# Komputer Aplikasi TE VI

Jana Utama

# *MODUL 1*

# PERINTAH DASAR MATLAB, ARRAY DAN OPERASINYA, GRAFIK 2 DIMENSI SERTA PEMROGRAMAN *.M FILE

## PERINTAH DASAR MATLAB

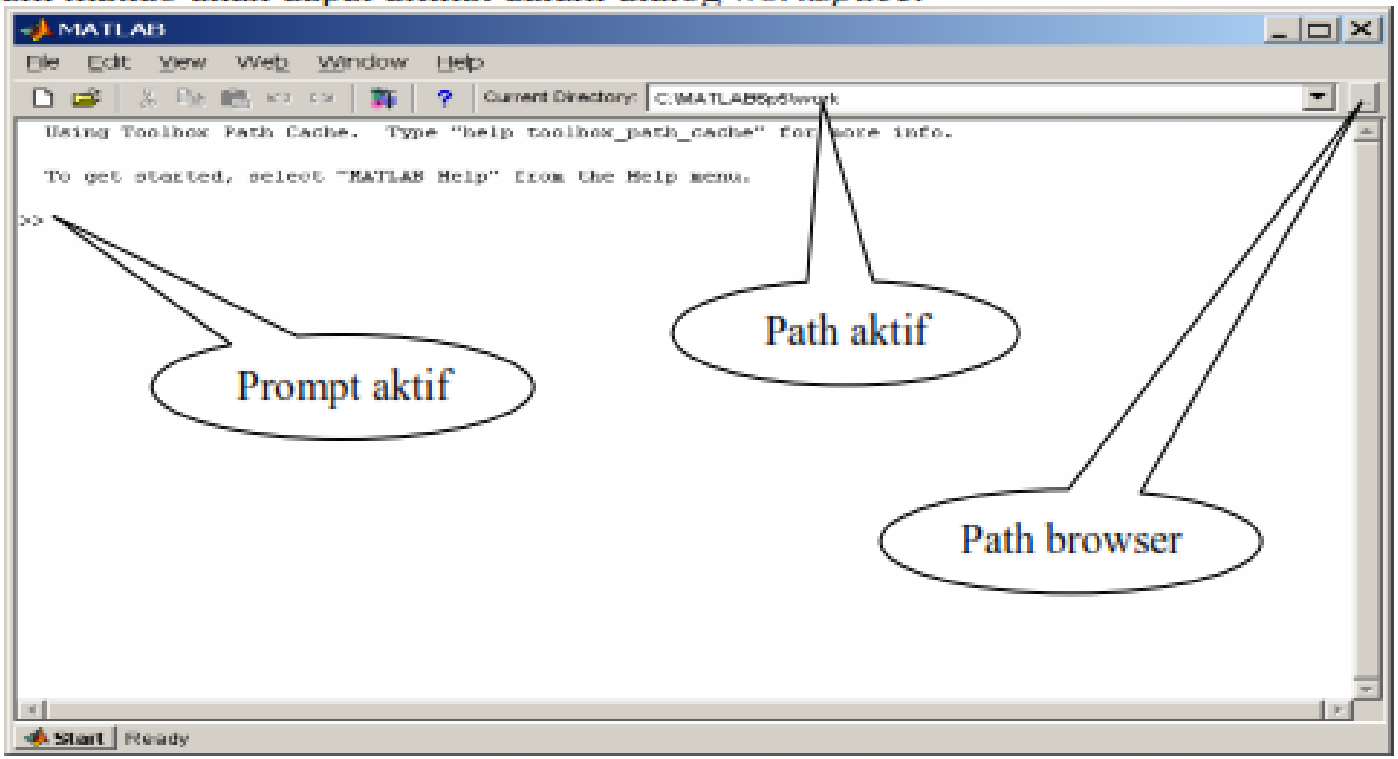
### Pendahuluan

Matlab atau singkatan dari Matrix Laboratory, adalah *software* yang dibuat oleh Mathworks ( **http://www.mathworks.com** ) dengan bahasa C. Matlab merupakan *software* yang berisi fungsi-fungsi Matematika lengkap dengan fasilitas-fasilitas grafis yang menarik. Fungsi-fungsi matematika yang disediakan oleh Matlab bisa bersifat numerik maupun simbolik (analitik).

Sejalan dengan perkembangan Matlab, telah muncul pula *software-software* sejenis yang mempunyai fungsi dan tujuan yang sama dalam pembuatannya, misalnya : *Mathematics, Derive, Mapple* dan lain sebagainya. Hanya saja ada beberapa kelebihan dari *software* Matlab ini yaitu kita dapat memprogram fungsi-fungsi Matematika yang kita inginkan (*User Defined Function*) dengan mudah karena dilengkapi dengan sintaks pemrograman yang sering kita jumpai dalam bahasa pemrograman lainnya seperti : *if .. then.., for .. to.. do.., while..do*, dan lain sebagainya. Selain itu kelebihan utama dari *software* ini adalah adanya *Toolbox* yang menyediakan sekumpulan fungsi-fungsi matematis dalam Matlab yang telah dibuat khusus untuk tujuan tertentu. Contohnya adalah : *Neural Network Toolbox*, *Fuzzy System Toolbox*, *Signal Processing Toolbox*, *Image Processing Toolbox*, *Statistics Toolbox*, *Financial Toolbox*, *Control System Toolbox*, *Communication Toolbox* dan lain sebagainya.

## Variabel, *Workspace* dan perintah dasar Matlab

Perhitungan yang dijalankan dalam program Matlab sama halnya dengan menggunakan kalkulator pada umumnya, artinya nilai yang akan dihitung tidak memerlukan deklarasi terlebih dahulu. Hasil perhitungan untuk suatu nilai yang tidak tersimpan dalam variabel akan disimpan dalam variabel *default* yaitu ans (*answer*). Seluruh variabel yang telah ditulis dalam memori oleh program matlab akan dapat dilihat dalam dialog *workspace*.




**Gambar 1. 1. Matlab *Command Windows***

Pada saat kita pertama kali menjalankan Matlab, maka akan muncul *prompt* Matlab dengan tanda » , artinya kita sedang berada di *Command Windows* Matlab , dan Matlab telah siap untuk menerima input dari *keyboard*.
Persamaan matematika biasa bisa langsung dikerjakan di sini, contoh :

```
» 3 * 8
ans =
    24
```

atau bisa juga melakukan perhitungan dengan variabel yang telah terdefinisi sebagai numerik, contoh :

```
» a = 3
a =
     3
» b = 8
b =
     8
» c = a * b
c =
    24
```

Operasi Aritmatika dasar yang disediakan oleh Matlab adalah : +,-,*,/,\ dan ^.
Matlab mengikuti aturan penamaan variabel secara *Case Sensitive*, artinya Matlab membedakan antara variabel yang ditulis dengan huruf kecil dan huruf besar, contoh : variabel a dengan A merupakan dua variabel yang berbeda. Selain itu karakter yang diperkenankan untuk menamakan variabel maksimal 31 karakter dan harus dimulai oleh huruf. Kemudian ada beberapa variabel khusus yang dimiliki oleh Matlab yaitu :

**Tabel 1. 1. Variabel khusus yang terdapat pada Matlab**

| Variabel Khusus | Keterangan |
|---|---|
| ans | Variabel *default* untuk hasil perhitungan. |
| pi | bilangan pi. |
| eps | bilangan terkecil dimana kalau ditambah 1 akan menghasilkan nilai *floating point* terkecil pada komputer. |
| NaN | bukan sebuah angka. |
| i dan j | $\sqrt{-1}$ (bilangan kompleks) |
| realmin | angka real positif terkecil yang dapat digunakan |
| Realmax | angka real positif terbesar yang dapat digunakan |

Perintah dasar *who* dan *whos* adalah perintah untuk melihat daftar dari variabel yang sedang aktif di memori beserta ukuran dan tipe bilangannya.

```
» who                % perintah menampilkan variabel aktif di memori
Your variables are:
a          ans        b          c

» whos               % perintah menampilkan variabel dan tipe bilangannya
  Name        Size          Bytes       Class
   a             8            double     array
   ans           1x1          8          double array
   b             1x1          8          double array
   c             1x1          8          double array
  Grand total is 4 elements using 32 bytes
```

Variabel-variabel yang ditampilkan di atas semuanya bersifat sementara, artinya apabila kita meninggalkan Matlab, maka variabel-variabel tersebut akan hilang dari memori. Untuk

menghindari hal tersebut variabel-variabel di atas bisa kita simpan dalam suatu *file* variabel dengan perintah

```
» save nama_file                                    % berisi_variabel
```

Pada saat perintah di atas dikerjakan seluruh variabel yang terdapat pada memori akan disimpan pada suatu *file* *.mat lengkap dengan tipe bilangan pada masing-masing variabel tersebut. Untuk memanggilnya kembali cukup dengan perintah :

```
» load nama_file                              % berisi_variabel
```

Sedangkan untuk menghapus variabel di memori perintahnya adalah :

```
» clear nama_variabel
```

Apabila perintah *Clear* tanpa nama variabel, maka semua variabel yang aktif dalam *workspace* semuanya akan terhapus. Untuk melihat semua *file* *.m, *file* *.mex serta *file* *.fig yang pada direktori aktif dapat dilakukan dengan menggunakan perintah dasar berikut:

```
» what      % perintah menampilkan file matlab dalam direktori aktif
```

Perintah membersihkan layar adalah :

```
» clc                        % perintah membersihkan layar  Command Windows
```

Bila kita bekerja dengan Matlab seringkali tampilan hasil proses intruksi Matlab melebihi dari satu halaman layar, untuk melihat dari hasil tampilan pada layar sebelumnya bisa kita gunakan *scroll bar* yang berada di sebelah kanan layar, tapi Matlab juga mempunyai perintah dasar *diary*, dimana perintah tersebut akan menyimpan hasil pekerjaan kita di dalam sebuah *file* *.txt. Perhatikan contoh berikut :

```
» diary on                  % untuk mengaktifkan diary
      <perintah>
      <perintah>
      <perintah>
» diary coba.txt           % menyimpan tampilan layar workspace pada file
                           % coba.txt
» diary off                % menutup kembali diary atau menutup file
                           % coba.txt
```

Untuk melihat hasilnya cukup dengan perintah *type*.

```
» type coba.txt
```

Perintah dasar lainnya adalah *help*, bila kita ingin mengetahui fungsi dan perintah dasar serta cara pemakaian parameter-parameter dari fungsi dan perintah-perintah dasar Matlab.

```
» help diary               % membuka informasi tentang kata cadangan diary
» help clc                 % membuka informasi tentang kata cadangan clc
```

## Bilangan Kompleks

Membangun persamaan matematis pada Matlab dapat dilakukan dengan menggunakan variabel khusus i atau j untuk menandakan komponen imajinernya, contoh bilangan kompleks adalah :

```
» c1 = 1 - 2i              % menuliskan bilangan kompleks pada variabel c1
c1 =
   1.0000 - 2.0000i
» c2 = 1 - 2j              % menuliskan bilangan kompleks pada variabel c2
c2 =
   1.0000 - 2.0000i
» c3 = sqrt(-2)
```

```
c3 =
          0 + 1.4142i
» c4 = c1 + c2
c4 =
   2.0000 - 4.0000i
» c5 = c2*c3
c5 =
   2.8284 + 1.4142i
```

Menurut identitas *Euler* berlaku hubungan berikut :

$$e^{j\theta} = a + bi$$

hubungan antara *Magnitude* M dan sudut θ ditunjukkan oleh hubungan berikut :

$$M = \sqrt{a^2 + b^2}$$

$$\theta = \tan^{-1}\left(\frac{b}{a}\right)$$

$$a = M\cos\theta$$

$$b = M\sin\theta$$

Dalam Matlab, konversi di atas dapat dijelaskan dengan intruksi berikut ini :

```
» c = 2 - 3i
c =
   2.0000 - 3.0000i
» mag_c = abs(c)                % mencari nilai magnitude = M = √(a² + b²)
mag_c =
    3.6056
» sudut_c = angle(c)            % mencari nilai sudut θ
sudut_c =
   -0.9828
» sudut_derajat = sudut_c*180/pi
sudut_derajat =
  -56.3099
» real_c = real(c)              % mencari komponen real suatu bilangan
kompleks
real_c =
     2
» imag_c = imag(c)              % mencari komponen imajiner suatu `
                                % bilangan kompleks
imag_c =
    -3
```

# ARRAY DAN OPERASINYA

## Pendahuluan

Dalam perhitungan matematik, seringkali kita bekerja dengan bilangan tunggal yang disebut dengan **skalar**, namun apabila kita akan mengoperasikan bilangan skalar lebih dari satu dengan operasi yang sama, maka akan memudahkan kita untuk menggunakan **Array**. Array yang dibangun dapat berupa matriks 1-dimensi, matriks 2-dimensi, kita pun dapat pula bekerja sampai matriks n-dimensi.

## Pengalamatan Array

Pengalamatan array dapat dilakukan secara manual berupa input dari *keyboard* dan dapat dilakukan secara otomatis menggunakan fungsi standar yang terdapat dalam Matlab. Misalkan kita mempunyai array x dengan data yang kita isi sebagai berikut :

```
» x = [  0   .1*pi   .2*pi   .3*pi   .4*pi   .5*pi   .6*pi   .7*pi   .8*pi
.9*pi  pi ]
x =
  Columns 1 through 7
         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850
  Columns 8 through 11
    2.1991    2.5133    2.8274    3.1416
```

selanjutnya variabel x dioperasikan oleh fungsi matematis menjadi variabel y sebagai berikut :

```
» y = sin(x)
y =
  Columns 1 through 7
         0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
  Columns 8 through 11
    0.8090    0.5878    0.3090    0.0000
```

Ada tiga cara dalam mengakses setiap elemen array, yaitu :

a. Menggunakan *subscript*, misalnya kita akan melihat elemen ke-5 dari x atau y dengan perintah berikut :

```
» x(5)
ans =
    1.2566
» y(5)
ans =
    0.9511
```

b. Menggunakan *tanda titik dua* (**:**), misalnya kita akan melihat elemen ke-2 sampai ke-6 dari matriks y, kita cukup memasukkan perintah :

```
» y(2:6)
ans =
    0.3090    0.5878    0.8090    0.9511    1.0000
```

c. Menggunakan pola (awal:step:akhir), perhatikan contoh-contoh berikut :

- menampilkan elemen ke-5 sampai akhir dari data y.

```
» y(6:end)
ans =
    1.0000    0.9511    0.8090    0.5878    0.3090    0.0000
```

- Menampilkan elemen data x ke dua sampai dengan elemen ke tujuh dengan step 2, maka yang akan ditampilkan adalah data elemen ke-2, ke-4 dan ke-6, data ke-8 tidak ditampilkan karena 8 lebih besar dari 7.

```
» x(2:2:7)
ans =
      0.3142    0.9425    1.5708
```

Untuk mengakses elemen matrik 2-dimensi dengan ukuran n x m dapat dilakukan dengan cara mengidentifikasi bahwa n adalah baris dan m adalah kolom.

Sintaks : Variabel( n , m )


```
» a = [ 1 2 3; 4 5 6; 7 8 9 ]
a =
     1     2     3
     4     5     6
     7     8     9
» a(1,2)
ans =
     2
```

## Pengisian Elemen Array secara Otomatis

Tabel 1.2 di bawah ini adalah intruksi cara pengisian elemen array secara otomatis.

**Tabel 1. 2. Cara Pengisian Elemen Array Otomatis**

| Pengisian Elemen Array | |
|---|---|
| **x=[ 2 3*pi sqrt(9) 2-3i ]** | Mengisi elemen array x dengan bilangan-bilangan yang di-input-kan. |
| **x=awal:akhir** | Mengisi elemen array x dimulai dengan nilai awal s.d akhir dengan penambahan/step 1. |
| **x=awal:step:akhir** | Mengisi elemen array x dimulai dengan nilai awal s.d akhir dengan penambahan dengan variabel step. |
| **x=linspace(awal,akhir,n)** | Mengisi elemen array x dari awal s.d akhir sebanyak n elemen, bila n tidak diisi maka banyaknya elemen secara otomatis adalah 100. |
| **x=logspace(awal,akhir,n)** | Mengisi elemen array secara logaritmik dimulai dari $10^{awal}$ s.d $10^{akhir}$ sebanyak n elemen. |

Lakukanlah contoh-contoh berikut di bawah ini :

```
» x = [ 2 2*pi 3-i 5 ]
» x = 0:25
» x = 0:0.01:1
» x = linspace(0,1)
» x = linspace(0,1,200)
» x = logspace(0.1,5,50)
```

Selain instruksi di atas adapula fungsi istimewa untuk pengisian matriks yang semua elemennya bernilai nol atau satu.

```
» ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
```

```
» zeros(3,6)
ans =
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

## Manipulasi Array

Untuk merubah isi suatu nilai dalam elemen matrik dapat dilakukan dengan mengakses elemen yang nilainya akan dirubah dan mengisinya kembali dengan suatu nilai baru. Perhatikan contoh berikut di bawah ini :

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

» a(3,3) = 0            % ubah elemen baris ke tiga kolom ke tiga = 0
a =
     1     2     3
     4     5     6
     7     8     0

» a(2,6) = 1            % set elemen baris ke dua kolom ke enam = 1
a =
     1     2     3     0     0     0
     4     5     6     0     0     1
     7     8     0     0     0     0
```

Meskipun pada mulanya matriks a adalah matriks 3x3 namun dengan pengisian di atas maka secara otomatis baris dan kolomnya akan bertambah menjadi matriks 3x6. Lakukan perintah-perintah berikut di bawah ini, amati hasil tampilan layarnya kemudian jelaskan arti dari perintah-perintah tersebut :

```
» a = [ 1 2 3; 4 5 6; 7 8 9 ]
» b = a(3:-1:1,1:3)
» b = a(3:-1:1,:)
» c = [ a b(:,[1,3]) ]
» b = a(1:2,2:3)
» c = [ 1 3 ]
» b = a(c,c)
» b = a(:)
» b = b'
» b = a
» b(:,2) = [ ]
» b = b.'
» b(2,:) = [ ]
» a(2,:) = b
» b = a(:,[2 2 2])
» d(3:4,:) = a(2:3,:)
» a(2,:) = 0
» iv = inv (a);
```

## Operasi Matematika pada Array

Operasi matematika yang sering digunakan pada suatu Matlab diantaranya +, -, *, /, \, dan ^. Untuk operasi penjumlahan, pengurangan, perkalian dan pemangkatan mungkin sudah banyak dipahami banyak orang karena operasi tersebut merupakan operasi standar yang ada pada setiap bahasa pemrogramman. Lain halnya pembagian, dalam Matlab dikenal dua buah operasi pembagian yaitu ***Right-Division*** **(/)** dan ***Left-Division*** **(\)**. Untuk mengetahui perbedaan antara keduanya dapat dilihat dari contoh berikut:

```
» 1 / 2
ans =
      0.5000
» 1 \ 2
ans =
      2
```

Dari kedua contoh tersebut bisa terlihat apabila kita menggunakan *Right-Division* akan sama dengan $(1)*(2)^{-1}$ sedangkan untuk *Left-Division* akan sama dengan $(1)^{-1}*(2)$. Lebih mudahnya kita dapat rumuskan sebagai,

$$\mathbf{A / B = A * (B)^{-1}}$$
$$\mathbf{A \setminus B = (A)^{-1} * B}$$

Untuk memahami operasi matematika pada matrik, amatilah keluaran yang dihasilkan oleh operasi di bawah ini.

```
» A = [ 1 2 4 ; 3 4 2  ; 1 4 5 ]
A =
     1     2     4
     3     4     2
     1     4     5
» B = magic (3)
B =
     8     1     6
     3     5     7
     4     9     2
» n = 2;
» A + B          % Jumlahkan setiap elemen A dengan elemen B
» A - B          % Kurangi setiap elemen A dengan elemen B
» A * B          % Kali matrik A dengan matrik B
» A.* B          % Kali setiap elemen A dengan elemen B
» A ^ n          % Pangkatkan setiap elemen A dengan n (n merupakan
                 % konstanta)
» A / B          % Right-Division A dengan B (B.A^-1 = A / B)
» A./ B          % Pembagian B / A per-elemen
» A \ B          % Left-Division A dengan B (A^-1.B = B / A)
» A.\ B          % Pembagian A \ B per-elemen
```

## Perbandingan Array

Variabel array dalam Matlab dapat dibandingkan dengan variabel array lain. Hasil dari perbandingan ini akan menunjukkan nilai Logika/Boolean yaitu bernilai ***"1"*** jika hasil perbandingan dinyatakan sama (*equal*) dan ***"0"*** jika hasil dari perbandingan dinyatakan tidak sama. Lakukan contoh berikut di bawah ini :

```
» a = [1 2 3; 4 5 6; 7 8 9]
» a = a'            % Transpose dari data sebelumnya
» b = a.*(-1).^a
» c = 1:9           % Array yang sama dengan a, tapi ukurannya beda
» isequal(a,c)      % Membandingkan ukuran matrik a dan c
» isequal(a,b)
» isequal(a,a)
» isequal(c,c')
» ismember(a,b)     % Membandingkan isi matrik a & b dengan operasi a ∈ b
» ismember(a,c)
» x = 0:2:20
» ismember(a,x)
» ismember(x,a)
» union(a,b)
» intersect(a,b)
» setdiff(a,b)
» setxor(a,b)
```

Amati setiap perubahan dari instruksi di atas, kemudian setelah itu jelaskan arti dari perintah-perintah *isequal*, *ismember*, *union*, *intersect*, *setdiff* dan *setxor*.

# GRAFIK 2-DIMENSI

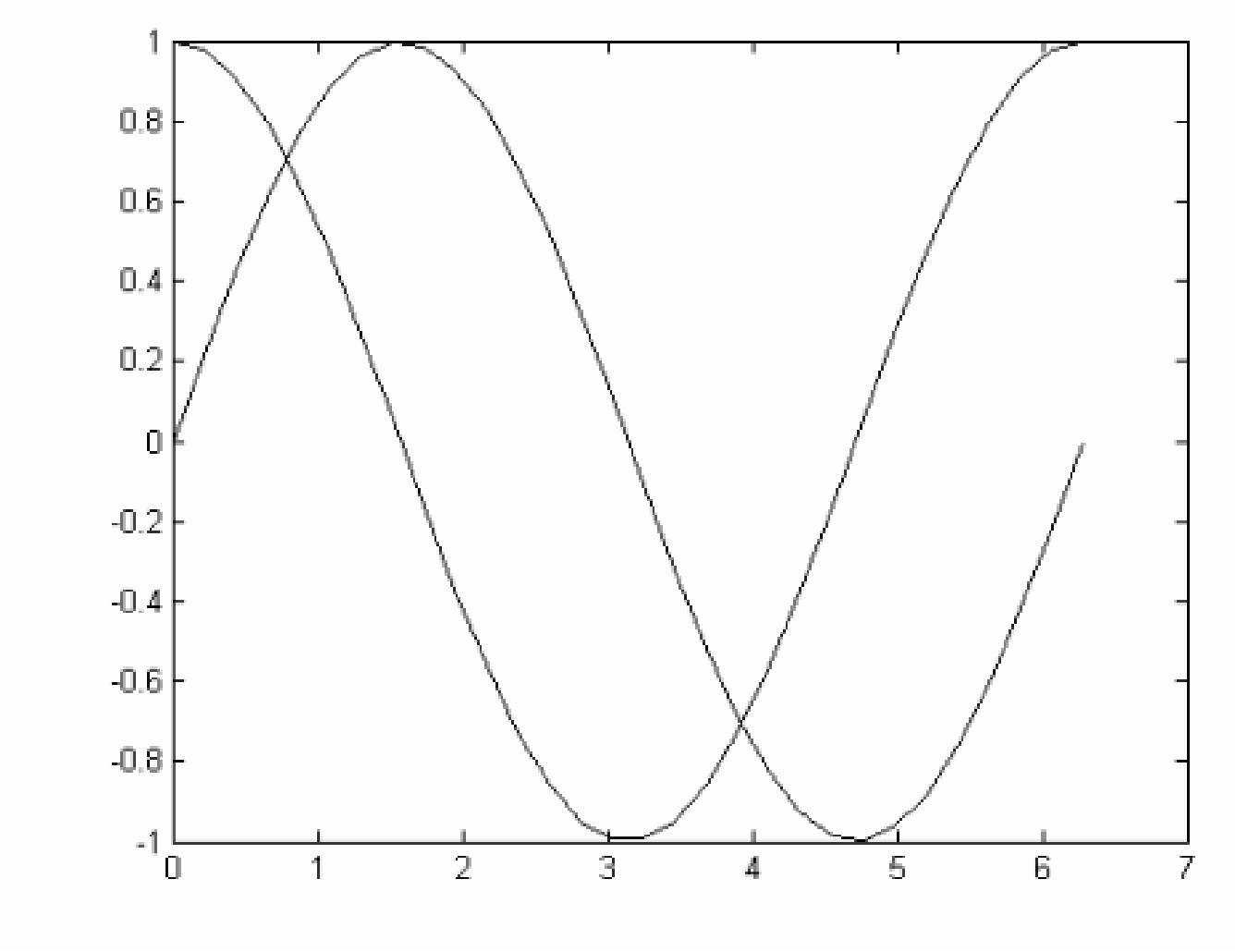
Grafik merupakan interpretasi dari data yang diperoleh, sehingga dalam pembuatannya tidak akan terlepas dari **variabel array** dengan ukuran data tertentu baik array 1-dimensi ataupun array 2-dimensi. Ketika variabel yang berisi data tersebut hanya memiliki satu nilai saja (variabel skalar) maka grafik tidak dapat di buat atau hanya akan berupa kurva titik pada koordinat yang digunakan.

## Perintah Plot

Perhatikan contoh berikut :

```
» x = linspace(0,2*pi,30);      % membuat variabel x sebagai array
» y = sin(x);
» z = cos(x);
» plot(x,y,x,z)                 % variabel x, y, z adalah variabel array
```

Gambar 1.2 menggambarkan kurva hasil plot matrik 1xn yang merupakan fungsi sinus dan cosinus dengan 30 data x adalah $0 \leq x \leq 2\pi$ , sedangkan y dan z berturut-turut merupakan hasil fungsi sinus dan cosinus dari data x. Perintah Plot akan menghasilkan grafik pada windows baru yang disebut dengan *Figure*. Perintah Plot juga memungkinkan kita untuk membuat grafik lebih dari satu dalam satu *Figure*.




**Gambar 1. 2. Grafik hasil Plot array 1 dimensi fungsi sinusoidal**

Selain itu Perintah Plot juga memungkinkan kita untuk memilih jenis garis dan warna yang akan digunakan dalam grafik tersebut. Bentuk dasar dari perintah Plot adalah sebagai berikut :

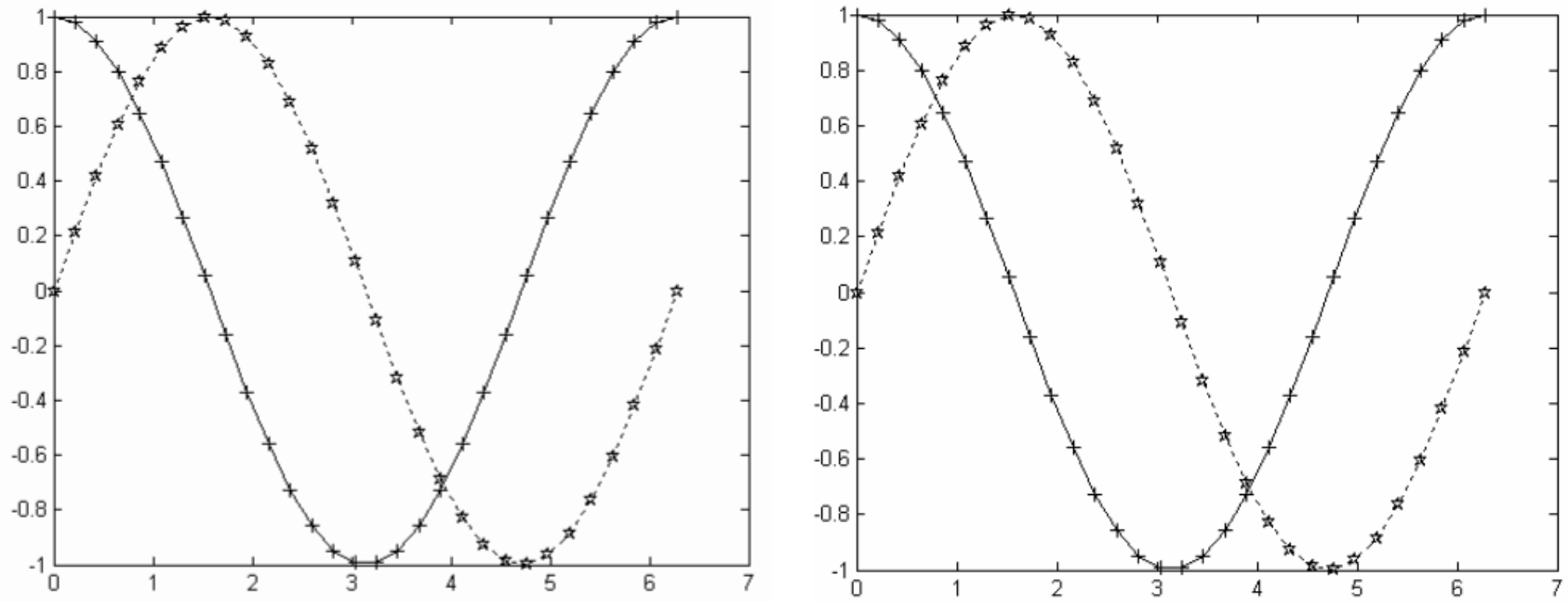
**Tabel 1. 3. Bentuk dasar & warna pada perintah plot**

| Symbol | Color | Symbol | Marker | Symbol | Line Style |
|---|---|---|---|---|---|
| B | Blue | . | Point | - | Solid line |
| G | Green | O | Circle | : | Dotted line |
| R | Red | X | x-mark | -. | Dash-dot line |
| C | Cyan | + | Plus | -- | Dashed line |
| M | Magenta | * | Star | | |
| Y | Yellow | s | Square | | |
| K | Black | d | Diamond | | |
| W | white | v | Triangle down | | |
| | | ^ | Triangle up | | |
| | | > | Triangle left | | |
| | | < | Triangle right | | |
| | | p | Pentagram | | |
| | | h | hexagram | | |

Plot(x,y,'s') dimana s adalah karakter-karakter string yang meliputi *Color, Marker,* dan *Line Style*. Jika dalam fungsi Plot kita tidak memberikan nilai warna, maka secara otomatis Matlab akan memulai warna biru kemudian hijau dan seterusnya sesuai dengan urutan tabel di atas, kemudian garis tebal adalah garis default dari fungsi Plot.

Perhatikan contoh berikut :

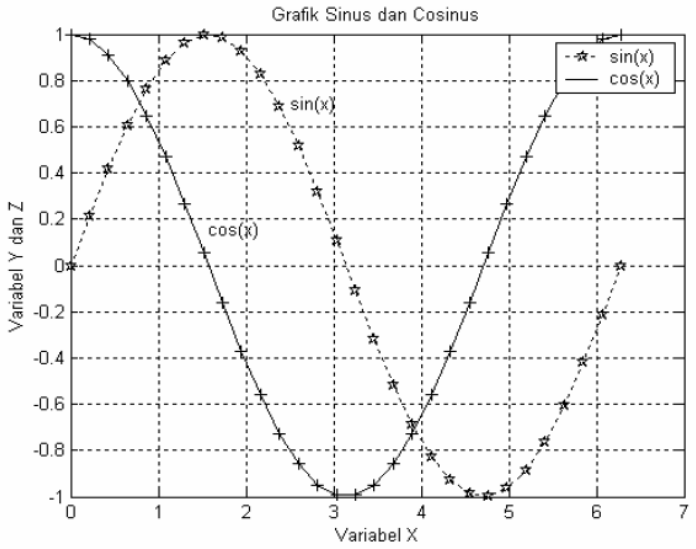
```
» plot(x,y,'g:p',x,z,'c-',x,z,'m+')
```




**Gambar 1. 3. Grafik hasil Plot array 1 dimensi fungsi sinusoidal dengan bentuk dan warna tertentu**

Sebagai latihan cobalah perintah-perintah berikut ini

```
» box off                                  % menghilangkan box pada figure
» xlabel('Variabel X')                     % memberikan label untuk
sumbu x
» ylabel('Variabel Y dan Z')               % memberikan label untuk
sumbu y
» title('Grafik Sinus dan Cosinus')        % memberikan judul
grafik
» grid on, box on                          % menimbulkan garis dan box
pada figure
» text(2.5,0.7,'sin(x)')
» gtext('cos(x)')                          % gunakan mouse untuk
menempatkan text
» legend('sin(x)','cos(x)')
```




**Gambar 1. 4. Grafik hasil Plot array 1 dimensi fungsi sinusoidal dengan legenda**

## Mencetak Grafik

Untuk mencetak dan mengetahui orientasi dari grafik pada figure yang dihasilkan, kita cukup mengetikkan perintah berikut :

```
» print
» orient                          % untuk mengetahui orientasi yang aktif
ans =
portrait
» orient landscape                % mengubah orientasi
» orient tall
```

Cara yang lain adalah dengan menggunakan menu *File* pada jendela *Figure* untuk kemudian memilih menu ***Print***.

## Manipulasi Grafik

Perintah *hold on* adalah perintah untuk menambahkan grafik pada *figure* yang telah ada, dengan perintah ini Matlab tidak akan menghapus grafik yang sudah ada ketika kita memberikan perintah plot yang baru. Lakukanlah perintah di bawah ini :

```
» plot(x,y)
» plot(x,z)
```

Perhatikan hasil dari perintah di atas, kemudian bedakan dengan perintah berikut di bawah ini :

```
» plot(x,y)
» hold on
» plot(x,z)
```

Perintah hold bersifat *toggle*, untuk mengnonaktifkannya cukup dengan perintah *hold off*.

Kita bisa menambahkan jendela baru dengan menggunakan menu New Figure dari menu *file* pada workspace atau menu *file* pada jendela *figure*. Selain itu kita bisa juga bisa menambah figure baru dengan perintah figure pada *command window*.

Selain itu kita bisa menghasilkan beberapa tampilan grafik dalam satu *figure* dengan menggunakan perintah subplot(m,n,p). Perintah ini akan membagi *figure* window menjadi sebuah matriks mxn dari area plot, dan mengaktifkan area-p. Subplot diberi nomor dari kiri ke kanan dan baris paling atas ke baris kedua dan seterusnya. Lakukan contoh berikut :

```
» x = linspace(0.2*pi,30);
» y = sin(x);
» z = cos(x);
» a = 2*sin(x).*cos(x);
» b = sin(x)./(cos(x)-eps);
» subplot(2,2,1)
» plot(x,y),axis([0 2*pi -1 1]), title('sin(x)')
» subplot(2,2,2)
» plot(x,z),axis([0 2*pi -1 1]), title('cos(x)')
» subplot(2,2,3)
» plot(x,a),axis([0 2*pi -1 1]), title('2*sin(x)*cos(x)')
» subplot(2,2,4)
» plot(x,b),axis([0 2*pi -20 20]), title('2*sin(x)/cos(x)')
```

Perintah fplot (plot fungsi) dapat membuat grafik dari suatu fungsi dengan batas yang ditentukan, formatnya adalah sebagai berikut :

```
» fplot('fun',[xmin xmax ymin ymax])
```

xmin, xmax, ymin dan ymax adalah batas-batas skala untuk sumbu x dan sumbu y, dan fun adalah fungsi yang akan diplot.

Lakukanlah perintah berikut

```
» fplot('cos(x)',[ 0 2*pi -1 1])
```

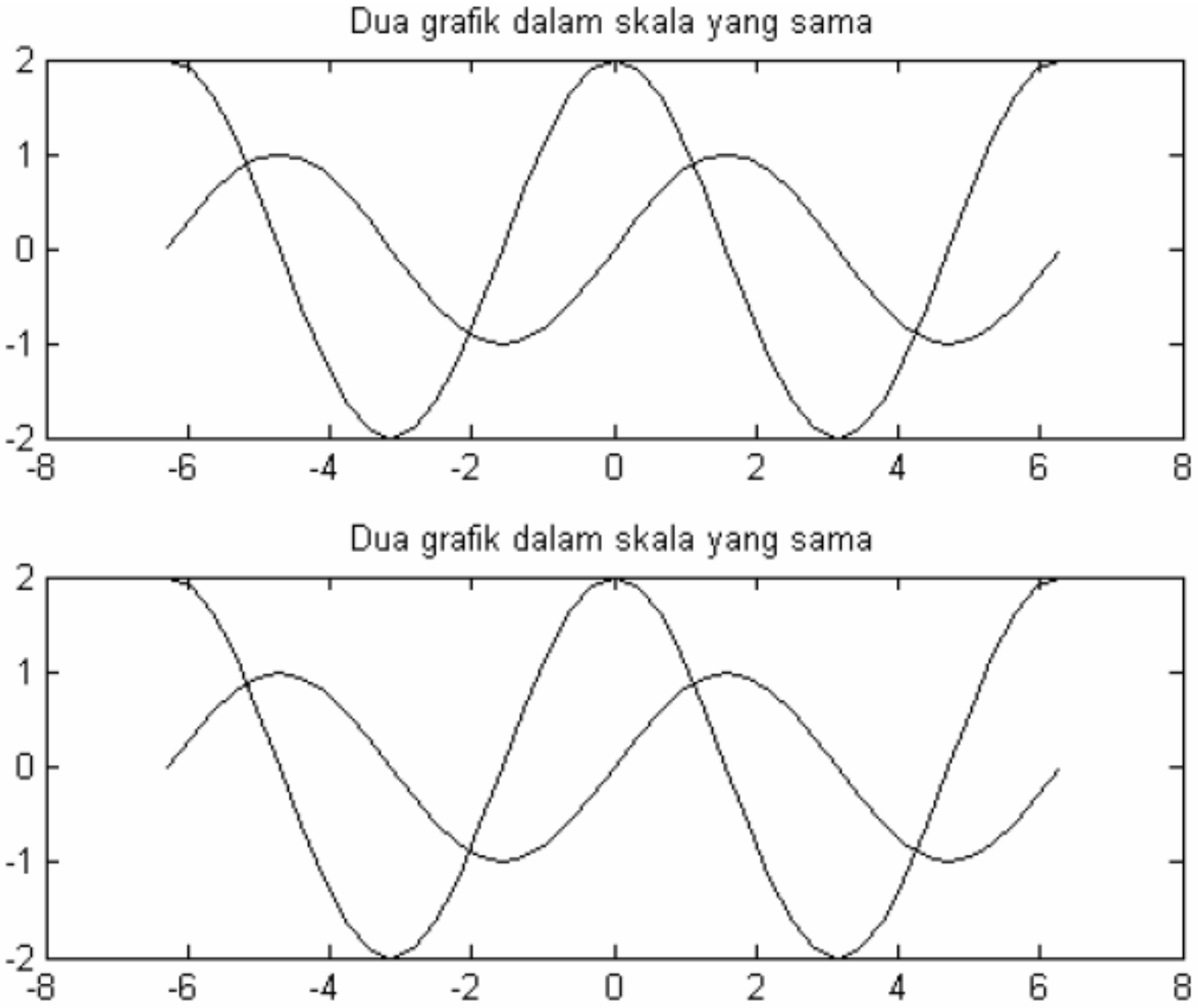
Perintah *Zoom* adalah perintah untuk memperbesar grafik dan bersifat *toggle*. Untuk memperbesarnya cukup dengan mengklik tombol *mouse* kanan dan tombol *mouse* kiri untuk sebaliknya.

## Fungsi-fungsi Grafik 2D lainnya.

- **Plotyy**

  Perintah ini bertujuan untuk menggambarkan dua buah fungsi yang berbeda dengan menggunakan sumbu x yang sama tetapi sumbu y yang berbeda.

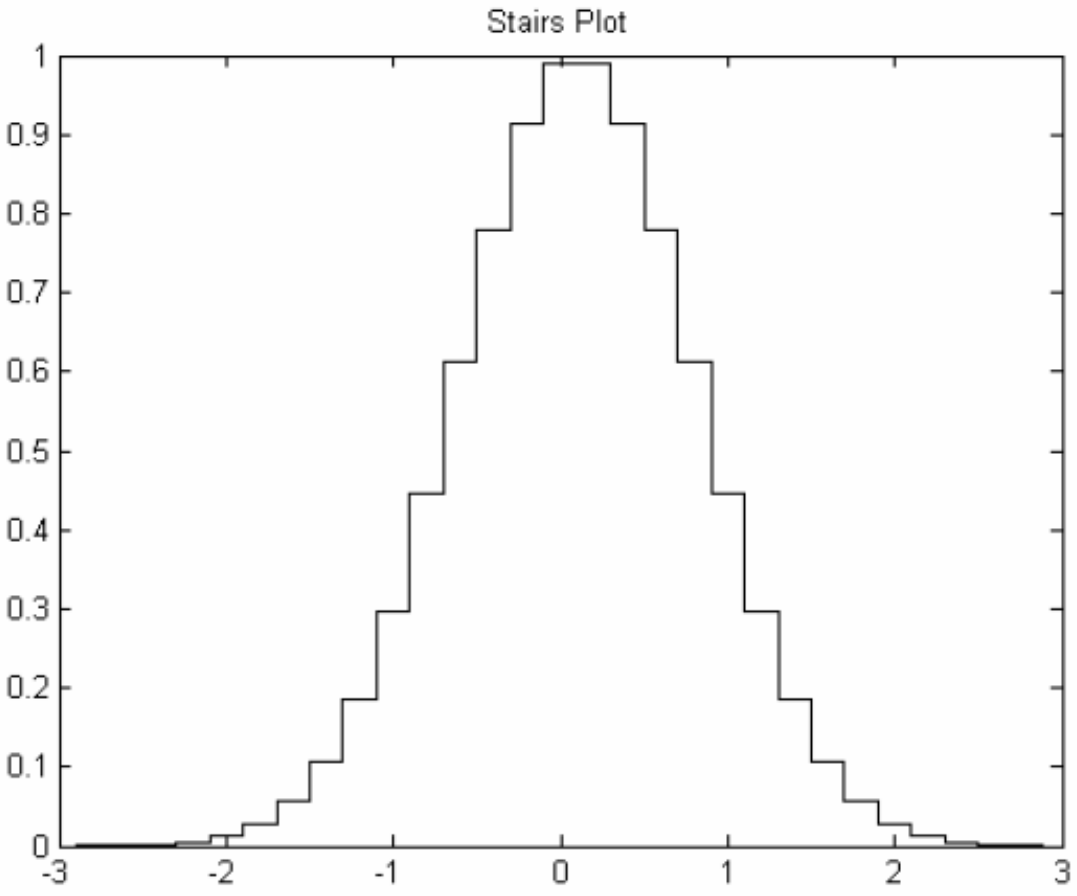
```
» x = -2*pi:pi/10:2*pi;
» y = sin(x)
» z = 2*cos(x)
» subplot(2,1,1)
» plot(x,y,x,z), title('Dua grafik dalam skala yang sama');
» subplot(2,1,2)
» plotyy(x,y,x,z), title('Dua grafik dalam skala yang berbeda');
```




**Gambar 1. 5. Penggunaan perintah Subplot**

- **Stair plot**

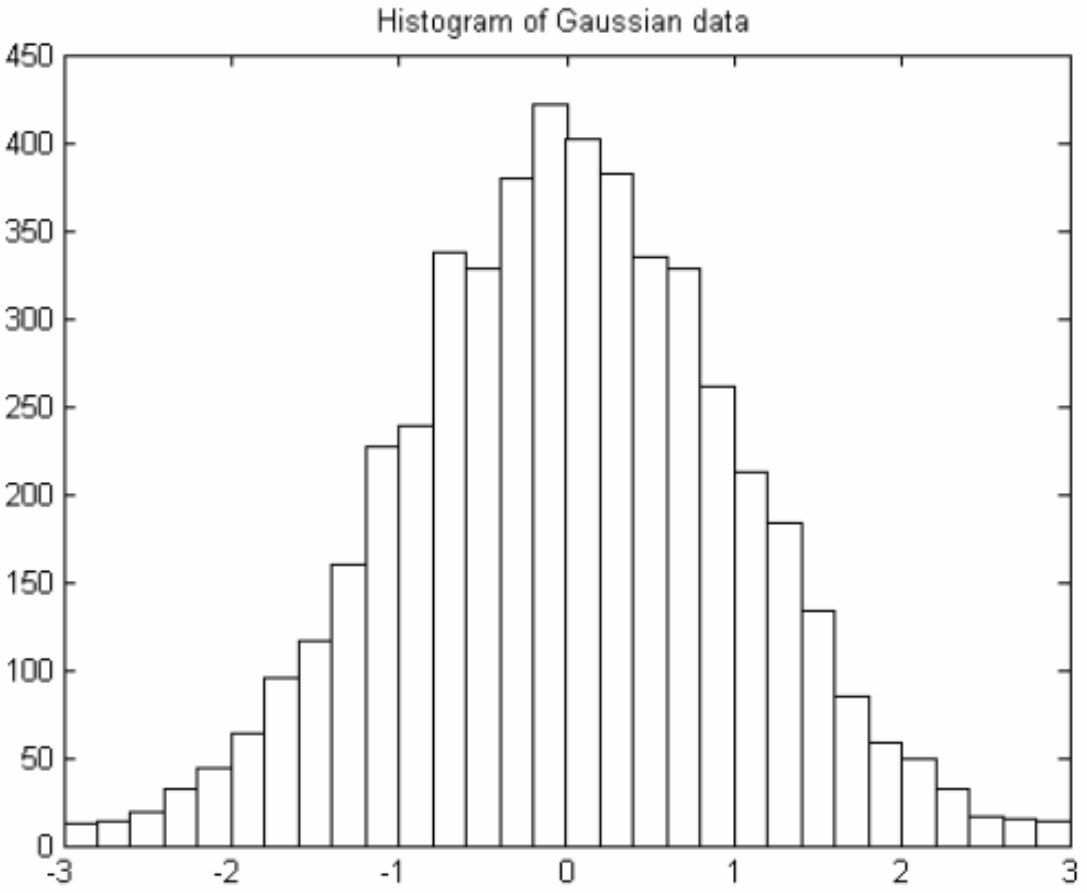
```
» x = -2.9:0.2:2.9;
» y = exp(-x.*x);
» stairs(x,y)
» title('Stairs Plot')
```




**Gambar 1. 6. Penggunaan perintah Staris**

- **hist(y)**

```
» close
» x = -2.9:0.2:2.9;
» y = randn(5000,1);
» hist(y,x)
» title('Histogram of Gaussian data')
```

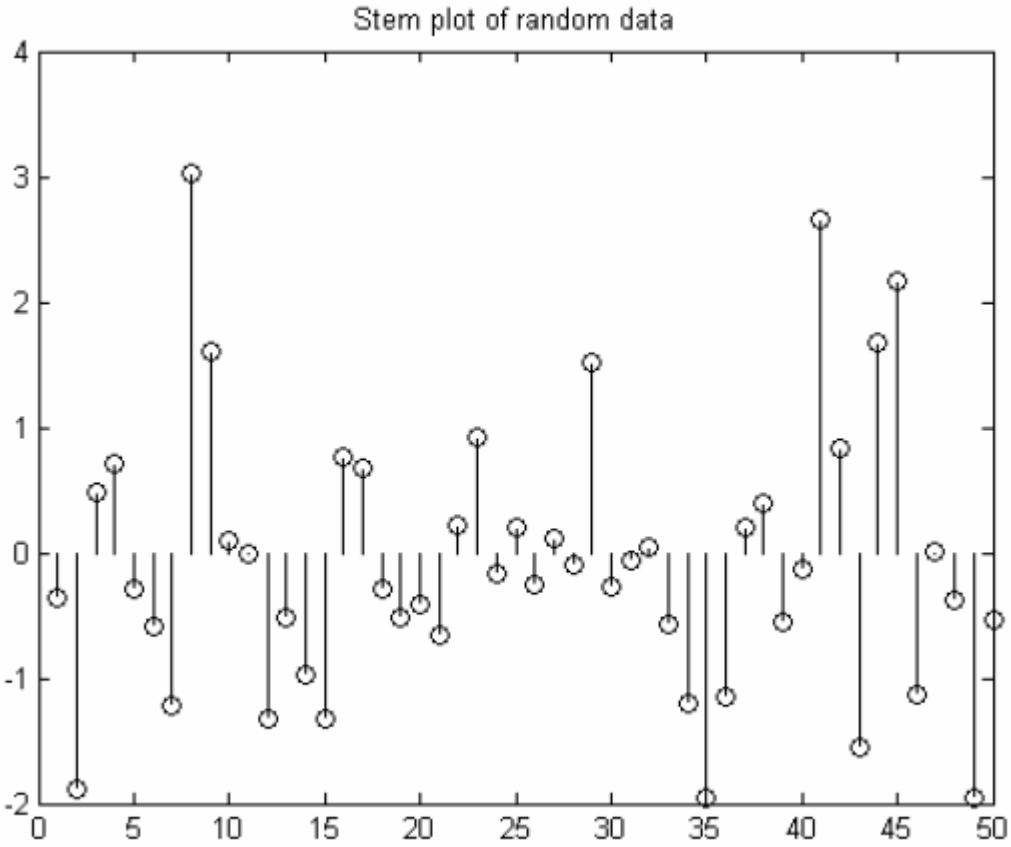



**Gambar 1. 7. Penggunaan perintah hist**

- **Stem (z)**

  Kumpulan data diskrit dapat diplot dengan fungsi ini.

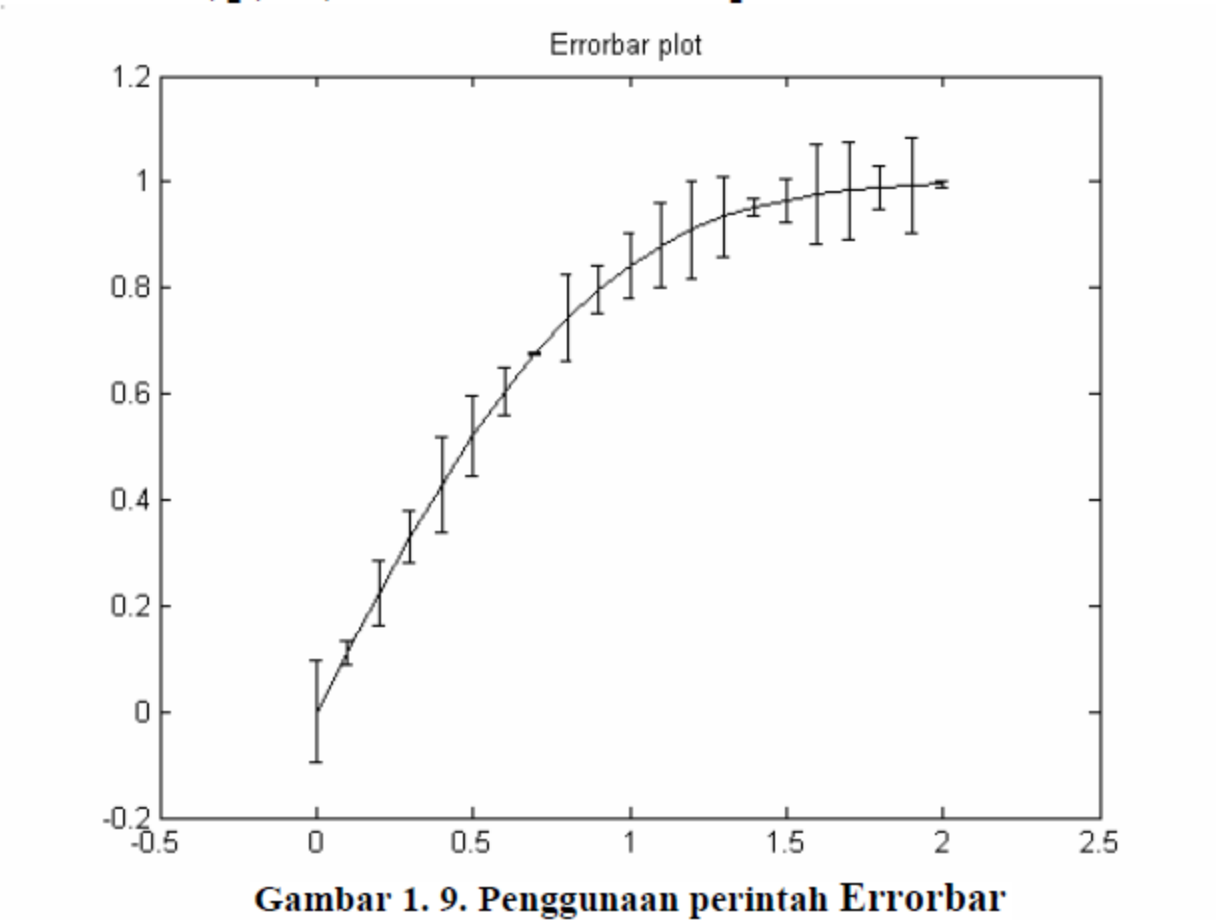
```
» z = randn(50,1);          % membuat data random
» subplot(2,1,1)
» stem(z)                   % membuat grafik stem dengan dotted
linestyle
» title('Stem plot of random data')
```




**Gambar 1. 8. Penggunaan perintah Stem**

- **Errorbar(x,y,e)**

```
» x = linspace(0,2,21);          % membuat vektor
» y = erf(x);                    % y adalah fungsi error dari x
» e = rand(size(x))/10;          % e adalah nilai error dari
bilangan random
» errorbar(x,y,e), title('Errorbar plot')
```

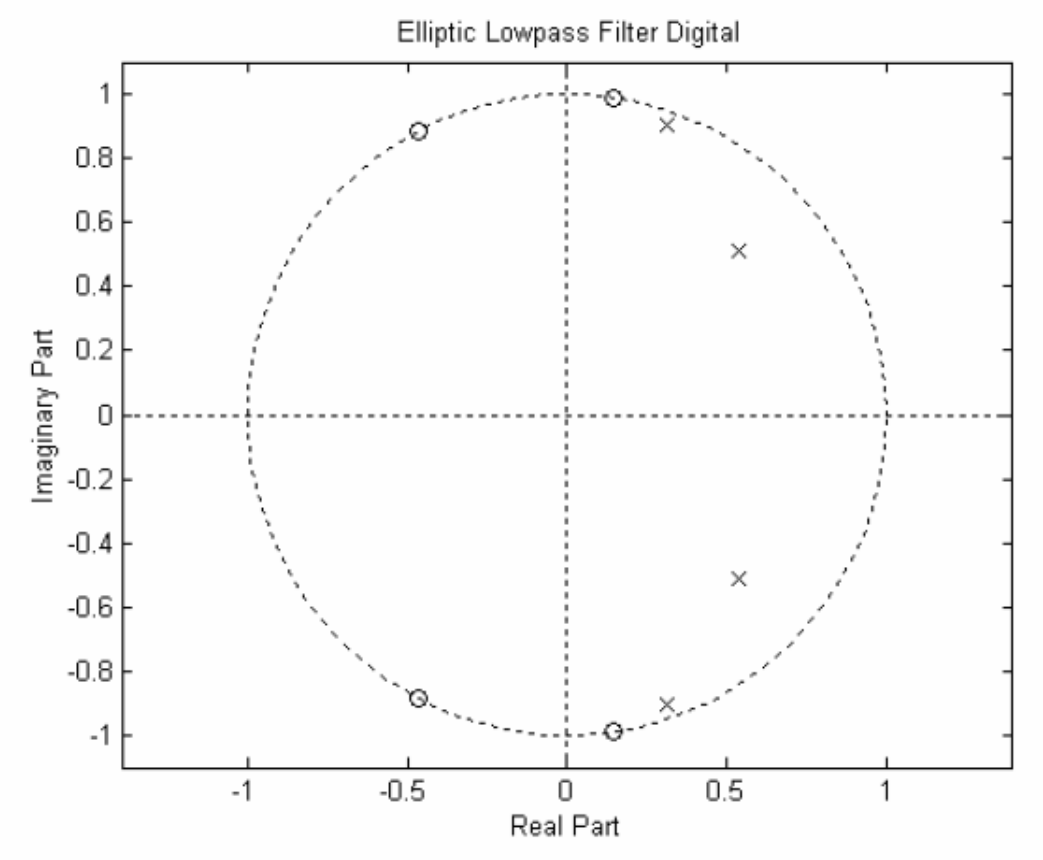



**Gambar 1. 9. Penggunaan perintah Errorbar**

- **Zplane**

  Zplane merupakan perintah untuk mem-plot bagian riil dan imajiner (*poles* dan *zeros*) pada suatu fungsi transfer dengan unit lingkaran sebagai referensi.

```
» [z,p,k] = ellip(4,3,30,200/500);
» zplane(z,p);
» title(' Elliptic Lowpass Filter Digital ');
```

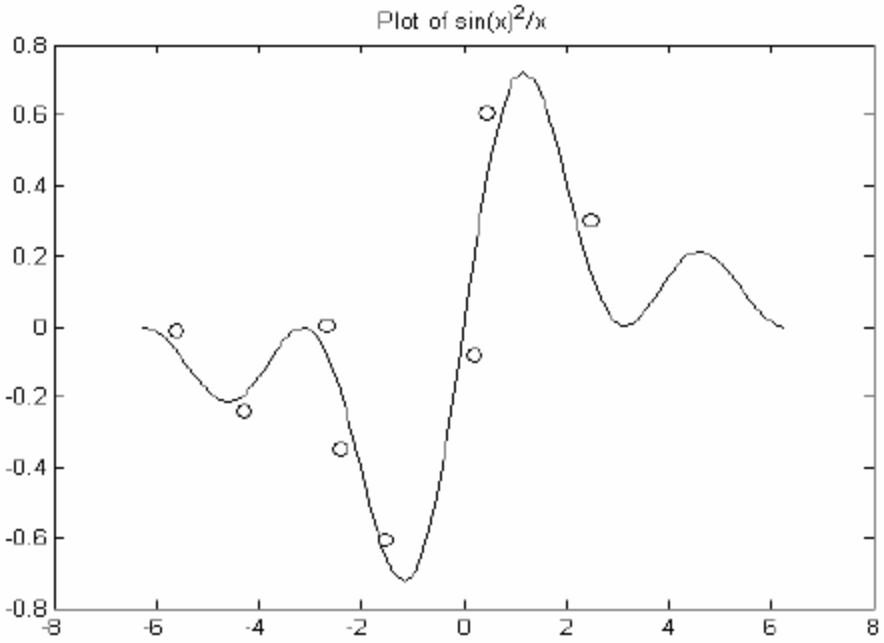



**Gambar 1.10. Penggunaan perintah Zplane**

- **Ginput**

*Ginput* digunakan untuk memasukkan titik yang akan ditambahkan ke dalam suatu grafik dengan mengklik tombol *mouse* kanan. [x,y]=ginput(n) artinya adalah akan ditambahkan data ke dalam grafik di posisi x dan y sebanyak n kali.

```
» x = linspace(-2*pi,2*pi,60);
» y = sin(x).^2./(x+eps);
» plot(x,y), title('Plot of sin(x)^2/x')
» [a,b] = ginput(8);
» hold on
» plot(a,b,'mo')
» hold off
```

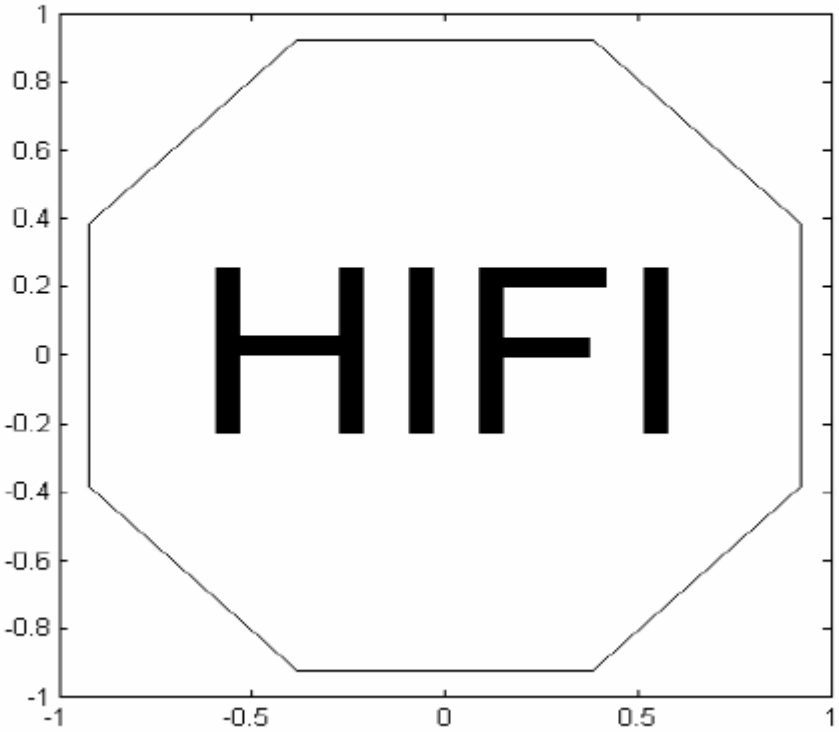



**Gambar 1. 11. Penggunaan perintah Ginput**

- **Fill**

fill(x,y,'c') adalah perintah untuk membuat *polygon* yang didefinisikan oleh vektor x dan y dengan warna yang diberikan oleh c.

```
» t = (1/8:2/8:15/8)'*pi;
» x = sin(t);
» y = cos(t);
» fill(x,y,'r')
» axis square
» text(0,0,'HIFI','Fontsize',96,'HorizontalAlignment','center')
```




**Gambar 1. 12. Penggunaan perintah Fill**

# FILE *.M

## Pendahuluan

Ada dua tipe *file* .m dalam Matlab, yang pertama adalah *file Script* dan yang kedua adalah *file Function*. Perbedaannya adalah bahwa fungsi *file function* merupakan *file* *.m yang diawali dengan kata function dan mempunyai parameter-parameter serta mempunyai tugas tertentu. Pada *file* fungsi setiap variabel yang digunakan akan didefinisikan oleh parameter yang digunakannya. *File* *.m berbentuk *file* teks, sehingga kita bisa menggunakan text editor atau word processor yang lain untuk membuat *file* *.m. Untuk membuat *file* *.m dari jendela Matlab cukup dengan mengklik menu *File* dilanjutkan dengan memilih *New M-File* , setelah itu maka Editor Matlab akan muncul, tinggal kita mengetikkan program yang kita inginkan.

Pada modul sebelumnya biasanya kita mengetikkan perintah di *command line* dalam *command windows* Matlab. Untuk perintah-perintah yang sedikit atau sederhana pengetikkan di *command line* akan lebih praktis, tetapi lain halnya apabila kita sudah berhubungan dengan perintah-perintah yang banyak dan data yang banyak pula, oleh sebab itu dengan *file* *.m kita bisa mengintegerasikan semua perintah-perintah yang kita perlukan dan kita juga bisa merancang fungsi sendiri (*user defined function*).

Beberapa perintah dalam pengelolaan *file* *.m dijelaskan pada Tabel 2.

Beberapa perintah dalam pengelolaan *file* *.m dijelaskan pada Tabel 2.

**Tabel 1. 4. Perintah pengelolaan *file* *.m**

| Perintah | Keterangan |
|---|---|
| What | Melihat semua *file* *.m yang ada di direktori aktif |
| dir atau ls | Melihat semua *file* yang berada di direktori aktif |
| type nama_file | Menampilkan isi *file* nama_file.m |
| delete test | Menghapus *file* nama_file.m |
| cd path | Untuk pindah direktori ke path |
| chdir path | Sama dengan cd path |
| cd, chdir atau pwd | Menampilkan direktori aktif |
| which nama_file | Menampilkan path direktori dari nama_*file**.m |

## File Script

Cara menjalankan *script* cukup dengan mengetikkan nama_file *.m dalam *command line* pada Matlab. Sebagai contoh simpan program di bawah ini dengan nama *file* coba.m setelah itu jalankan dari *command line* Matlab.

```
clear                    % untuk menghapus semua variabel dalam memori
clc                      % untuk menghapus layar
close                    % untuk menutup semua grafik yang aktif
echo on                  % agar semua perintah dalam program ditampilkan
di workspace
x = linspace(0,2*pi);
y = 2*sin(x).*cos(x);
z = 2*cos(x);
plot(x,y,'r-')
hold on
pause                    % tekan sembarang tombol
plot(x,z,'b-')
echo off
hold off
```

Setelah itu jalankan dari *command line program* di atas dengan menuliskan nama *file*-nya.

## Fungsi

Bila suatu *file* *.m didahului dengan kata function berarti *file* *.m tersebut adalah *file* fungsi. Perbedaannya dengan *script* adalah bahwa pada fungsi kita bisa melewatkan argumen dan variabel yang didefinisikan di dalam *file* lokal terhadap fungsi dan tidak bekerja secara global pada *workspace*. Agar variabel dalam fungsi bersifat global, maka cukup dengan menggunakan perintah global.

Fungsi fungsi pada Matlab yang *built-in* banyak sekali contohnya adalah : sin,cos, abs, linspace dan lain-lain. Untuk melihat isi dari fungsi tersebut ketikkan ***type*** nama_*file* pada *command line*. Sebagai contoh kita lihat isi dari fungsi **linspace** :

```
»type linspace
  function y = linspace(d1, d2, n)
  % LINSPACE Linearly spaced vector.
  % LINSPACE(x1, x2) generates a row vector of 100 linearly
  % equally spaced points between x1 and x2.
  %
  % LINSPACE(x1, x2, N) generates N points between x1 and x2.
  %
  % See also LOGSPACE, :.
  % Copyright (c) 1984-97 by The MathWorks, Inc.
  % $Revision: 5.3 $  $Date: 1997/04/08 05:50:40 $

  if nargin == 2
      n = 100;
```

```
end
y = [d1+(0:n-2)*(d2-d1)/(n-1) d2];
```

Sekarang marilah kita mencoba untuk membuat fungsi sendiri.

Ketikkan program di bawah ini dan simpan dengan nama yang sama dengan nama fungsinya, yaitu akar.m

```
function [x1,x2] = akar(a,b,c);
% Fungsi ini bertujuan untuk mencari
% akar-akar persamaan ax^2+bx+c=0
if nargin ~= 3
   error('Parameternya kurang !')
end
d = b^2 - 4*a*c;
x1 = (-b+sqrt(d))/(2*a);
x2 = (-b-sqrt(d))/(2*a);
```

Setelah disimpan kemudian jalankan dengan mengetikkan perintah berikut :

```
»[a,b] = akar(2,1,9)
a =
  -0.2500 + 2.1065i
b =
  -0.2500 - 2.1065i
```

## Control Flow

Dalam Matlab disediakan empat macam *Control Flow* yaitu : *for loops, while loops, if-else-end* dan *switch-case*.

### *for Loops*

*For Loops* adalah sintaks *looping* yang paling umum, hampir di setiap bahasa pemrograman memilikinya. Dalam Matlab sintaksnya adalah sebagai berikut :

```
for x = awal:akhir
      Perintah…
end
```

Contoh :

```
for n = 1:10
      x(n)=sin(n*pi/10)
end
```

*for loops* juga bisa di *nested*-kan seperti contoh berikut :

```
for n = 1:5
      for m = 5:-1:1
            A(n,m) = n^2 + m^2;
      end
end
```

***While loop***

Perbedaan *while loop* dengan *for loops* adalah pada *while loops* biasanya batas akhir pengulanganannya menggunakan logika. Bentuk umum *while loop* adalah :

```
while ekspresi_logika
            Perintah…
end
```

Contoh :

```
Num = 0; eps = 1
while (1+eps) > 1
      eps = eps/2
      num = num+1
end
```

***Struktur If-Else-End***

Bentuk Umum :

```
If ekspresi_logika
      Perintah jika ekspresi_logika dipenuhi…
end
```

atau :

```
if ekspresi_logika
      Perintah jika ekspresi_logika dipenuhi…
else
      Perintah jika ekspresi_logika tidak dipenuhi…
end
```

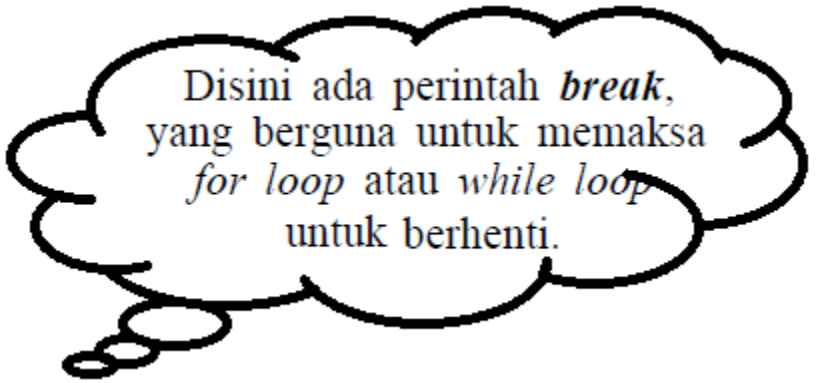
atau :

```
if ekspresi_1
      Perintah jika ekspresi_1 dipenuhi…
elseif ekspresi_2
      Perintah jika ekspresi_2 dipenuhi…
elseif ekspresi_3
      Perintah jika ekspresi_3 dipenuhi…
elseif …
      …
      …
else
      Perintah jika tidak ada ekspresi yang terpenuhi…
end
```

Disini hanya perintah yang berhubungan dengan ekspresi *True* pertama yang ditemui yang akan dijalankan. Sisanya tidak diuji lagi. Kita bisa menggunakan atau pun tidak perintah *else* dalam pemilihan *if* (bergantung pada program yang dibuat).

Contoh :

```
EPS = 1:1000
for num = 1:1000
      EPS = EPS/2;
      if(1+EPS) <= 1
            EPS = EPS*2
            break
      end
end
num
```



***Struktur Switch-Case***

Bentuk Umum :

```
switch ekpresi_switch
      case ekspresi_1
            Perintah_1
      case ekspresi_2
            Perintah_2
            ...
      otherwise
            Perintah
End
```

Sebenarnya perintah *switch-case* ini hampir sama dengan perintah *if-else-end*, hanya saja disini semua *case*-nya diuji.

Contoh:

```
metode = 'Bilinear';
switch lower(metode)
                case {'linear','bilinear'}
                disp('Metode Linear')
                case 'cubic'
                disp('Metode Kubik')
                case 'nearest'
                disp('Metode Nearest')
                otherwise
                disp('Metode Lainnya.')
end
```

## *Operasi Relasi dan Logika*

Sebagai input untuk semua ekspresi relasi dan logika, Matlab menganggap semua angka *non zero* sebagai *True* dan *zero* sebagai *False*. Output dari semua ekspresi relasi dan logika menghasilkan 1 untuk *True* dan 0 untuk *False*.

### Operasi Relasi

Operator relasi Matlab termasuk semua perbandingan umum :

| | |
|---|---|
| < | kurang dari |
| <= | kurang dari atau sama dengan |
| > | lebih besar dari |
| >= | lebih besar dari atau sama dengan |
| == | sama dengan |
| ~= | tidak sama dengan |

### *Operasi logika*

Operasi logika digunakan untuk menggabungkan atau menegasikan ekspresi relasional. Diantaranya :

| | |
|---|---|
| **&** | AND |
| \| | OR |
| ~ | NOT |

Beberapa contoh penggunaan operator logika:

```
A = 1:9;
B = 9 - A;
Tf = A>4
Tf = ~(A>4)
Tf = (A>2)&(A<6)
```

Kemampuan-kemampuan ini memudahkan kita untuk menghasilkan array yang merepresentasikan signal dengan ketidak-kontinuan atau signal yang terbuat dari segmen-segmen dari signal-signal lain. Ide dasarnya ialah dengan mengalikan nilai-nilai pada array yang kita inginkan dengan satu, dan mengalikan yang lainnya dengan nol. Misalnya :

```
x = linspace(0,10,100);
y = sin(x)
z = (y>=0).*y;
z = z + 0,5*(y<0);
z = (x<=8).*z;
plot(x,y)
xlabel('x'),ylabel('z=f(x)')
title('A Discontinuous Signal')
```

Ketikkan program di bawah ini, kemudian simpan dengan nama **coba.m**

```
% Program untuk mencari akar-akar persamaan
% dengan metoda Bisection
% by Fiskom 2000.

clc
clear
close

x1 = input('Input nilai x1 : ');
x2 = input('Input nilai x2 : ');
tol = 0.0001;
iter = 0;

fprintf('iterasi   x1      x2       x3         f(x3)\n')
while abs(x1 - x2) > tol
      iter = iter + 1;
            x3 = (x1+x2)/2;
            fprintf('%3d.6f%.6f%.6f%.6f%.6f%.6f\n'\n',iter,x1,x2,x3)
            if ((fung(x3)*fung(x1)) < 0)
                        x2 = x3;
            else
                        x1 = x3;
      end
end
fprintf('\nAkarnya adalah = %.8f\n',x3)
```

Setelah itu buatlah *file* **fung.m** yang berisi bentuk fungsi dari program di atas.

```
function fx = fung(x)
fx = x^3 + x^2 - 3*x - 3;
```

Jalankan program di atas setelah itu ubahlah bentuk fungsi dari **fung.m**.

## MODUL LATIHAN MATLAB DASAR

**Keterangan :**

1. Simpan seluruh pekerjaan anda pada suatu *file* *.txt dengan menuliskan perintah **diary**.
2. Simpan seluruh variable lengkap dengan datanya pada suatu *file* *.mat dengan menuliskan perintah **save nama_file.**

## LATIHAN I

1. bagian I

```
» what
» a = 2 + 3
» b = a + 5
» who
» whos
» save simpan
» clear
» load simpan
» a,b
» whos
```

2. Bagian II

```
» x = [ 1 2 3 ; 4 5 6]
» y = [1 2 ; 4 3 ; 7 8]
» u = rand (4)
» x (2,1)
» x(2,1) = 0
» o = x.^2
» xt = x'
» z = xt+y
» p = xt-y
» q = x*y
» u = inv (q)
» p = [1 j ; -j*5 2]
» u = conj (p)
```

3. Bagian III

```
» x = linspace (0,2*pi)
» y = sin (x)
» z = cos (x)
» simplus = max (y)
» simin = min (y)
» x (5:10)
» y (50:end)
» p = [x(5:10); y(10:15)]
» a = 0 : 1 : 20
» b = a * a'
```

4. Bagian IV

```
» t = 0:0.5:10;
» y = sin (t);
» z = cos (t);
» plot (t,y,'o',t,z,'x')
» grid
» title ('Kurva Sinus Dan Cosinus')
» xlabel ('waktu (detik)')
» ylabel ('Amplitudo')
» text (3,0.45,'sin t')
» text (0.8,-0.3,'cost')
```

5. Bagian V

```
» x = 0 : 0.1 :3;
» y = x.^2;
» plot (x,y)
» grid
» title ('Kurva parabolic y=x^2')
» xlabel ('Sumbu X')
» ylabel ('Sumbu Y')
```

## LATIHAN II

1. Disajikan persamaan bilangan kompleks berikut :

$$a = 3 + 4i$$
$$b = 2 + 5i$$

cari magnitude (M), sudut fasa Θ, nilai real dan imaginer dari persamaan tersebut menggunakan Matlab. Buktikan bahwa hasil solusi dari bilangan kompleks sama dengan hasil dari persamaan trigonometri (metoda Euler). Cari pula hasil perhitungan dibawah ini :

   a. c = a + b
   b. d = a * b
   c. e = a /b

2. x = linspace(0,4*pi,200)
   y = sin (x)
   z = 2 + cos (x)
   tampilkan kurva yang diperoleh dari persamaan matematis diatas :

3. Buat matrik linear dengan elemen matrik dari 0 sampai 1 dengan *step* 0.1
   X = 2 x sin (ωt) ; ω = 2πf ; f = 5 Hz.
   Tampilkan kurva persamaan diatas dengan skala waktu yang sesuai.

4. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $B = \begin{bmatrix} 4 & 2 & 1 \\ 3 & 9 & 6 \\ 3 & 3 & 0 \end{bmatrix}$

a. Hitung Determinan dari matrik A dan B.
b. hitung inversi dari matrik A dan B ( $A^{-1}$ dan $B^{-1}$).
c. Buktikan bahwa $A^{-1} \times A \neq I$ dan $B^{-1} \times B = I$ . Jelaskan dengan Matlab mengapa matrik A tidak memiliki matrik identitas sedangkan matrik B memiliki matrik identitas.

5. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $B = \begin{bmatrix} 4 & 2 & 1 \\ 3 & 9 & 6 \\ 3 & 3 & 0 \end{bmatrix}$

a. Hitung perkalian matrik A x B
b. Hitung perkalian elemen matrik A dan B
c. Ubah matrik A(2,1) = 0
d. Ambil matrik A elemen baris 1 dan 2 untuk semua kolom, simpan di matrik C. Ambil matrik B elemen kolom 2 dan 3 untuk semua baris, simpan di matrik D
e. Hitung $C \times D^T$.
f. Ubah matrik A menjadi matrik A 1-dimensi, simpan di matrik E

6. $G = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

   a. Ubah Matrik G tersebut menjadi matrik identitas dan simpan pada variabel I.
   b. Buatlah matrik G menjadi matrik diagonal dengan isi elemen bernilai 4

7. Buatlah deret Fibonachi dengan menggunakan *control flow*. Deret Fibonachi tersusun oleh sederetan angka sebagai berikut :

   Fb = 1 1 2 3 5 8 … n

8. Buatlah Matrik di bawah dengan menggunakan *Looping* dan Logika Matematika, selanjutnya *plot* kurva yang didapatkan.
   a. $X_1$ = [ 1 2 4 8 16 32 64 128]
   b. $X_3$ = [ 1 3 5 7 9 … ] dan $X_4$ = [ 2 4 6 8 10 … ] dalam satu Program
   c. $X_5$ = [ 1 1 0 1 1 0 1 1 0 … ]

9. Buatlah matrik acak 1-dimensi yang bernilai kompleks, lalu plot komponen real, imajinernya, dan z-plane-nya.

10. Buatlah program solusi factorial. Factorial diselesaikan dengan persamaan : $Y = x!$

11. Buatlah program function untuk menentukan akar2 dari suatu persamaan kuadrat!