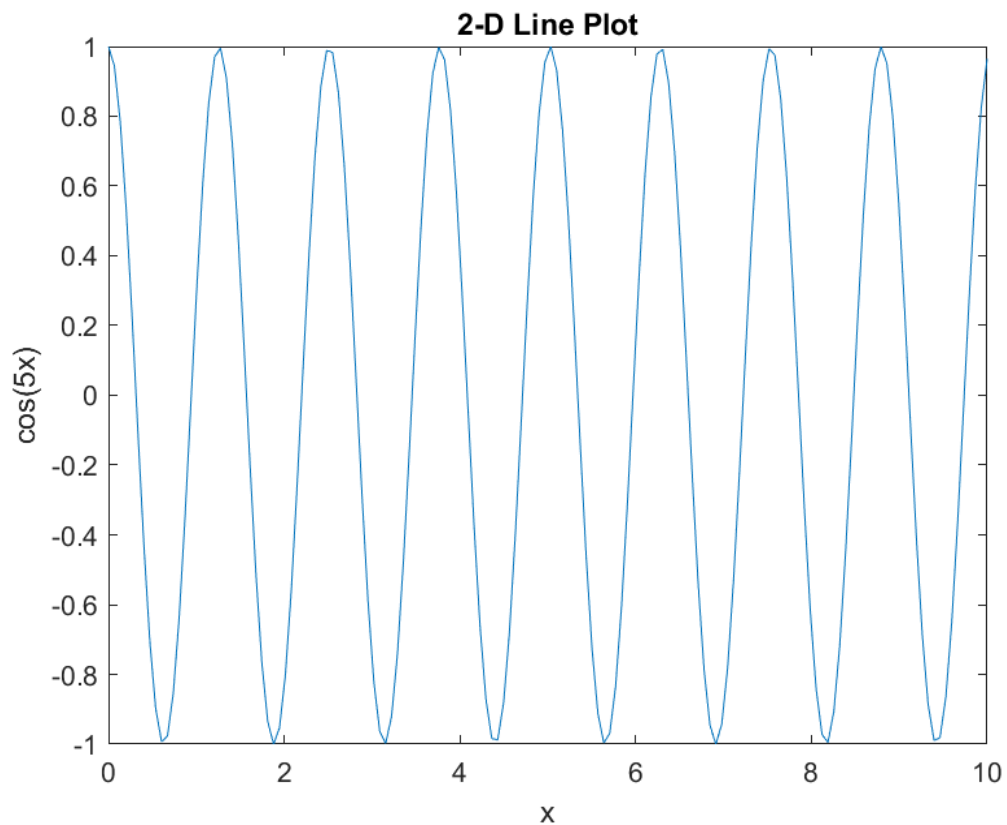
Kode	Fungsi
plot	Membuat plot
fplot	Membuat plot dengan fungsi $y = f(x)$
loglog	Membuat plot skala logaritma
semilogx	Membuat plot dengan sumbu x skala logaritma
semilogy	Membuat plot dengan sumbu y skala logaritma
polarplot	Membuat plot dengan koordinat polar

Selanjutnya akan diberikan beberapa contoh penggunaan plot grafik yang biasa digunakan pada Matlab.

### 3.1. Menambahkan Judul dan Label Sumbu

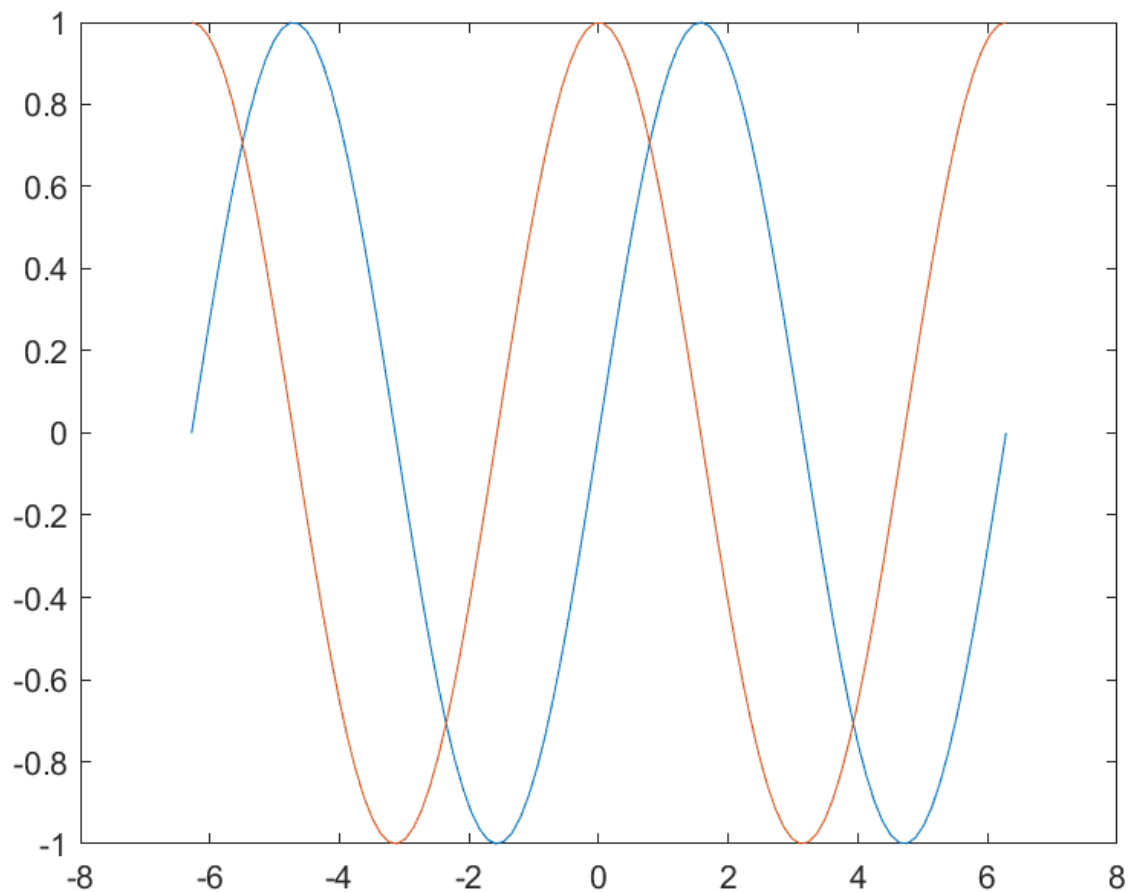
Penamaan judul dan pemberian label pada grafik juga merupakan hal yang tidak kalah penting dibanding lainnya, bahkan hal ini merupakan langkah awal yang seharusnya dilakukan dalam melakukan plot grafik

```
x = linspace(0,10,150);  
y = cos(5*x);  
plot(x,y)  
%% Untuk memberikan judul pada grafik  
%% tittle ('{nama judul}')  
title('2-D Line Plot')  
%% Untuk memberikan label pada masing masing sumbu  
%% xlabel('{nama label sumbu x}') | ylabel('{nama label sumbu y}')  
xlabel('x')  
ylabel('cos(5x)')
```



### 3.2. Plot Banyak Grafik

```
x = linspace(-2*pi,2*pi);  
y1 = sin(x);  
y2 = cos(x);  
figure  
plot(x,y1,x,y2) %% Untuk plot beberapa grafik dalam 1 gambar
```



figure

%% Untuk membagi beberapa grafik dalam beberapa gambar

%% subplot (jumlah baris yang diinginkan, jumlah kolom yang diinginkan, letak grafik)

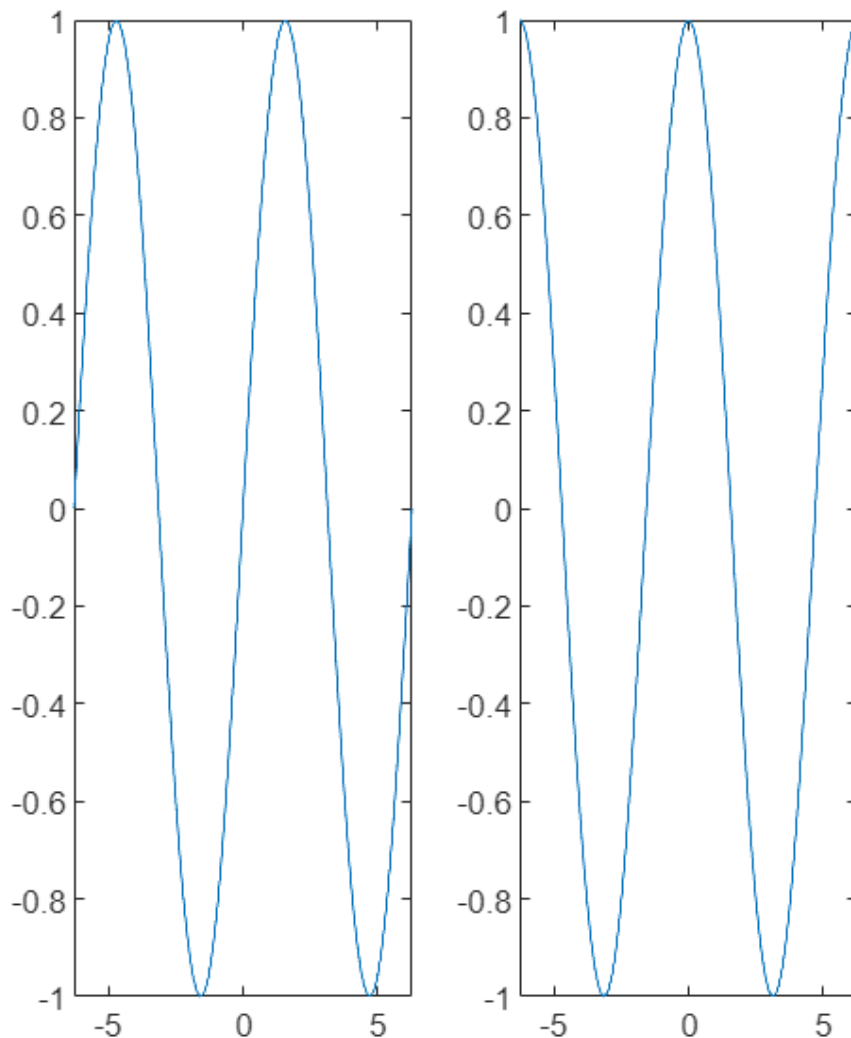
subplot(1,3,1)

plot(x,y1)

subplot(1,3,2)

plot(x,y2)



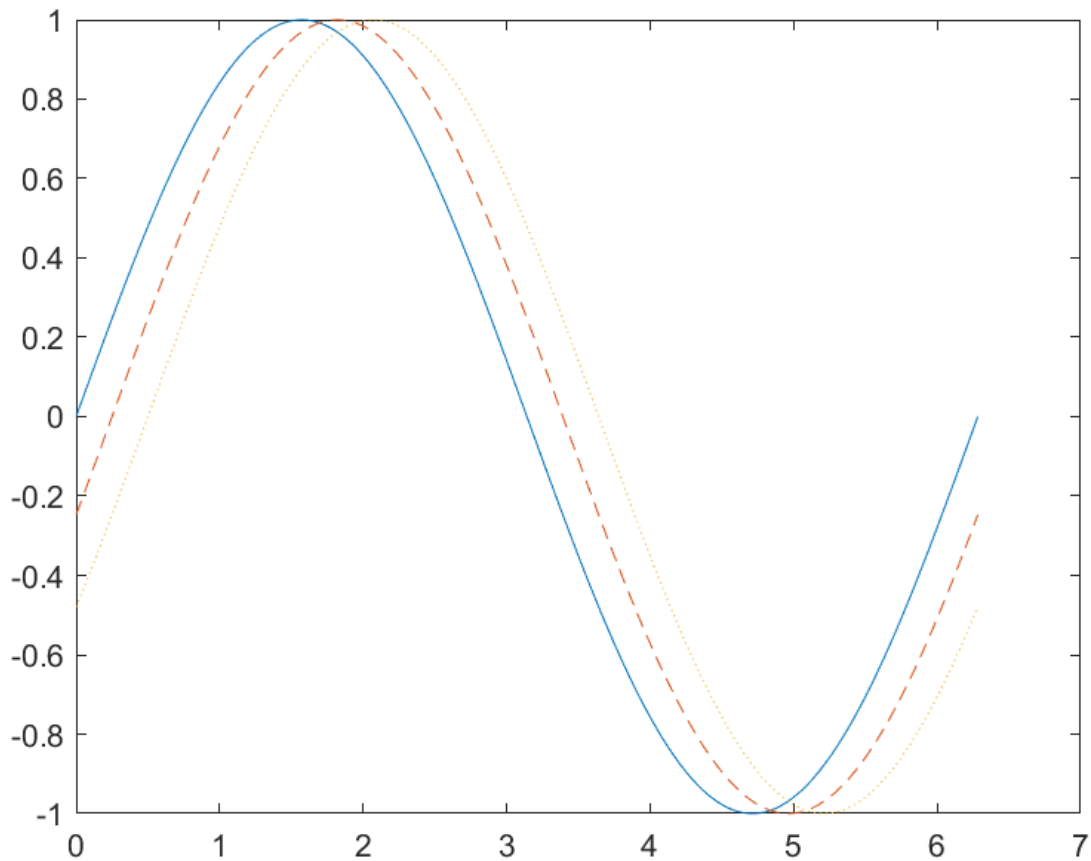


### 3.3. Menentukan Gaya Garis dan Warna pada Grafik

```
x = 0:pi/100:2*pi;  
y1 = sin(x);  
y2 = sin(x-0.25);  
y3 = sin(x-0.5);  
figure  
%% Untuk membuat gaya berbeda pada garis  
%% plot (x,y,'{kode gaya garis}')
```

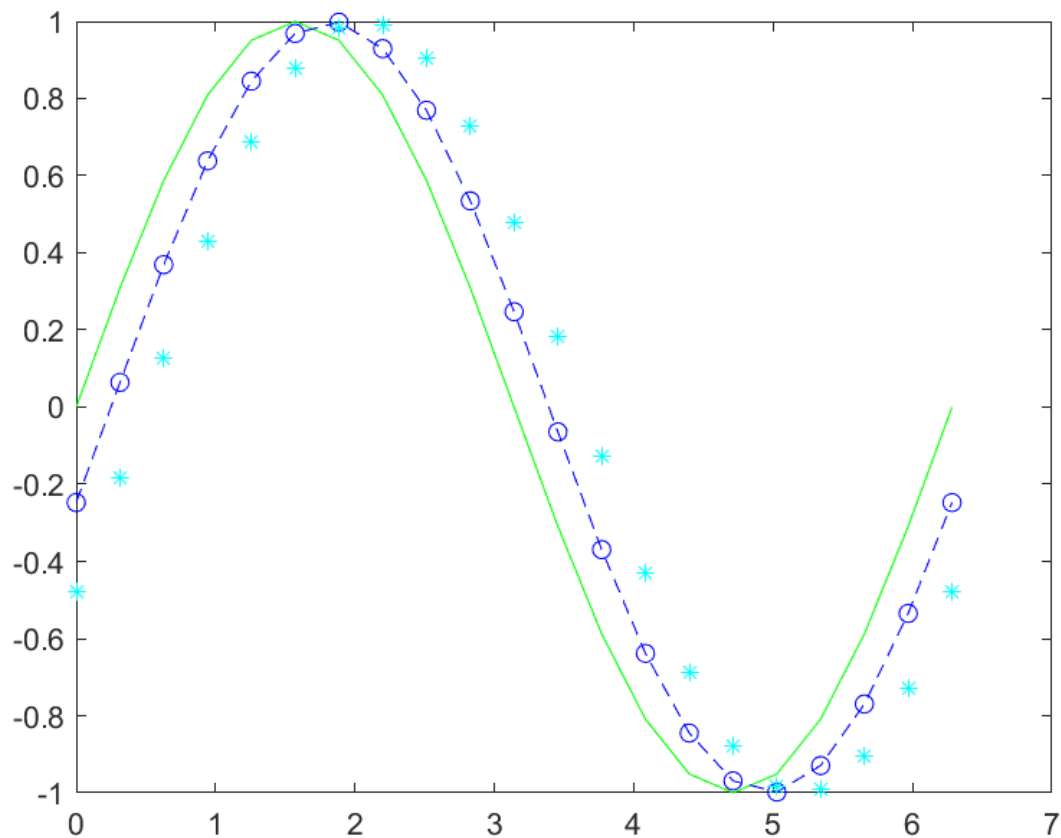
```
plot(x,y1,x,y2, '--',x,y3, ':')
```





```
x = 0:pi/10:2*pi;  
y1 = sin(x);  
y2 = sin(x-0.25);  
y3 = sin(x-0.5);  
  
figure  
%% Untuk memberikan warna pada tiap grafik  
%% plot (x,y,'{kode warna}{kode gaya garis}')
```

```
plot(x,y1, 'g', x,y2, 'b--o', x,y3, 'c*')
```



Untuk list style garis dan penanda yang dapat digunakan adalah sebagai berikut dan untuk warna yang dapat digunakan dapat dikombinasikan sedemikian rupa sehingga menghasilkan beragam warna. Untuk lebih lengkapnya dapat dilihat pada link berikut <https://www.mathworks.com/help/matlab/ref/plot.html>

Line Style	Description	Resulting Line
'—'	Solid line	————
'--'	Dashed line	- - - - -
'.'	Dotted line	.....
'-.'	Dash-dotted line	- . - . - .



# Laboratorium Kontrol dan Otomasi

Departemen Teknik Elektro ITS  
Fakultas Teknologi Elektro dan Informatika Cerdas  
Institut Teknologi Sepuluh Nopember








Marker	Description	Resulting Marker
'o'	Circle	○
'+'	Plus sign	+
'*'	Asterisk	*
'.'	Point	•
'x'	Cross	×
'_'	Horizontal line	—
' '	Vertical line	
's'	Square	□
'd'	Diamond	◇
'^'	Upward-pointing triangle	△
'v'	Downward-pointing triangle	▽
'>'	Right-pointing triangle	▷

## Laboratorium Kontrol dan Otomasi

Ruang AJ104 & B105, Departemen Teknik Elektro ITS  
Keputih, Sukolilo, Surabaya



'<'	Left-pointing triangle	◁
'p'	Pentagram	☆
'h'	Hexagram	⬠

Colour Name	Short Name	RGB Triplet	Appearance
'red'	'r'	[1 0 0]	
'green'	'g'	[0 1 0]	
'blue'	'b'	[0 0 1]	
'cyan'	'c'	[0 1 1]	
'yellow'	'y'	[1 1 0]	
'black'	'k'	[0 0 0]	
'white'	'w'	[1 1 1]	



## BAB 4 Operasi Matriks

Pada bagian ini akan dibahas mengenai pengoperasian matriks pada MATLAB, mulai dari membentuk matriks hingga operasi aritmatika matriks pada MATLAB.

### 4.1. Membentuk Vektor dengan Matriks

#### 4.1.1. Menuliskan secara langsung

Membentuk Matriks  $1 \times 1$ ,

```
A = [3]
```

A = 3

Vektor baris, Matriks  $1 \times n$ ,

```
A = [1 2 3]
```

A =  $1 \times 3$

1      2      3

Vektor kolom, Matriks  $n \times 1$ ,

```
A = [1;2;3]
```

A =  $3 \times 1$

1

2

3

```
A = [1
```

2

3]

A =  $3 \times 1$

1

2

3

```
A = [1 2 3]'
```



$$A = 3 \times 1$$

1

2

3

Matriks  $n \times n$ ,

$$A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$$

$$A = 3 \times 3$$

1      2      3

4      5      6

7      8      9

$$A = [1 \ 2 \ 3 \\ 4 \ 5 \ 6 \\ 7 \ 8 \ 9]$$

$$A = 3 \times 3$$

1      2      3

4      5      6

7      8      9

## 4.1.2. Menggunakan Colon Operator

Opsi lain untuk membentuk suatu vektor dapat menggunakan colon operator ":".

Membuat vektor baris dimulai dari 1 dan diakhiri 10, dengan increment 1, kita bisa menuliskan dengan script berikut:

```
1:10 % generate vector baris dari 1 sampai 10
```

$$A = 1 \times 10$$

1      2      3      4      5      6      7      8      9      10

Membuat vektor baris dengan increment angka desimal, contoh 0.1

```
1.5:0.1:2 % increment 0.1
```

$$A = 1 \times 6$$



1.5000      1.6000      1.7000      1.8000      1.9000      2.0000

Selain increment, kita bisa membuat decrement. Membuat vektor baris dari 2 ke 1.5 dengan decrement -0.1

```
2:-0.1:1.5 % decrement negative
```

A = 1×6

2.0000      1.9000      1.8000      1.7000      1.6000      1.5000

Jika kita membuat angka terakhir tidak bisa dicapai dengan increment, maka perhitungan MATLAB tidak akan berakhir di angka terakhir tersebut

```
2.7:0.1:pi % berhenti di 3.1, bukan 3.1416...
```

A = 1×5

2.7000      2.8000      2.9000      3.0000      3.1000

Colon operator akan berguna saat kita membutuhkan spacing yang spesifik antara elemen vektor yang berurutan

### 4.1.3. Menggunakan Linspace

linspace dapat digunakan untuk membuat vektor baris yang terpisah merata secara linear. linspace(x1,x2,n) berarti vektor baris sebanyak n elemen yang terpisah merata secara linear antara x1 dan x2. Jarak antara dua titik sebesar  $(x2-x1)/(n-1)$ .

Membuat vektor baris yang terpisah secara merata sebanyak 15 elemen dengan elemen pertama adalah 1 dan elemen terakhir adalah 10

```
linspace(1,10,15) % vektor baris yang terpisah secara merata sebanyak 15
```

Membuat vektor baris yang terpisah secara merata sebanyak 10 elemen dengan elemen pertama adalah 0 dan elemen terakhir adalah  $2\pi$

```
linspace(0,2*pi,10) % vektor baris yang terpisah secara merata sebanyak 10
```

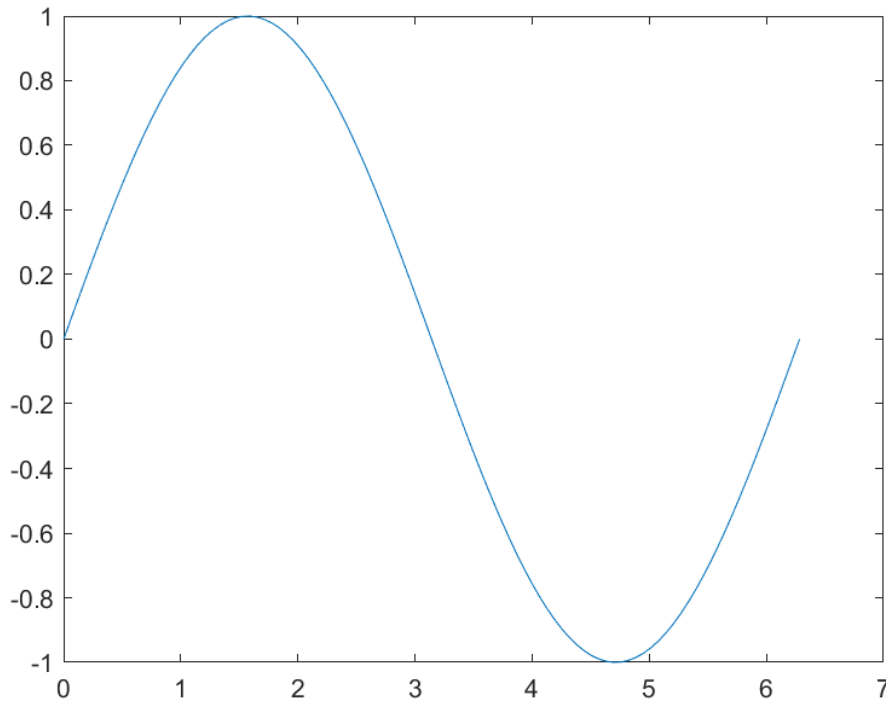
linspace akan berguna jika ingin mendefinisikan vektor dengan elemen bilangan irasional

### 4.2. Membuat Vektor dengan Vektor

Membuat vektor menggunakan vektor. Contoh, kita dapat membuat vektor nilai sinus dari  $x = 0$  ke  $x = 2\pi$ , menggunakan script berikut:



```
X = linspace(0,2*pi,100);  
Y = sin(X); % Y adalah vektor  
plot(X,Y)
```



Y adalah vektor dengan ukuran yang sama dengan X.

### 4.3. Membentuk Matriks Khusus dengan *Syntax*

Selanjutnya akan dipelajari untuk pembentukan matriks-matriks khusus yang memiliki *syntax* tersendiri pada MATLAB.

#### 4.3.1. Matriks Zeros

Matriks *zeros* atau umumnya disebut juga dengan matriks nol, dimana semua elemen matriksnya bernilai nol (0).

```
n = 3 %deklarasi variabel untuk ukuran matriks di code selanjutnya
```

```
n = 3
```

```
z = zeros(n) %membentuk matriks zeros n x n
```

```
z = 3x3
```

```
0    0    0
```





0	0	0
0	0	0

## 4.3.2. Matriks Ones

*Syntax ones digunakan untuk membentuk matriks satuan, dimana semua elemen matriksnya bernilai satu (1).*

```
o = ones(n)
```

*o = 3x3*

1	1	1
1	1	1
1	1	1

## 4.3.3. Matriks Eye

*Matriks Eye umumnya lebih dikenal dengan sebutan matriks identitas.*

```
i = eye(n)
```

*i = 3x3*

1	0	0
0	1	0
0	0	1

## 4.3.4. Matiks Digital

Dapat dilakukan pembuatan matriks yang hanya terisi diagonalnya saja pada MATLAB dengan menggunakan *syntax diag*.

```
d = diag([1 2 3])
```

*d = 3x3*

1	0	0
0	2	0
0	0	3

*Syntax diag juga dapat digunakan untuk mendapatkan nilai diagonal utama suatu matriks*



```
A %memanggil matriks A
```

A = 3×3

1	2	3
4	5	6
7	8	9

```
d = diag(A)
```

d = 3×1

1
5
9

Penggunaan *syntax diag* dua kali dapat membentuk matriks diagonal yang terdiri dari nilai diagonal utama dari matriks aslinya

```
d = diag(diag(A))
```

d = 3×3

1	0	0
0	5	0
0	0	9

#### 4.3.5. Matriks Magic

Matriks *magic* merupakan matriks acak yang elemen penyusunnya adalah bilangan bulat dari 1 hingga  $n^2$  dengan jumlah baris dan kolom yang sama.

```
m = magic(n)
```

m = 3×3

8	1	6
3	5	7
4	9	2



```
n = 4
```

```
n = 4
```

```
m = magic(n)
```

```
m = 4x4
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

## 4.3.6. Matriks Random

Melalui MATLAB dapat dibentuk matriks dengan elemen penyusunnya dipilih secara acak,  $n$  adalah dimensi dari matriks dan elemen penyusun matriksnya adalah bilangan antara 0 sampai 1.

```
n = 3 %deklarasi variabel untuk ukuran matriks di code selanjutnya
```

```
n = 3
```

```
r = rand(n)
```

```
r = 3x3
```

0.9001	0.7803	0.4039
0.3692	0.3897	0.0965
0.1112	0.2417	0.1320

```
r = randn(n) %bilangan acak yang terdistribusi normal
```

```
r = 3x3
```

2.5260	-1.2571	0.7914
1.6555	-0.8655	-1.3320
0.3075	-0.1765	-2.3299

## 4.4. Menentukan Ukuran Matriks

*Length*, menghitung panjang dimensi array terbesar pada suatu matriks



```
m %memanggil matriks m
```

```
m = 4x4
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

```
l = length(m)
```

```
l = 4
```

```
x = rand(3,7) %membentuk matriks random dengan ukuran 3 x 7
```

```
x = 3x7
```

0.1690	0.6477	0.2963	0.6868	0.6256	0.9294	0.4359
0.6491	0.4509	0.7447	0.1835	0.7802	0.7757	0.4468
0.7317	0.5470	0.1890	0.3685	0.0811	0.4868	0.3063

```
l = length(x)
```

```
l = 7
```

```
w = rand(8,4) %membentuk matriks random ukuran 8,4
```

```
w = 8x4
```

0.4909	0.2417	0.2963	0.6868
0.4893	0.4039	0.7447	0.1835
0.3377	0.0965	0.1890	0.3685
0.9001	0.1320	0.2963	0.6868
0.3692	0.9421	0.7447	0.1835
0.1112	0.5470	0.1890	0.3685
0.7803	0.6477	0.2963	0.6868
0.6491	0.4509	0.7447	0.1835

```
l = length(w)
```



1 = 8

Dari dua contoh diketahui bahwa syntax length mengambil nilai terbesar antara kolom atau baris suatu matriks. Bila yang diukur adalah sebuah vektor, maka nilai length sama dengan jumlah elemen yang ada pada vektor tersebut. Bila mengukur himpunan kosong, nilai length sama dengan nol.

**Size**, mengukur dimensi matriks dalam bentuk vektor baris ([A,B]), artinya matriks yang diukur memiliki ukuran A x B.

```
sz = size(m)
```

m = 1x2

4 4

```
sz = size(x)
```

m = 1x2

3 7

max dan min, digunakan untuk menentukan elemen terbesar dan terkecil pada matriks

x %memanggil matriks x

x = 3x7

0.1690	0.6477	0.2963	0.6868	0.6256	0.9294	0.4359
0.6491	0.4509	0.7447	0.1835	0.7802	0.7757	0.4468
0.7317	0.5470	0.1890	0.3685	0.0811	0.4868	0.3063

max(x) %mengambil nilai maksimal per kolom

ans = 1x7

0.7317	0.6477	0.7447	0.6868	0.7802	0.9294	0.4468
--------	--------	--------	--------	--------	--------	--------

min(x) %mengambil nilai minimal per kolom

ans = 1x7

0.1690	0.4509	0.1890	0.1835	0.0811	0.4868	0.3063
--------	--------	--------	--------	--------	--------	--------



## 4.5. Modifikasi Elemen-Elemen Matriks

Pada bagian ini akan dibahas mengenai cara melakukan ekstraksi elemen-elemen pada matriks, membalik elemen pada matriks, menghapus elemen pada matriks, menambahkan elemen pada matriks, mengganti elemen pada matriks, dan mengubah dimensi matriks.

### 4.5.1. Ekstraksi Matriks

MATLAB memungkinkan penggunaanya untuk mengambil elemen-elemen tertentu dari suatu matriks.

```
A %memanggil matriks A
```

```
A = 3x3
```

1	2	3
4	5	6
7	8	9

```
e = A(1,2) %A(baris,kolom)
```

```
e = 2
```

```
e = A(1,end) %A(baris, elemen pada kolom terakhir)
```

```
e = 3
```

```
e = A(end,1) %A(elemen pada baris terakhir, kolom)
```

```
e = 7
```

```
e = A(1,:) %ekstraksi semua elemen pada baris pertama
```

```
e = 1x3
```

1	2	3
---	---	---

```
e = A(:,1) %ekstraksi semua elemen pada kolom pertama
```

```
e = 3x1
```

1
---

4
---



7

```
e = A(1:2,3) %ekstraksi submatriks
```

e = 2×1

3

6

```
e = A(1:2,2:3) %ekstraksi submatriks
```

e = 2×2

2     3

5     6

## 4.5.2. Membalik Elemen pada Matriks

Selanjutnya, tools MATLAB juga memungkinkan untuk membalik elemen pada suatu matriks

```
A %memanggil matriks A
```

A = 3×3

1     2     3

4     5     6

7     8     9

```
A1 = A([2 1 3], :) %membalik baris 1 dan 2 dari matriks A
```

A1 = 3×3

4     5     6

1     2     3

7     8     9

```
A2 = A(:, [2 1 3]) %membalik kolom 1 dan 2 dari matriks A
```

A2 = 3×3

2     1     3



5	4	6
8	7	9

```
A3 = A([2 1 3], [2 1 3]) %membalik baris dan kolom 1 dan 2 dari matriks A
```

A3 = 3×3

5	4	6
2	1	3
8	7	9

### 4.5.3. Membalik Elemen pada Matriks

Selanjutnya, tools MATLAB juga memungkinkan untuk membalik elemen pada suatu matriks

```
A % memanggil matriks A
```

A = 3×3

1	2	3
4	5	6
7	8	9

```
A1 = A([2 1 3], :) % membalik baris 1 dan 2 dari matriks A
```

A1 = 3×3

4	5	6
1	2	3
7	8	9

```
A2 = A(:, [2 1 3]) % membalik kolom 1 dan 2 dari matriks A
```

A2 = 3×3

2	1	3
5	4	6





8 7 9

```
A3 = A([2 1 3], [2 1 3]) % membalik baris dan kolom 1 dan 2 dari matriks A
```

A3 = 3x3

5	4	6
2	1	3
8	7	9

## 4.5.4. Menghapus Elemen pada Matriks

MATLAB memungkinkan pengguna untuk menghapus baris dan kolom dari suatu matriks dengan menggunakan operator kosong

```
B = magic(5) %membuat matriks magic 5 x 5
```

B = 5x5

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

```
B(1,:) = [] %menghapus baris pertama
```

B = 4x5

23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

```
B(:,1) = [] %menghapus kolom pertama
```

B = 4x4

5	7	14	16
---	---	----	----



6	13	20	22
12	19	21	3
18	25	2	9

```
B([1 2], :) = [] %menghapus baris pertama dan kedua
```

B = 2x4

12	19	21	3
18	25	2	9

## 4.5.5. Menambahkan Elemen pada Matriks

Selain menghapus, MATLAB juga memiliki fungsi untuk menambahkan elemen pada matriks

```
B = [[6 13 20 22];B(1,:);B(2,:)] %menambahkan satu baris pada matriks B
```

B = 3x4

6	13	20	22
12	19	21	3
18	25	2	9

```
B = [[23 4 10]' B(:,1) B(:,2) B(:,3) B(:,4)] %menambahkan satu kolom pada matriks B
```

B = 3x5

23	6	13	20	22
4	12	19	21	3
10	18	25	2	9

## 4.5.6. Mengganti Elemen pada Matriks

Untuk mengganti elemen dapat dilakukan teknik berikut

```
A % memanggil matriks A
```

A = 3x3

1	2	3
---	---	---



4	5	6
7	8	9

`A(1,1) = 0` %mengganti elemen pada baris pertama kolom pertama dengan 0

A = 3x3

0	2	3
4	5	6
7	8	9

`A(1,:) = 1` % baris pertama dengan 1

A = 3x3

1	1	1
4	5	6
7	8	9

`A(1,:) = [1 2 4]` % mengganti baris pertama dengan 1, 2, dan 4

A = 3x3

1	2	4
4	5	6
7	8	9

## 4.5.7. Mengubah Dimensi Matriks

Untuk mengubah dimensi matriks, digunakan *syntax reshape*

`reshape(A,1,9)` % `reshape(A,n,m)` (m=kolom baru n=baris baru)

ans = 1x9

1	4	7	2	5	8	4	6	9
---	---	---	---	---	---	---	---	---

## 4.6. Transpose Matriks

Melakukan operasi transpose matriks pada MATLAB memungkinkan dengan menggunakan operator *quote* '



```
a = [1 2 3;4 5 6;7 8 9]
```

a = 3x3

1	2	3
4	5	6
7	8	9

```
a = a'
```

a = 3x3

1	4	7
2	5	8
3	6	9

## 4.7. Inverse Matriks

Melalui MATLAB, dapat menghitung invers matriks dengan lebih sederhana dan berlaku untuk semua ordo, sebagai berikut

```
X = [1 0 2;-1 5 0;0 3 -9]
```

X = 3x3

1	0	2
-1	5	0
0	3	-9

```
Y = inv(X)
```

Y = 3x3

0.8824	-0.1176	0.1961
0.1765	0.1765	0.0392
0.0588	0.0588	-0.0980

```
Z = X^(-1)
```

Z = 3x3

0.8824	-0.1176	0.1961
--------	---------	--------



0.1765	0.1765	0.0392
0.0588	0.0588	-0.0980

%pembuktian dengan hasil invers \* matriks original = matriks identitas

Y1 = Y\*X

Y1 = 3x3

1.0000	0.0000	-0.0000
0	1.0000	-0.0000
0	-0.0000	1.0000

Z1 = Z\*X

Z1 = 3x3

1.0000	0.0000	-0.0000
0	1.0000	-0.0000
0	-0.0000	1.0000

## 4.8. Determinan

Selain invers, MATLAB juga dapat menghitung determinan dari suatu matriks

X

X = 3x3

1	0	2
-1	5	0
0	3	-9

X1 = det(X)

X1 = -51

P = [1 1 2; 1 1 2; 10 8 15]

P = 3x3



1      1      2

1      1      2

10    8    15

$P1 = \det(P)$  % karena  $\det = 0$ , maka matriks  $P$  adalah matriks singular

$P1 = 0$

## 4.9. Rank

Rank adalah jumlah kolom yang *linearly independent* dalam sebuah matriks.

$X$

$X = 3 \times 3$

1      0      2

-1    5      0

0      3    -9

$k1 = \text{rank}(X)$

$k1 = 3$

$P = [1 \ 1 \ 2; 1 \ 1 \ 2; 10 \ 8 \ 15]$

$P = 3 \times 3$

1      1      2

1      1      2

10    8    15

$k2 = \text{rank}(P)$

$k2 = 2$

## 4.10. Operasi Aritmatika

Pada bagian ini akan dibahas mengenai empat jenis operasi, yaitu penjumlahan, pengurangan, perkalian, dan pembagian matriks



## 4.10.1. Penjumlahan Matriks

Sebelum melakukan penjumlahan matriks, ada syarat yang harus dipenuhi, yaitu dimensi antar matriks yang akan dijumlahkan harus sama

$$A = [1 \ 3 \ 4; \ 2 \ 7 \ 9; \ 5 \ 6 \ 8]$$

$$A = 3 \times 3$$

$$\begin{bmatrix} 1 & 3 & 4 \\ 2 & 7 & 9 \\ 5 & 6 & 8 \end{bmatrix}$$

$$B = [7 \ 3 \ 5; \ 6 \ 9 \ 1; \ 2 \ 4 \ 8]$$

$$B = 3 \times 3$$

$$\begin{bmatrix} 7 & 3 & 5 \\ 6 & 9 & 1 \\ 2 & 4 & 8 \end{bmatrix}$$

$$S = A+B$$

$$S = 3 \times 3$$

$$\begin{bmatrix} 8 & 6 & 9 \\ 8 & 16 & 10 \\ 7 & 10 & 16 \end{bmatrix}$$

## 4.10.2. Pengurangan Matriks

Operasi pengurangan matriks memiliki syarat yang sama dengan operasi penjumlahan

$$M = A-B$$

$$M = 3 \times 3$$

$$\begin{bmatrix} -6 & 0 & -1 \\ -4 & -2 & 8 \end{bmatrix}$$



-3      2      0

## 4.10.3. Perkalian Matriks

Operasi perkalian matriks memiliki syarat, yaitu jumlah kolom matriks pertama sama dengan jumlah baris matriks kedua

$$C = A * B$$

$$C = 3 \times 3$$

33	46	40
74	105	89
87	101	95

$$C1 = A * B \text{ \% perkalian per elemen}$$

$$C1 = 3 \times 3$$

7	9	20
12	63	9
10	24	64

## 4.10.4. Pembagian Matriks

Teknik pembagian dibagi menjadi dua yaitu *left division* dan *right division*. *Left division* digunakan untuk mendapatkan nilai X dari persoalan  $A * X = B$ , sedangkan *right division* digunakan untuk mendapatkan nilai X dari persoalan  $X * A = B$ . Syarat untuk melakukan operasi pembagian matriks adalah jumlah kolom matriks pertama sama dengan jumlah baris matriks kedua, serta determinan matriks kedua tidak sama dengan nol. Misalkan kita memiliki persoalan  $A * B = C$ ,

$$D1 = A \setminus C$$

$$D1 = 3 \times 3$$

7.0000	3.0000	5.0000
6.0000	9.0000	1.0000
2.0000	4.0000	8.0000

$$D2 = C / B$$

$$D2 = 3 \times 3$$





1.0000	3.0000	8.0000
2.0000	7.0000	9.0000
5.0000	6.0000	8.0000

D3 = A./B % pembagian per elemen

D1 = 3x3

0.14291.0000	0.8000	
0.3333	0.7778	9.0000
2.5000	1.5000	1.0000

## 4.11. Operasi Vektor

### 4.11.1. Dot Product

Terdapat *syntax dot*, yang dapat digunakan untuk memperoleh nilai *dot product* dari dua matriks

dotproduct = dot(A,B)

dotproduct = 1x3

29	96	93
----	----	----

% dotproduct(1) = dot product dari A(:,1) dan B(:,1)

### 4.11.2. Cross Product

Terdapat *syntax cross*, yang dapat digunakan untuk memperoleh nilai *cross product* dari dua matriks

crossproduct = cross(A,B)

crossproduct = 3x3

-26	-26	64
33	6	8
-8	8	-41

% crossproduct(1) = cross product dari A(:,1) dan B(:,1)



## 4.12. Eigenvalue dan Eigenvector

Melalui MATLAB juga bisa didapatkan nilai eigen dan vektor eigen dengan menggunakan *syntax eig*

```
N = magic(3)
```

N = 3x3

8	1	6
3	5	7
4	9	2

```
[V,D] = eig(N)
```

% eigenvalue dinyatakan dalam diagonal matriks D

% eigenvector dinyatakan dalam matriks V

V = 3x3

-0.5774	-0.8131	-0.3416
-0.5774	0.4714	-0.4714
-0.5774	0.3416	0.8131

D = 3x3

15.0000	0	0
0	4.8990	0
0	0	-4.8990

## 4.13. Penyelesaian Sistem Persamaan Linier

Pada bagian ini akan dibahas mengenai bagaimana mendefinisikan persamaan linier hingga bagaimana menemukan solusi dari suatu persamaan linier.

### 4.13.1. Mendefinisikan Sistem Persamaan Linier

Misalkan diketahui sebuah persamaan linier sebagai berikut

$$2x + y + z = 2$$

$$-x + y - z = 3$$

$$x + 2y + 3z = -10$$



Untuk mendefinisikan persamaan linier ke MATLAB dapat digunakan teknik sebagai berikut

```
syms x y z %membuat simbol variabel
```

```
p1 = 2*x + y + z == 2
```

p1 =

$$2x + y + z = 2$$

%pada MATLAB, simbol = adalah sama, sedangkan simbol == adalah sama dengan

```
p2 = -x + y - z == 3
```

p2 =

$$-x + y - z = 3$$

```
p3 = x + 2*y + 3*z == -10
```

p3 =

$$x + 2y + 3z = -10$$

## 4.13.2. Menyelesaikan Sistem Persamaan Linier

Selanjutnya, untuk menyelesaikan persoalan persamaan linier, terdapat dua teknik yang dapat digunakan

### 4.13.2.1. Linsolve

*Syntax linsolve* digunakan untuk menyelesaikan persamaan linier dalam bentuk matriks  $AX = B$ . Untuk mengonversi persamaan ke bentuk matriks, dapat menggunakan *syntax equationsToMatrix*.

```
[A,B] = equationsToMatrix([p1,p2,p3], [x,y,z])
```

A =

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

B =

$$\begin{bmatrix} 2 \\ 3 \\ -10 \end{bmatrix}$$



3

-10

```
X = linsolve(A,B)
```

X =

3

1

-5

Dari X, dapat diketahui nilai  $x = 3$ ,  $y = 1$ , dan  $z = -5$

## 4.13.2.2. Solve

*Syntax solve* digunakan untuk menyelesaikan persamaan linier tanpa perlu mengonversi bentuk persamaan menjadi matriks.

```
sol = solve([p1, p2, p3], [x, y, z]);
```

```
x = sol.x
```

x =

3

```
y = sol.y
```

y =

1

```
z = sol.z
```

z = -5



## BAB 5 Persamaan Diferensial (PD)

Persamaan diferensial adalah persamaan matematika yang menghubungkan suatu fungsi dengan turunannya. Persamaan diferensial biasanya digunakan untuk memodelkan fenomena yang melibatkan perubahan atau pertumbuhan, seperti laju pertumbuhan populasi, dinamika fluida, dan perambatan gelombang. Selain itu, persamaan diferensial juga digunakan untuk memodelkan sistem kontrol non-linear. Persamaan diferensial pada sistem kontrol non-linear biasanya lebih kompleks dan dapat melibatkan turunan orde yang lebih tinggi, sehingga memerlukan teknik-teknik analisis yang lebih canggih, seperti metode numerik atau simulasi komputer.

### 5.1. Solusi Analitis Persamaan Diferensial (PD)

Untuk mendapatkan solusi analitis persamaan diferensial di Matlab dapat menggunakan fungsi `dsolve` dengan atau tanpa kondisi awal. Sebagai contoh digunakan persamaan diferensial

#### 5.1.1. PD orde 1

$$\frac{dy}{dt} = -0.5y$$

Mendefinisikan fungsi y

```
syms y(t)
```

Mendefinisikan PD yang akan diselesaikan dengan fungsi diff dan persamaan dengan ==

```
ode = diff(y,t) == -0.5*y
```

ode(t) =

$$\frac{\partial}{\partial t} y(t) = -\frac{y(t)}{2}$$

Cari solusi PD tersebut dengan fungsi dsolve

```
ySol(t) = dsolve(ode)
```

ySol(t) =

$$C_1 e^{-\frac{t}{2}}$$

Pada penyelesaian persamaan diferensial tanpa kondisi awal masih terdapat nilai konstanta  $C_1$ , untuk mendapatkan nilai konstanta tersebut maka dibutuhkan nilai



kondisi awal  $y(0)$ . Dimisalkan  $y(0) = 0.5$ , maka dilakukan sedikit perubahan untuk fungsi dsolve sebagai berikut:

```
cond = y(0) == 0.5;
ySol(t) = dsolve(ode,cond)
```

$$ySol(t) = \frac{e^{-\frac{t}{2}}}{2}$$

## 5.1.2. PD orde 2 dengan kondisi awal

$$\frac{d^2y}{dt^2} = \sin(3t) + y$$

$$y(0) = 1$$

$$y'(0) = 0$$

Untuk kondisi awal  $y'(0)=0$  turunkan fungsi  $y$   $Dy=diff(y,t)$  lalu definisikan  $Dy(0)=0$ .

```
syms y(t)
Dy = diff(y,t);
ode = diff(y,t,2) == sin(3*t)+y;
cond1 = y(0) == 1;
cond2 = Dy(0) == 0;
```

Solusi PD untuk  $y$  dapat dicari dengan

```
conds = [cond1 cond2];
ySol(t) = dsolve(ode,conds)
```

$$ySol(t) =$$

$$\frac{7e^{-t}}{20} - \frac{\sin(3t)}{10} + \frac{13e^t}{20}$$

## 5.1.3. Beberapa contoh PD

PD orde 1

$$\frac{dy}{dx} = x + y$$

```
syms y(x)
```



$y(0) = 0$	<pre>ode = diff(y,x) == x+y; cond = y(0) == 0; ySol(x) = dsolve(ode,cond)</pre> $ySol(x) = e^x - x - 1$
PD orde 2 $\frac{d^2y}{dt^2} + 0.8\frac{dy}{dt} + 0.2y = 0$	<pre>syms y(t) ode = diff(y,t,2)+0.8*diff(y,t)+0.2*y == 0; ySol(t) = dsolve(ode)</pre> $ySol(t) = C_1 \cos\left(\frac{t}{5}\right) e^{-\frac{2t}{5}} - C_2 \sin\left(\frac{t}{5}\right) e^{-\frac{2t}{5}}$
PD non linear tanpa kondisi awal $\frac{dy}{dt} - \frac{1}{y} = 0$ Persamaan ini memiliki lebih dari satu penyelesaian	<pre>syms y(t) ode = diff(y,t)-1/y == 0; ySol(t) = dsolve(ode)</pre> $ySol(t) = \begin{pmatrix} \sqrt{2} \sqrt{C_1+t} \\ -\sqrt{2} \sqrt{C_1+t} \end{pmatrix}$

## 5.2. Solusi Numerik Persamaan Diferensial (PD)

Teknik numerik dapat digunakan untuk menyelesaikan persamaan diferensial ketika metode analitik sulit digunakan atau saat kita bekerja dengan data empirik.

### 5.2.1. Metode euler

Algoritma metode euler:

- Definisikan  $y' = f(x_i, y_i)$
- Inisialisasi  $x_0, y_0$
- Input batas akhir nilai  $x_{final}$
- Tentukan step  $h$  dan jumlah  $N = (x_{final} - x_0)/h$
- Untuk  $n = 0, 1, \dots, N$  hitung

$$x_{n+1} = x_n + h$$



$$y_{n+1} = y_n + hf(x_n, y_n)$$

- Output  $x_{n+1}, y_{n+1}$

Contoh untuk  $y' = x + y$  dengan kondisi awal  $y(0) = 0$  dan  $h = 0.2$

```

clear;clc
f = @(x,y)(x+y);    % Mendefinisikan persamaan diferensial
x(1) = 0;           % Inisialisasi x
y(1) = 0;           % Inisialisasi y
xn = 1;             % Batas akhir x
h = 0.2;            % Step size
N = (xn-x(1))/h;     % Banyaknya iterasi
for n=1:N
    y(n+1) = y(n) + h*f(x(n),y(n));
    x(n+1) = x(n) + h;
end

```

Tabel hasil perhitungan dan perbandingan metode euler dan analitik tiap iterasi

x	y (euler)	y (analitis)	error
0.000000	0.000000	0.000000	0.000000
0.200000	0.000000	0.021403	0.021403
0.400000	0.040000	0.091825	0.051825
0.600000	0.128000	0.222119	0.094119
0.800000	0.273600	0.425541	0.151941
1.000000	0.488320	0.718282	0.229962

$$\text{mean error} = 0.0915417$$

## 5.2.2. Metode runge-kutta orde 4

Algoritma metode runge-kutta orde 4:

- Definisikan  $y' = f(x_i, y_i)$
- Inisialisasi  $x_0, y_0$
- Tentukan step  $h$  dan jumlah  $N = (x_{final} - x_0)/h$
- Untuk  $n = 0, 1, \dots, N$  hitung

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)$$

$$k_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2)$$

$$k_4 = hf(x_n + h, y_n + k_3)$$





$$x_{n+1} = x_n + h$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

- Output  $x_{n+1}, y_{n+1}$

Contoh untuk  $y' = x + y$  dengan kondisi awal  $y(0) = 0$  dan  $h = 0.2$

```
clear;clc
f = @(x,y)(x+y);    % Mendefinisikan persamaan diferensial
x(1) = 0;           % Inisialisasi x
y(1) = 0;           % Inisialisasi y
xn = 1;             % Batas akhir x
h = 0.2;            % Step size
N = (xn-x(1))/h;     % Banyaknya iterasi
for n=1:N            %menghitung solusi dari x0 sampai xn dengan step size h
    %formula RK orde 4
    k1 = h*f(x(n),y(n));
    k2 = h*f(x(n)+0.5*h,y(n)+0.5*k1);
    k3 = h*f(x(n)+0.5*h,y(n)+0.5*k2);
    k4 = h*f(x(n)+h,y(n)+k3);
    y(n+1) = y(n) + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
    x(n+1) = x(n) + h;    %increment x dengan step size h
end
```

Tabel hasil perhitungan dan perbandingan metode RK4 dan analitik tiap iterasi

x	y (RK4)	y (analitis)	error
0.000000	0.000000	0.000000	0.000000
0.200000	0.021400	0.021403	0.000003
0.400000	0.091818	0.091825	0.000007
0.600000	0.222106	0.222119	0.000012
0.800000	0.425521	0.425541	0.000020
1.000000	0.718251	0.718282	0.000031

*mean error = 0.000012*

### 5.3. ODE23 and ODE45 Solver Matlab

ODE23 dan ODE45 adalah fitur ODE solver numerik yang ada di Matlab. Dalam metode ini akan diselesaikan Persamaan Diferensial orde pertama yaitu:



$$\frac{dy}{dx} = x + y$$

$$y(0) = 0$$

$$h = 0.2$$

## 5.3.1. ODE23

Fungsi yang digunakan dalam metode ini adalah  $[x,y] = \text{ode23}(\text{odefun}, \text{tspan}, y_0)$ . Dengan  $x$  adalah evaluation point yang dikembalikan dalam bentuk vektor array,  $y$  adalah solusi dalam bentuk vektor array,  $\text{odefun}$  adalah fungsi yang akan diselesaikan,  $\text{tspan}$  adalah interval  $x$  antara  $[x_0 \text{ } x_f]$  apabila ingin diberi step size  $h$  maka menjadi  $[x_0:h:x_f]$ , dan  $y_0$  adalah kondisi awal dari persamaan diferensial.

```
clear;clc
dy = @(x,y)(x+y);           % Persamaan Diferensial
x0 = 0;                     % Inisialisasi x / interval awal
xn = 1;                     % interval akhir
y0 = 0;                     % initial condition y(0)
h = 0.2; % step size
[x,y] = ode23(dy,[x0:h:xn],y0);
```

Tabel hasil perhitungan dan perbandingan ode23 dan analitik tiap iterasi

x	y (ode23)	y (analitis)	error
0.000000	0.000000	0.000000	0.000000
0.200000	0.021402	0.021403	0.000001
0.400000	0.091815	0.091825	0.000010
0.600000	0.222093	0.222119	0.000026
0.800000	0.425492	0.425541	0.000049
1.000000	0.718207	0.718282	0.000075

*mean error = 0.000027*

## 5.3.2. ODE45

Fungsi yang digunakan dalam metode ini adalah  $[x,y] = \text{ode45}(\text{odefun}, \text{tspan}, y_0)$ . Dengan  $x$  adalah evaluation point yang dikembalikan dalam bentuk vektor array,  $y$  adalah solusi dalam bentuk vektor array,  $\text{odefun}$  adalah fungsi yang akan diselesaikan,  $\text{tspan}$  adalah interval  $x$  antara  $[x_0 \text{ } x_f]$  apabila ingin diberi step size  $h$  maka menjadi  $[x_0:h:x_f]$ , dan  $y_0$  adalah kondisi awal dari persamaan diferensial.

```
clear;clc
dy = @(x,y)(x+y);           % Persamaan Diferensial
x0 = 0;                     % Inisialisasi x / interval awal
xn = 1;                     % interval akhir
```



```
y0 = 0; % initial condition y(0)
h = 0.2; % step size
[x,y] = ode45(dy,[x0:h:xn],y0);
```

Tabel hasil perhitungan dan perbandingan ode45 dan analitik tiap iterasi

x	y (ode45)	y (analitis)	error
0.000000	0.000000	0.000000	0.000000
0.200000	0.021403	0.021403	0.000000
0.400000	0.091825	0.091825	0.000000
0.600000	0.222119	0.222119	0.000000
0.800000	0.425541	0.425541	0.000000
1.000000	0.718282	0.718282	0.000000

mean error = 0

#### 5.4. Tabel Perbandingan Solusi Numerik

Metode	Mean error
Euler	0.0915417
Runge-Kutta Orde 4	0.000012
ode23	0.000027
ode45	0

Berdasarkan keempat metode yang digunakan, solusi dengan ode45 adalah yang terbaik berdasarkan MAE.

*\*script untuk perbandingan tabel ada di lampiran*



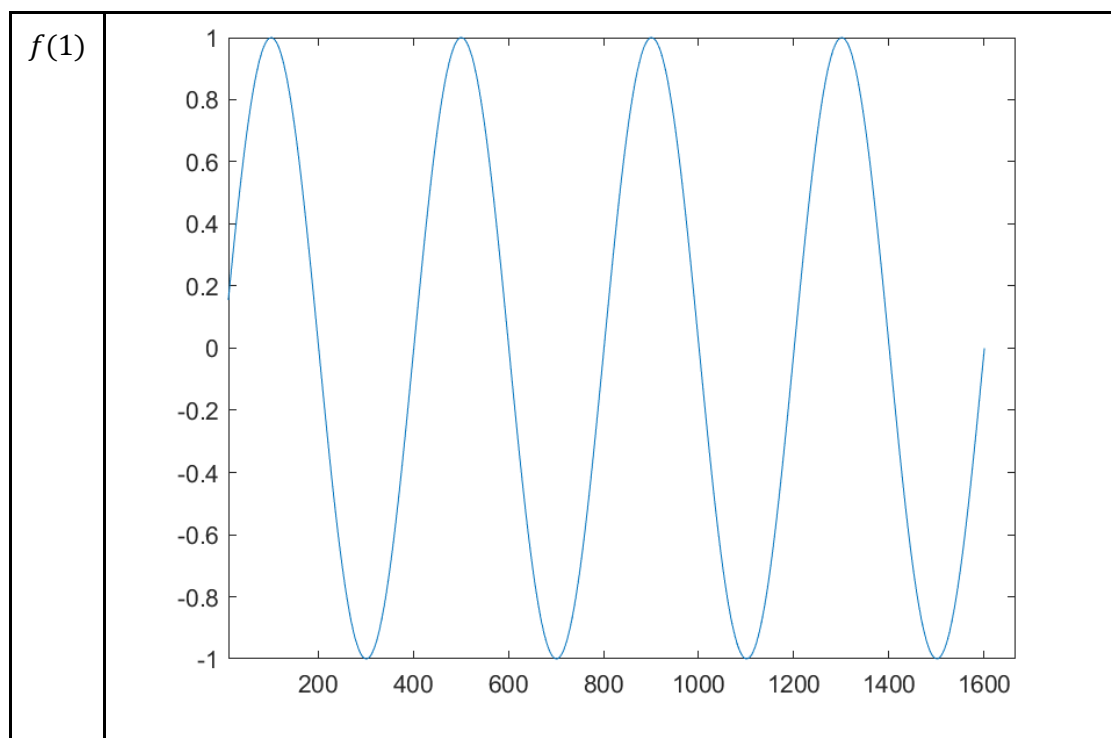
## BAB 6 Transformasi

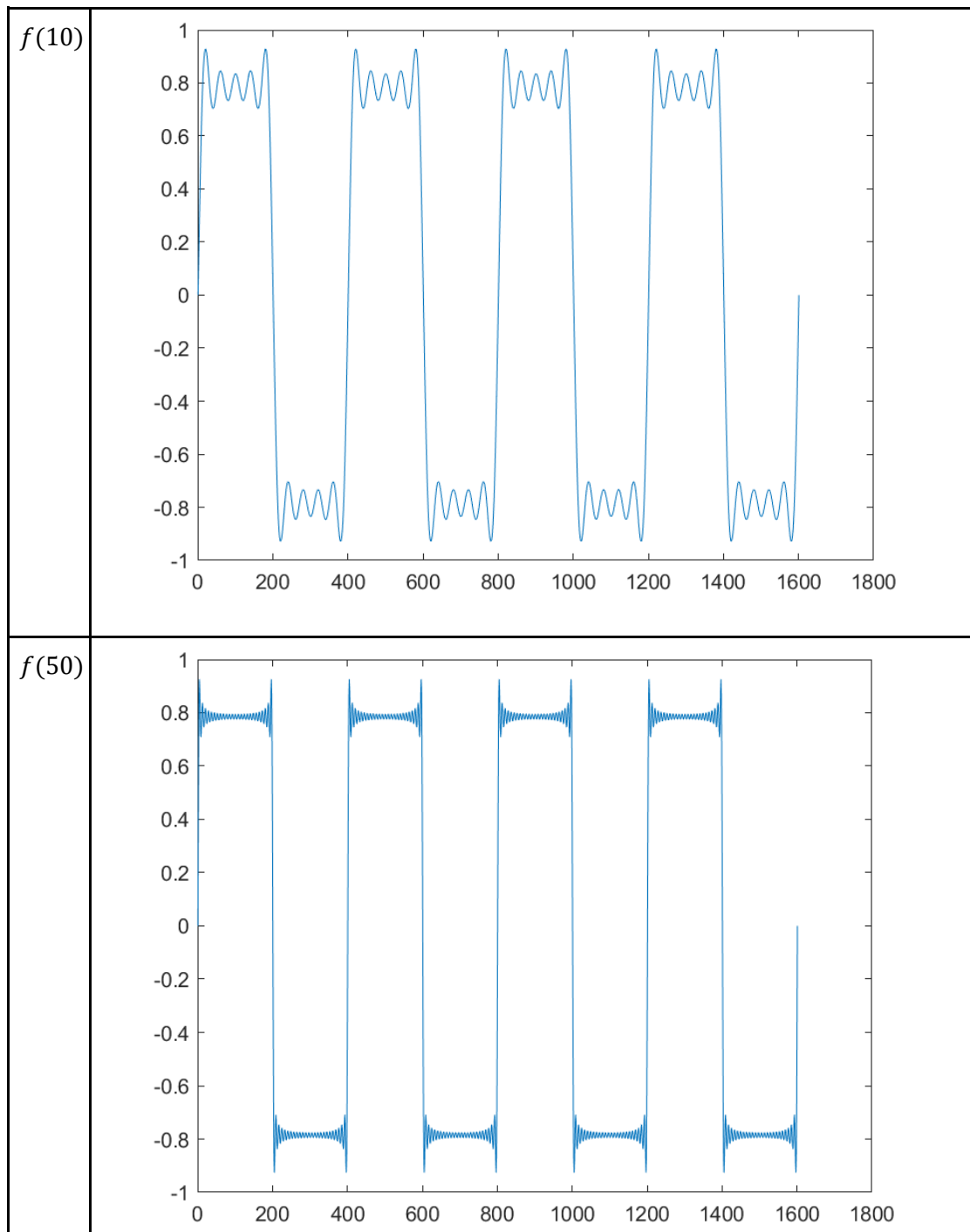
Transformasi atau transformasi integral merupakan suatu metode matematis yang dapat merepresentasikan suatu fungsi ke dalam bentuk lain atau dari suatu domain ke domain lainnya. Ada berbagai macam transformasi integral yang dapat dipelajari diantaranya adalah transformasi fourier, transformasi laplace, transformasi weierstrass, transformasi mellin, dan lainnya. Pada bab kali ini akan dibahas dua macam transformasi yang umumnya akan dipelajari di teknik elektro yaitu transformasi fourier dan transformasi laplace.

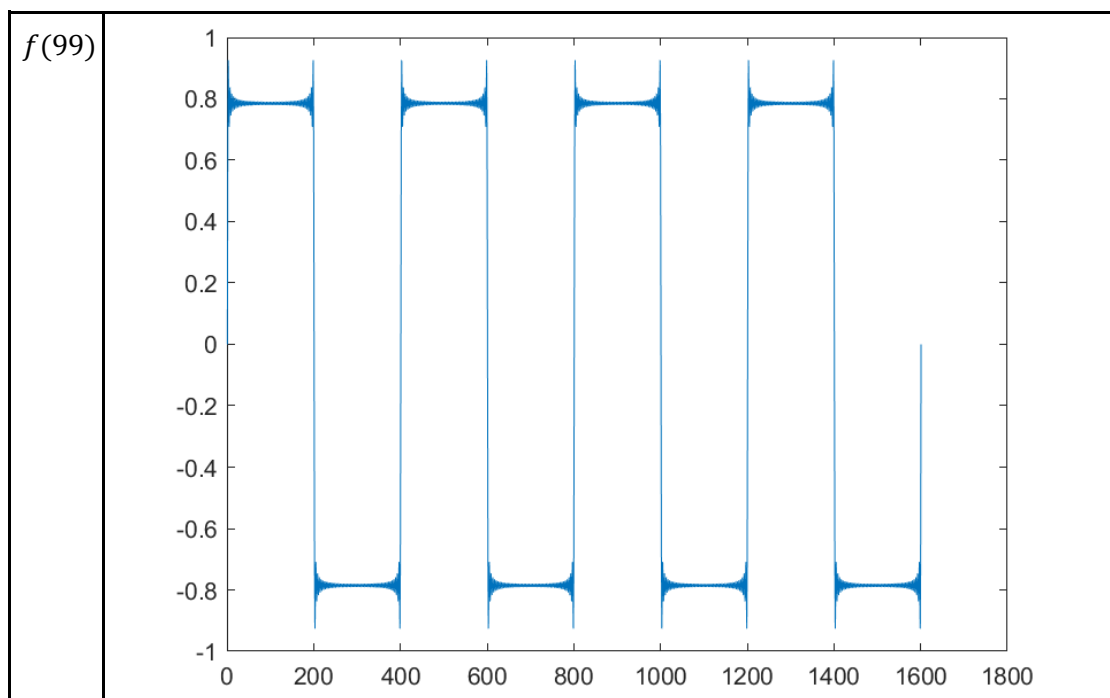
### 6.1. Transformasi Fourier Kontinu

Transformasi Fourier sendiri merupakan salah satu transformasi integral yang dapat merepresentasikan domain waktu  $x(t)$  ke dalam domain frekuensi  $X(\omega)$ . Penemu dari konsep Fourier, Joseph Fourier, mengemukakan bahwa "Setiap fungsi periodik (sinyal) dapat dibentuk dari penjumlahan gelombang-gelombang sinus dan cosinus". Sebagai contoh, sinyal kotak merupakan penjumlahan dari fungsi-fungsi sinus sebagai berikut

$$f(x) = \sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \frac{\sin(7x)}{7} + \frac{\sin(9x)}{9} + \dots$$







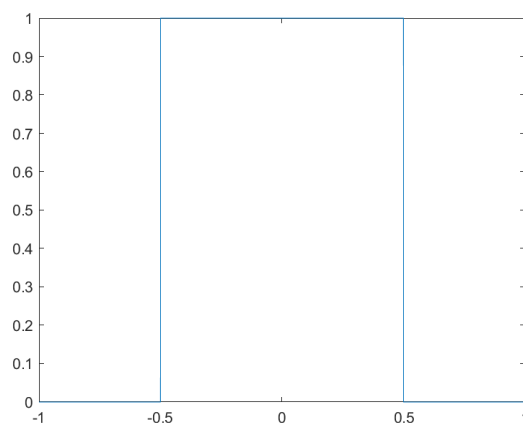
Fungsi diatas merupakan salah satu contoh dari deret fourier dan dari deret tersebutlah transformasi fourier diturunkan. Rumus dari transformasi fourier adalah sebagai berikut

$$X(\omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$

Catatan,  $e^{-j\omega t} = \cos(\omega t) - j \sin(\omega t)$

Sebagai contoh untuk transformasi fourier, suatu fungsi  $\text{rect}(t)$  yang merupakan fungsi rectangle dari  $-1/2 \leq t \leq 1/2$  akan dicari transformasi fouriernya

```
syms t %mendefinisikan variabel waktu t
fplot(rectangularPulse(t), [-1 1])
```



$$X(\omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$



$x(t)$  merupakan fungsi rectangle sehingga hanya memiliki nilai 1 ketika  $-1/2 \leq t \leq 1/2$ , maka

$$X(\omega) = \int_{-1/2}^{1/2} (1)e^{-j\omega t} dt$$

$$X(\omega) = -\frac{1}{j\omega} [e^{-j\omega t}]_{-1/2}^{1/2} = -\frac{1}{j\omega} (e^{-\frac{j\omega}{2}} - e^{\frac{j\omega}{2}})$$

$$X(\omega) = -\frac{1}{j\omega} (e^{-\frac{j\omega}{2}} - e^{\frac{j\omega}{2}}) \times \left(\frac{2}{2}\right) = \frac{2}{\omega} \left( \frac{e^{-\frac{j\omega}{2}} - e^{\frac{j\omega}{2}}}{2j} \right) = \left(\frac{2}{\omega}\right) \left(\sin\left(\frac{\omega}{2}\right)\right)$$

Dari sini dapat dilihat bahwa metode penyelesaian transformasi fourier membutuhkan sedikit waktu untuk penyelesaiannya, dengan adanya matlab sebagai software komputasi akan mempermudah proses pengerjaan suatu masalah salah satunya yaitu transformasi fourier ini. Dengan soal yang sama akan diselesaikan menggunakan program pada matlab sebagai berikut

```
syms t
f=rectangularPulse(t)
fourier(f) %syntax untuk fourier transform
```

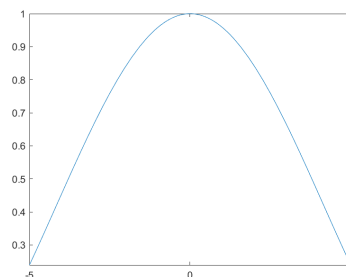
ans =

$$\frac{\sin(\frac{\omega}{2}) + \cos(\frac{\omega}{2})i}{\omega} - \frac{-\sin(\frac{\omega}{2}) + \cos(\frac{\omega}{2})i}{\omega}$$

Dari jawaban tersebut dapat kita sederhanakan menjadi

$$\frac{\sin(\frac{\omega}{2}) + \cos(\frac{\omega}{2})i}{\omega} - \frac{-\sin(\frac{\omega}{2}) + \cos(\frac{\omega}{2})i}{\omega} = \frac{\sin(\frac{\omega}{2}) + \cos(\frac{\omega}{2})i + \sin(\frac{\omega}{2}) - \cos(\frac{\omega}{2})i}{\omega}$$
$$= \left(\frac{2\sin(\frac{\omega}{2})}{\omega}\right)$$

```
fplot(fourier(f))
```







Untuk contoh lainnya akan digunakan persamaan diferensial. Perlu diketahui bahwa persamaan diferensial pada transformasi fourier memiliki konsep yang sama hanya saja perlu diingat beberapa persamaan berikut

$$\mathcal{F}\{x'(t)\} = i\omega \mathcal{F}\{x(t)\} \text{ dan } \mathcal{F}\{x''(t)\} = -\omega^2 \mathcal{F}\{x(t)\}$$

Sebagai contoh akan digunakan persamaan orde 1 yaitu  $\frac{dy}{dt} + 5y = 1$ .

```
syms t w y(t) Y
f = diff(y,t)+5*y-1
```

f(t) =

$$\frac{\partial}{\partial t} y(t) + 5y(t) - 1$$

```
F=fourier(f,t,w)
```

F =  
5 fourier(y(t), t, w) - 2π∂(w) + w fourier(y(t), t, w)i

```
F=subs(F,fourier(y(t), t, w),Y)
```

F =  
5 Y - 2 π ∂(w) + Y w i

```
ys=solve(F,Y) %untuk mencari nilai Y
```

ys =  
$$\frac{2 \pi \partial(w)}{5 + w i}$$

Untuk contoh lainnya yaitu menggunakan bilangan euler. Perlu diingat pula bahwa dalam menggunakan bilangan euler sebagai fungsi dalam transformasi fourier perlu ditambahkan nilai absolut pada fungsi waktunya.

```
g=exp(-5*abs(t))
```

g =  
$$e^{-5|t|}$$

```
gf=fourier(g)
```

gf =  
$$\frac{10}{w^2 + 25}$$





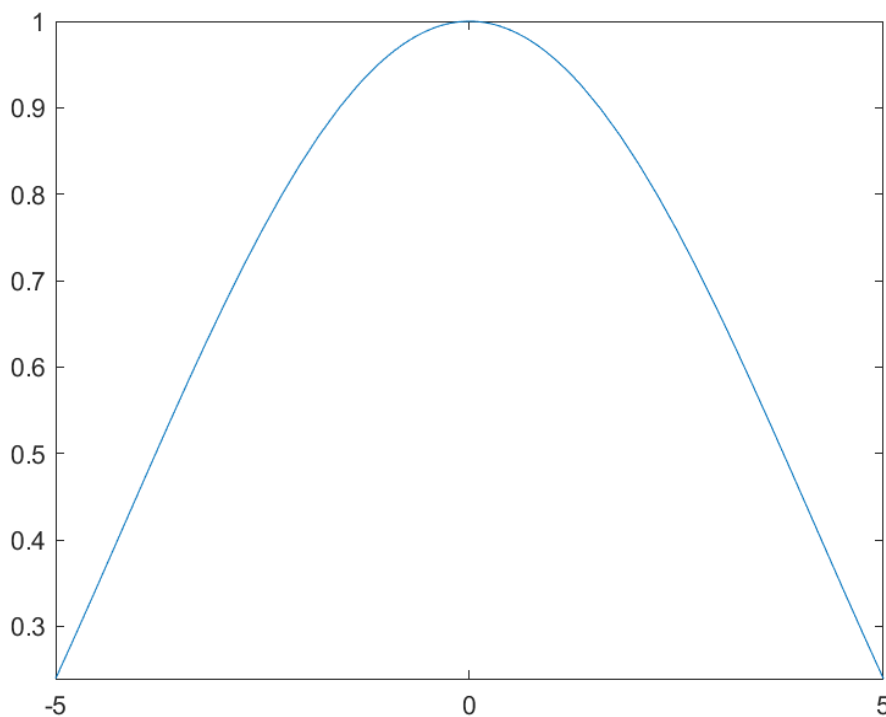
## 6.2. Transformasi Fourier Balik (Inverse)

Setelah mempelajari bagaimana cara melakukan transformasi fourier, selanjutnya akan dipelajari juga invers dari transformasi fourier atau perubahan domain frekuensi ke domain waktu. Rumus dari transformasi fourier adalah sebagai berikut

$$x(t) = \mathcal{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{-j\omega t} d\omega$$

Untuk contoh yang akan kita pakai adalah dengan menggunakan  $\frac{2\sin(\frac{\omega}{2})}{\omega}$  yang kita dapatkan pada contoh transformasi fourier sebelumnya. Untuk syntax yang digunakan adalah sebagai berikut

```
syms w %definisikan variabel w
g=(2*sin(w/2))/w;
fplot(g)
```



```
ifourier(g)
```

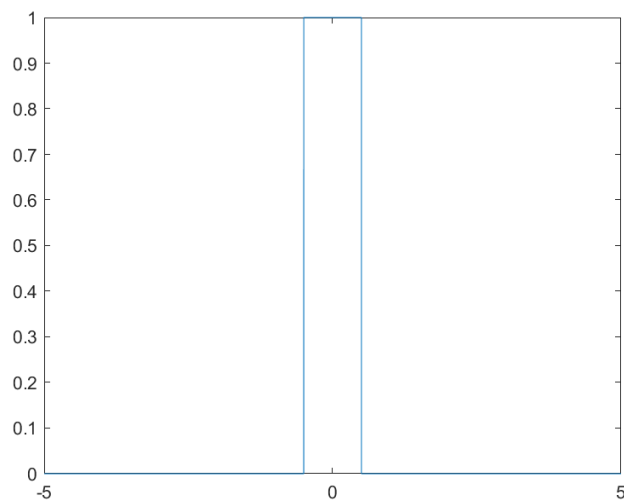
ans =

$$\frac{2\pi \operatorname{heaviside}(x - 1/2) - 2\pi \operatorname{heaviside}(x + 1/2)}{2\pi}$$



Note : heaviside merupakan fungsi yang keluarannya hampir sama seperti pada sinyal step.

```
fplot(ifourier(g))
```



## 6.3. Transformasi Laplace

Transformasi laplace adalah metode transformasi integral yang mengubah fungsi dalam domain real (biasanya dalam domain waktu) ke domain variabel kompleks (dalam domain frekuensi kompleks atau domain s). (MATLAB HANYA DAPAT MENYELESAIKAN TL UNILATERAL).

$$\mathcal{L}\{x(t)\} = X(s) = \int_0^{\infty} x(t)e^{-st} dt$$

Contoh:

### 1. Eksponensial

$$x(t) = e^{-2t}$$

Perhitungan Analitis:

$$\begin{aligned} X(s) &= \int_0^{\infty} e^{-2t} e^{-st} dt = \int_0^{\infty} e^{-(s+2)t} dt \\ X(s) &= -\frac{1}{s+2} (e^{-(s+2)t})_0^{\infty} = -\frac{1}{s+2} (0 - 1) \\ X(s) &= \frac{1}{s+2} \end{aligned}$$

Dengan Matlab:

Buat simbolik variabel t s

```
syms t s
```



Definisikan fungsi

```
x=exp(-2*t)
```

x =

$$e^{-2t}$$

Transformasi laplace dengan fungsi berikut

```
X=laplace(x,t,s)
```

X =

$$\frac{1}{s+2}$$

2. PD orde 1

$$2 \frac{dy}{dt} + 3y = 10$$

$$y(0) = 10$$

Deklarasikan variabel baru

```
syms t s y(t) Y
```

Definisikan persamaan diferensial yang akan ditransformasikan

```
f=2*(diff(y,t))+3*y-10
```

f(t) =

$$2 \frac{\partial}{\partial t} y(t) + 3y(t) - 10$$

Transformasi laplace F dengan fungsi berikut

```
F=laplace(f,t,s)
```

F =



$$2s \text{ laplace}(y(t), t, s) - 2y(0) - \frac{10}{s} + 3 \text{ laplace}(y(t), t, s)$$

Substitusikan  $\text{laplace}(y(t), t, s)$  dengan variabel Y

```
F=subs(F,laplace(y(t), t, s),Y)
```

F =

$$3Y - 2y(0) + 2Ys - \frac{10}{s}$$

Substitusikan kondisi awal  $y(0)=0$

```
F=subs(F,y(0),0)
```

F =

$$3Y + 2Ys - \frac{10}{s}$$

Cari Y yang merupakan hasil transformasi dengan:

```
ys=solve(F,Y)
```

ys =

$$\frac{10}{s(2s + 3)}$$

3. PD orde 2

$$\frac{d^2x}{dt^2} + 5 \frac{dx}{dt} + 8x = 20$$

$$x(0) = 5 \text{ dan } x'(0) = 3$$

```
syms x(t) t s X
```

```
F=diff(diff(x,t),t)+7*diff(x,t)+10*x-20
```

F(t) =



$$\frac{\partial^2}{\partial t^2} x(t) + 7 \frac{\partial}{\partial t} x(t) + 10x(t) - 20$$

Transformasi laplace F dengan fungsi berikut

```
LF=laplace(F,t,s)
```

LF =

$$7s \text{ laplace}(x(t), t, s) - 7x(0) - s x(0) + s^2 \text{ laplace}(x(t), t, s) - ((\frac{\partial}{\partial t} x(t))_{t=0}) - \frac{20}{s} + 10 \text{ laplace}(x(t), t, s)$$

Substitusikan laplace(y(t), t, s) dengan variabel X

```
LF=subs(LF,laplace(x(t), t, s),X)
```

LF =

$$10X - 7x(0) + 7Xs - s x(0) - ((\frac{\partial}{\partial t} x(t))_{t=0}) + X s^2 - \frac{20}{s}$$

Substitusikan kondisi awal x(0)=5

```
LF=subs(LF,x(0),5)
```

LF =

$$10X - 5s + 7Xs - ((\frac{\partial}{\partial t} x(t))_{t=0}) + X s^2 - \frac{20}{s} - 35$$

Substitusikan kondisi awal x'(0)=3

```
LF=subs(LF,subs(diff(x(t), t), t, 0),3)
```

LF =

$$10X - 5s + 7Xs + X s^2 - \frac{20}{s} - 38$$

Cari X yang merupakan hasil transformasi dengan:



```
ys=solve(LF,X)
```

ys =

$$\frac{5s + \frac{20}{s} + 38}{s^2 + 7s + 10}$$

## 6.4. Transformasi Laplace Balik (Inverse)

Transformasi Laplace balik adalah metode integral yang mengubah fungsi domain s ke domain waktu

$$\mathcal{L}^{-1}\{X(s)\} = x(t) = \frac{1}{2\pi} \int_{\sigma-j\omega}^{\sigma+j\omega} X(s)e^{st} ds$$

Mencari Invers dengan metode ini melibatkan integral Kontur dalam bidang kompleks yang relatif sulit, karena itu hanya akan digunakan komputasi dengan Matlab

1.  $X(s) = \frac{1}{s+2}$

Buat simbolik variabel t

```
syms t s
```

Definisikan fungsi

```
X=1/(s+2)
```

X =

$$\frac{1}{s+2}$$

Transformasi laplace dengan fungsi berikut

```
x=ilaplace(X,s,t)
```

x =



$$e^{-2t}$$

$$2. Y(s) = \frac{10}{s^2 + 3s}$$

Buat simbolik variabel t

```
syms t s
```

Definisikan fungsi

```
Y=10/(s^2+3*s)
```

Y =

$$\frac{10}{s^2 + 3s}$$

Transformasi laplace dengan fungsi berikut

```
y=ilaplace(Y,s,t)
```

y =

$$\frac{10}{3} - \frac{10e^{-3t}}{3}$$

$$3. X(s) = \frac{5s + \frac{20}{s} + 38}{s^2 + 7s + 10}$$

Buat simbolik variabel t

```
syms t s
```

Definisikan fungsi

```
X=(5*s+(20/s)+38)/(s^2+7*s+10)
```

X =

$$\frac{5s + \frac{20}{s} + 38}{s^2 + 7s + 10}$$

Transformasi laplace dengan fungsi berikut



```
x=ilaplace(X,s,t)
```

x =

$$6e^{-2t} - 3e^{-5t} + 2$$