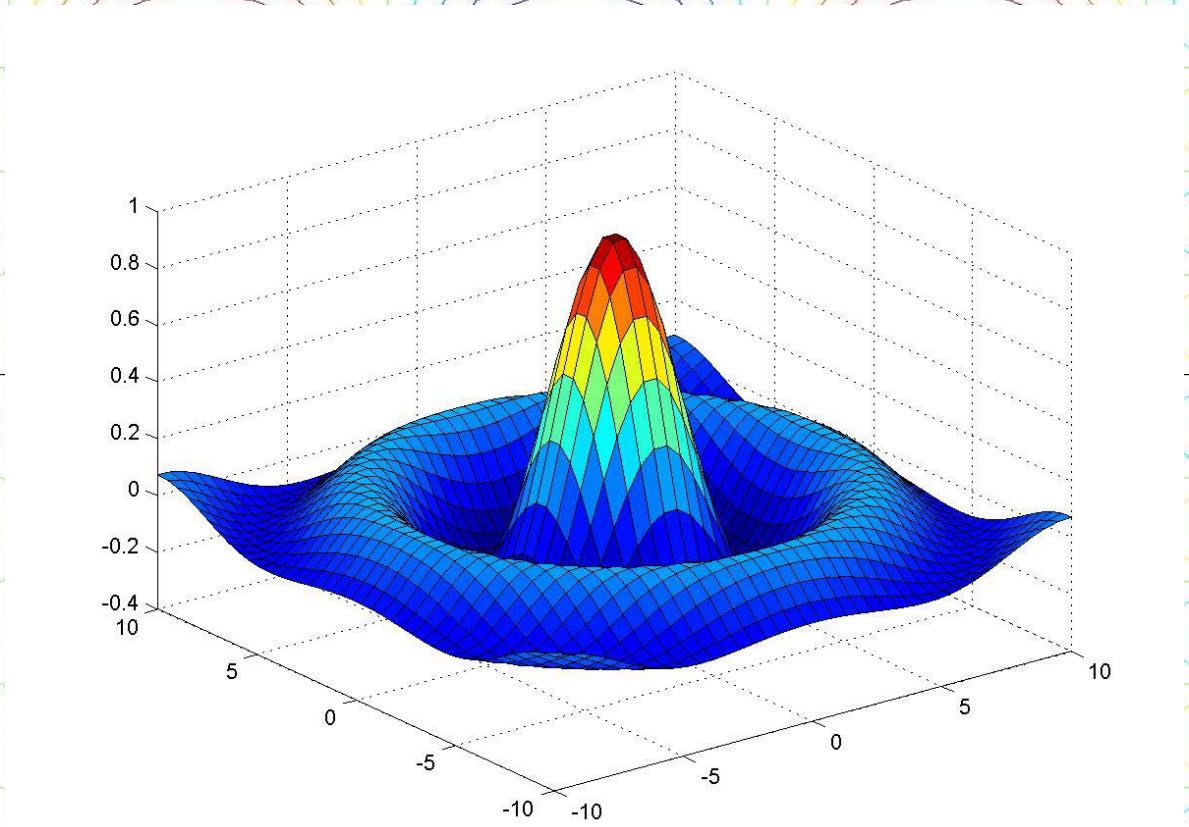
TUTORIAL PRAKTIS

BELAJAR MATLAB



Teguh Widiarsono, M.T.

TUTORIAL PRAKTIS BELAJAR MATLAB

Teguh Widiarsono, M.T.

KATA PENGANTAR

Pertama-tama, penulis bersyukur kepada Allah SWT, karena hanya dengan limpahan rahmat dan karunia-Nya penulis bisa menyelesaikan buku tutorial ini.

Buku ini membahas tutorial penggunaan MATLAB secara praktis bagi pengguna mula ataupun yang sudah familiar. Pembahasan dimulai dengan pengenalan variabel, matriks, serta fungsi yang lazim ditemui dalam kasus perhitungan sehari-hari. Berikutnya dikenalkan teknik grafis 2 dan 3-dimensi, kemudian pemrograman MATLAB sehingga pengguna bisa mendefinisikan fungsi sendiri. Pada bagian akhir dibahas topik-topik yang lebih khusus meliputi: analisis data, statistika, polinomial, analisis fungsi, serta perhitungan integral.

Lebih dari 200 contoh dan soal latihan disajikan dalam buku ini, meliputi: perhitungan, program, dan *command* MATLAB yang ada pada setiap bab; sehingga akan mempermudah pemahaman sekaligus bisa digunakan sebagai rujukan yang bermanfaat.

Mahasiswa tingkat awal hingga akhir bisa memanfaatkan berbagai kemampuan MATLAB untuk menyelesaikan perhitungan rumit yang kerap ditemui dalam kuliah, atapun membuat simulasi untuk skripsi / tugas akhir.

Penulis menyampaikan rasa terima kasih dan penghargaan setinggi-tingginya kepada keluarga dan rekan-rekan yang telah mendorong penulis untuk menyelesaikan buku ini; dan juga kepada rekan-rekan yang turut menyebarkan buku ini secara cuma-cuma dalam bentuk *softcopy* “e-book” ataupun *hardcopy*.

Penulis sangat mengharapkan kritik dan saran dari para pembaca untuk memperbaiki kualitas buku ini. Penulis berharap buku ini akan bermanfaat bagi banyak pihak, aamiin.

Jakarta,
Ramadhan 1426 / Oktober 2005

Buku ini kupersembahkan untuk istri tercinta, Anna Nurul Inayati Shofia, dan anakku yang sholeh Faska Ulul 'Azmi Mir. Juga kepada Widjayanto (EL2000) dan Mas Teguh Prakoso (EL96) yang turut mendorong dan menyebarluaskan buku ini.

DAFTAR ISI

BAB 1

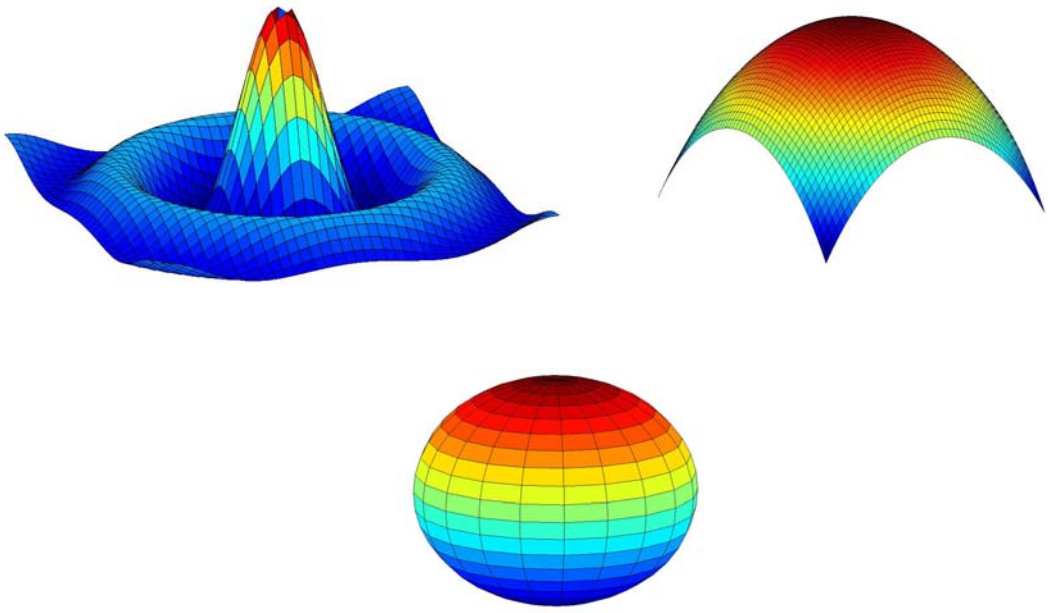
APA ITU MATLAB?

MATLAB merupakan suatu program komputer yang bisa membantu memecahkan berbagai masalah matematis yang kerap kita temui dalam bidang teknis. Kita bisa memanfaatkan kemampuan MATLAB untuk menemukan solusi dari berbagai masalah numerik secara cepat, mulai hal yang paling dasar, misalkan sistem 2 persamaan dengan 2 variabel:

$$x - 2y = 32$$
$$12x + 5y = 12$$

hingga yang kompleks, seperti mencari akar-akar polinomial, interpolasi dari sejumlah data, perhitungan dengan matriks, pengolahan sinyal, dan metoda numerik.

Salah satu aspek yang sangat berguna dari MATLAB ialah kemampuannya untuk menggambarkan berbagai jenis grafik, sehingga kita bisa memvisualisasikan data dan fungsi yang kompleks. Sebagai contoh, tiga gambar berikut diciptakan dengan *command* **surf** di MATLAB.

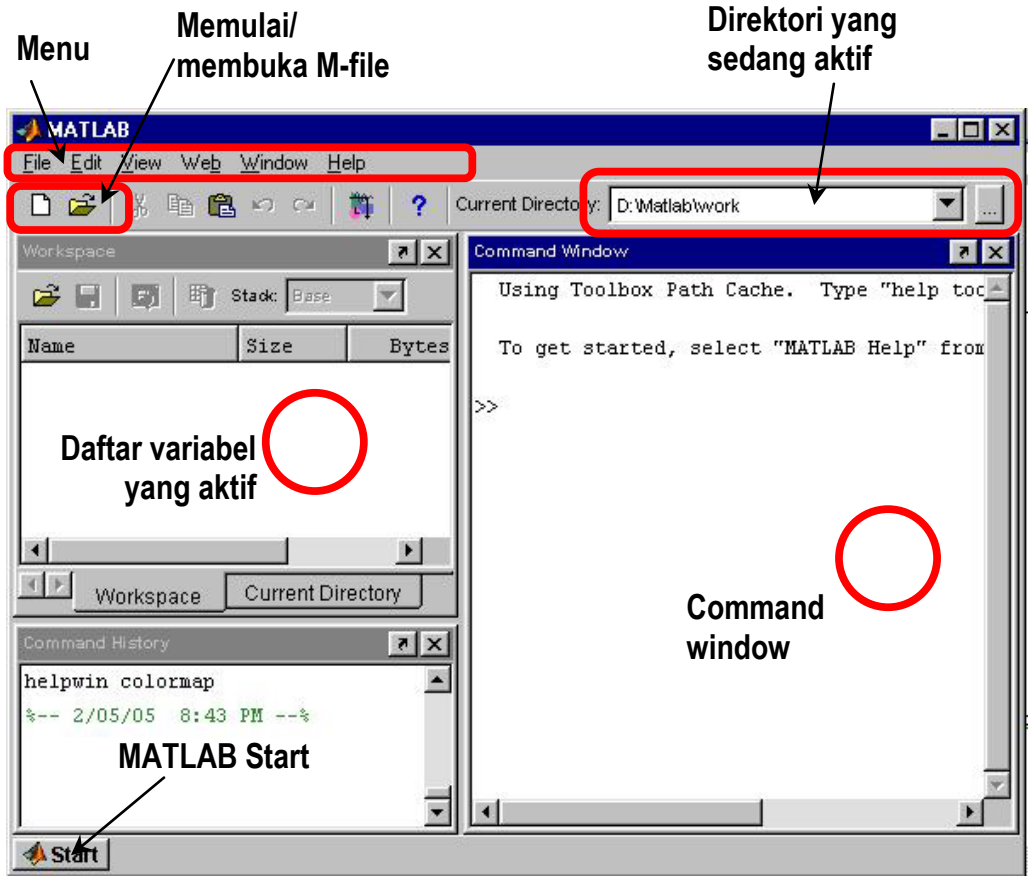


Gambar 1. 1 Grafik 3-dimensi diciptakan dengan *command* "surf" di MATLAB.

Dalam buku ini kita akan mempelajari MATLAB setahap demi setahap, mulai dari hal yang sederhana hingga yang cukup kompleks. Yang perlu kita persiapkan untuk belajar MATLAB ialah seperangkat komputer yang sudah terinstal program MATLAB di dalamnya. Kita bisa gunakan MATLAB versi 5, 6 ataupun 7 untuk mempraktekkan berbagai contoh yang ada di buku ini. Di dalam buku ini kita akan mempelajari 'teori' penggunaan MATLAB, namun untuk menjadi mahir Anda harus duduk di depan komputer dan mempraktekkannya secara langsung!

1.1 Memulai MATLAB

Kita memulai MATLAB dengan mengeksekusi ikon MATLAB di layar komputer ataupun melalui tombol **Start** di Windows. Setelah proses *loading* program, jendela utama MATLAB akan muncul seperti berikut ini.



Gambar 1. 2 Jendela utama MATLAB.

Setelah proses loading usai, akan muncul *command prompt* di dalam *command window*:

```
>>
```

Dari *prompt* inilah kita bisa mengetikkan berbagai *command* MATLAB, seperti halnya *command prompt* di dalam DOS.

Sebagai permulaan, mari kita ketikkan *command* **date** :

```
>> date
```

setelah menekan Enter, akan muncul

```
ans =
05-Feb-2005
```

date adalah *command* MATLAB untuk menampilkan tanggal hari ini. Berikutnya cobalah *command* **clc** untuk membersihkan *command window*:

```
>> clc
```

Ketika kita selesai dengan sesi MATLAB dan ingin keluar, gunakan *command* **exit** atau **quit**.

```
>> exit
```
Atau...
```
>> quit
```

Atau bisa juga dengan menggunakan menu:
File → Exit MATLAB.

1.2 Mencoba Kemampuan MATLAB

Jika Anda baru pertama kali menggunakan MATLAB, ada baiknya kita mencoba beberapa *command* untuk melihat sepintas berbagai kemampuan dan keunggulan MATLAB.

MATLAB dapat kita pergunakan seperti halnya kalkulator:

```
>> 2048 + 16
ans =
       2064
```

Menuliskan beberapa *command* sekaligus dalam satu baris:

```
>> 5^2, 2*(6 + (-3))
```

```
ans =
       25
ans =
       6
```

Menciptakan variabel untuk menyimpan bilangan, serta menjalankan berbagai *command* atau fungsi yang sudah ada di MATLAB.

```
>> x=12; y=0.25; z=pi/2;
>> a=3*x*y, b=sin(z), c=cos(z)
a =
       9
b =
       1
c =
       0
```

Menciptakan dan memanipulasi vektor dan matriks:

```
>> Vektor1=[1 3 -6], Vektor2=[4; 3; -1]
Vektor1 =
       1    3   -6
Vektor2 =
       4
       3
      -1

>> Matrix=[1 2 3;4 5 6;7 8 9]
Matrix =
     1     2     3
     4     5     6
     7     8     9

>> Vektor1 * Vektor2
ans =
    19

>> Vektor2 * Vektor1
ans =
     4    12   -24
     3     9   -18
    -1    -3     6

>> Matrix * Vektor2
ans =
     7
    25
    43
```

Menciptakan deret secara efisien:

```
>> deret1=1:1:10
deret1 =
1     2     3     4     5     6     7     8     9   10

>> deret2=linspace(0,5,11)
deret2 =
Columns 1 through 7
0   0.5000   1.0000   1.5000   2.0000   2.5000  3.0000

Columns 8 through 11
3.5000   4.0000   4.5000   5.0000
```

MATLAB juga dapat kita pergunakan untuk mencari akar-akar polinomial. Misalkan akar-akar dari:

$$y = x^4 - 10\,x^2 + 9$$

```
>> akar=roots([1 0 -10 0 9])
akar =
    3.0000
   -3.0000
    1.0000
   -1.0000
```

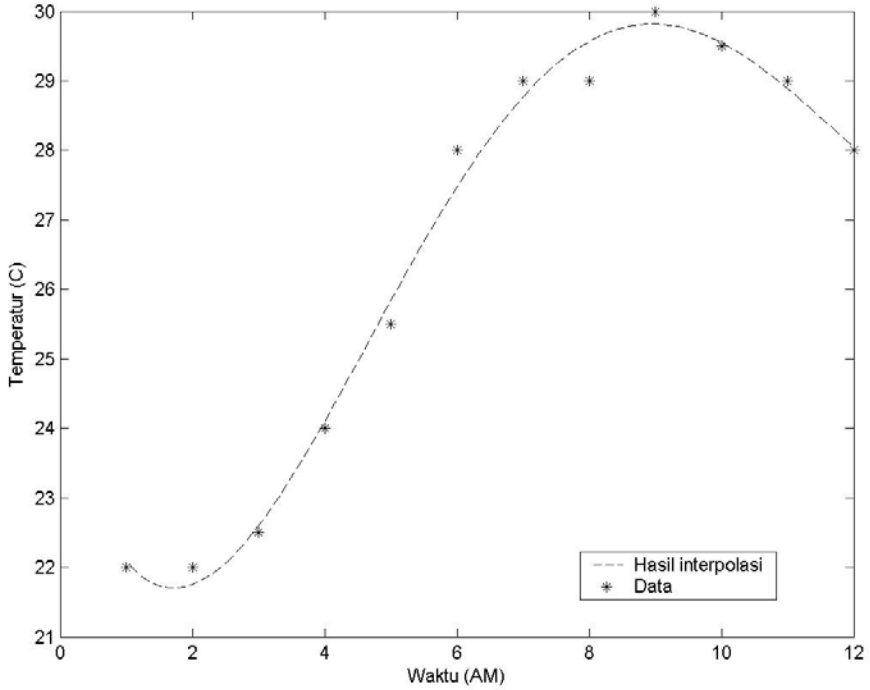
Melakukan interpolasi dengan berbagai metode, misalkan dengan pendekatan polinomial.

Misalkan kita memiliki data pengamatan temperatur selama 12 jam:

```
>> t=1:12;
>> data=[22 22 22.5 24 25.5 28 29 29 30 29.5 29 28];
```

Data tersebut kita interpolasi menjadi kurva mulus polinomial orde-5:

```
>> p=polyfit(t,data,5);
>> x=linspace(1,12,100); y=polyval(p,x);
>> plot(x,y,'k--',t,data,'k*')
>> p
p =
0.0000    0.0038   -0.1245    1.2396   -3.2370 24.2045
```



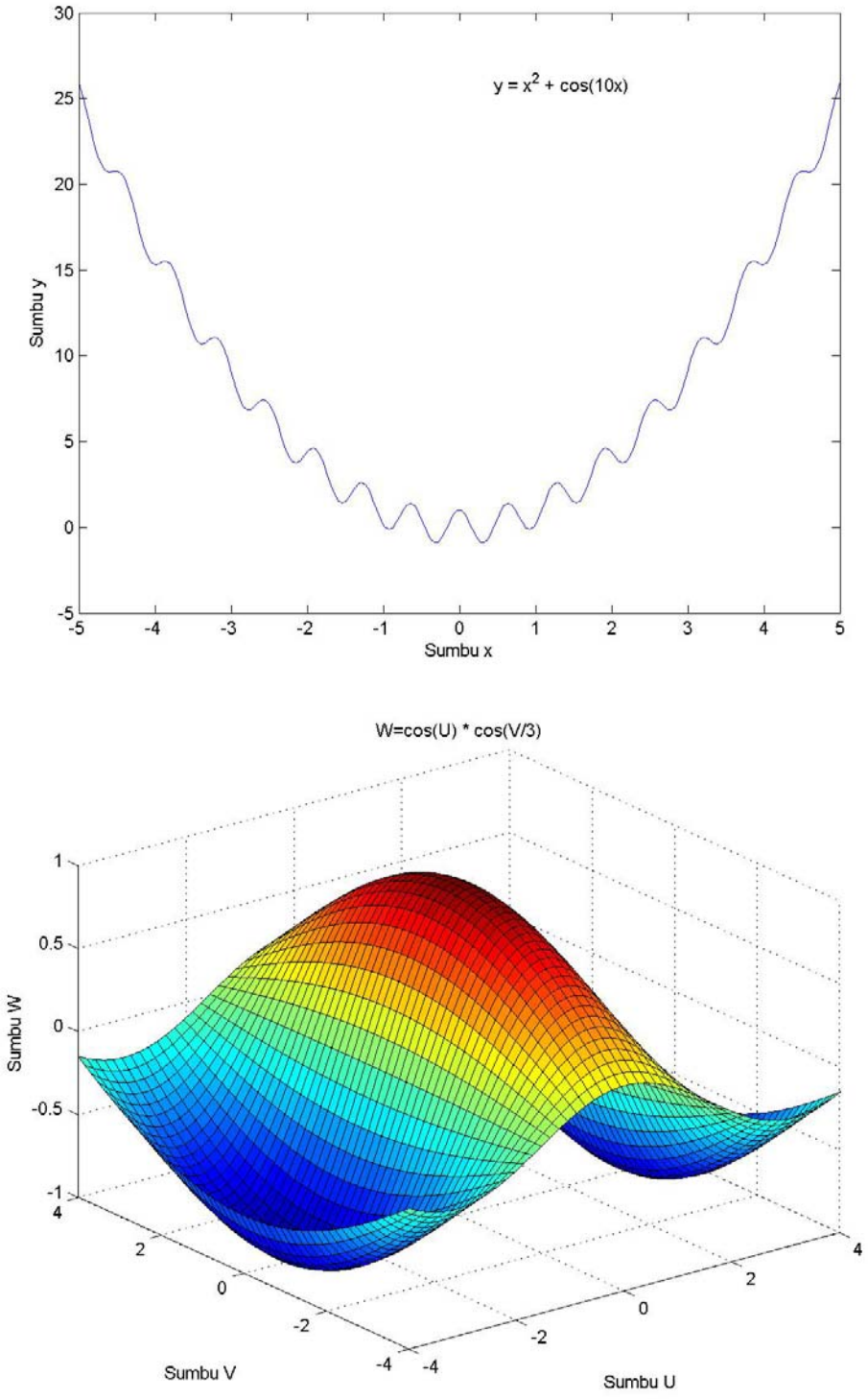
Gambar 1. 3 Interpolasi data temperatur terhadap waktu, didekati dengan polinom $y = 0{,}038\ x^4 - 0{,}1245\ x^3 + 1{,}2396\ x^2 - 3{,}237\ x + 24{,}2045$

Salah satu keunggulan MATLAB ialah kemudahannya untuk membuat grafik dan suara. Misalkan membuat grafik 2-dimensi,

```
>> x=linspace(-5,5,200);
>> y=x.^2+cos(10*x);
>> plot(x,y)
```

atau bahkan grafik 3-dimensi:

```
>> u=linspace(-4,4,50);
>> [U,V]=meshgrid(u,u);
>> W=cos(U).*cos(V/3);
>> surf(U,V,W)
```



Gambar 1. 4 Grafik 2 dan 3-dimensi diciptakan dengan *command* plot dan surf.

Dan juga membuat suara, misalkan nada DO, RE, MI:

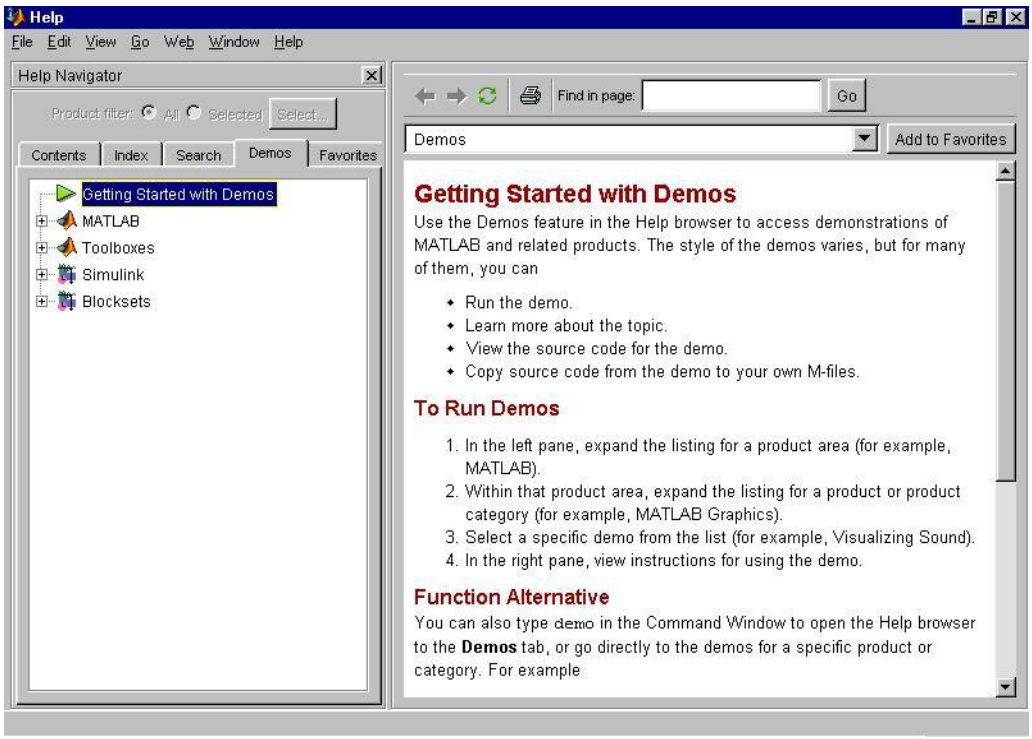
```
>> Fs=8000;                    %Frekuensi sampling 8 kHz
>> t=0:1/Fs:0.5;               %Durasi nada 1/2 detik
```

```
>> frek=[262 294 330];     %Frekuensi DO RE MI
>> m=[];

>> for i=1:3
m=[m cos(2*pi*frek(i)*t)]; %Membuat vektor DO RE MI
end

>> sound(m,Fs)
```

Penjelasan dan langkah-langkah yang detail mengenai berbagai contoh di atas akan kita pelajari dalam bab-bab berikutnya dari buku ini.

1.3 Demo di MATLAB

Ketika sudah membuka MATLAB, kita bisa menjalankan demo yang ada di dalamnya. Dari *command window* ketiklah `demo`, maka akan muncul jendela *browser* di mana kita bisa memilih demo mana yang akan dijalankan.

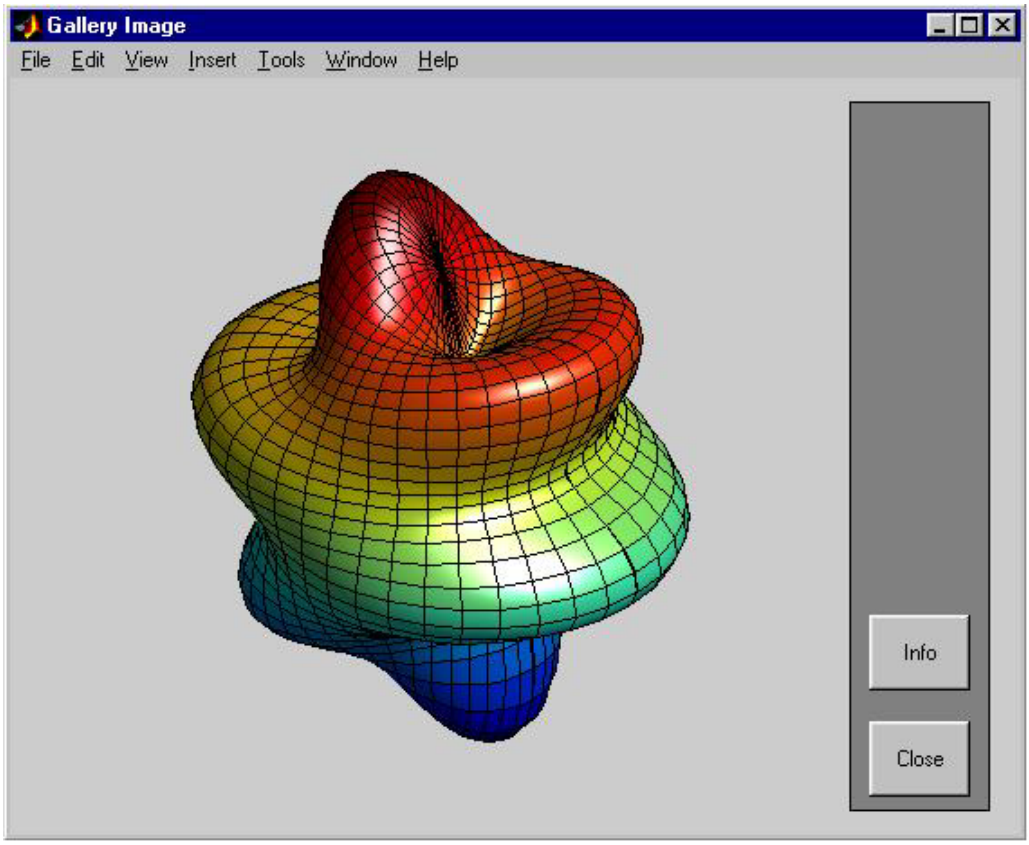


Gambar 1. 5 Jendela tempat memulai demo.

Kita bisa melihat dan merasakan berbagai aplikasi dari MATLAB dengan cara mengeksplorasi demo. Di dalam demo tersebut terdapat beberapa *game* yang bisa kita mainkan, grafik-grafik yang

menarik, dan sejumlah simulasi dari berbagai bidang teknik.

Kita bisa mengekspansi folder MATLAB (klik tanda +) dan melihat berbagai kategori demo. Misalkan kita memilih **Gallery → Slosh**, lalu coba jalankan; maka akan muncul grafik berikut.



Gambar 1. 6 Salah satu gambar di dalam galeri demo

Demo ini memperlihatkan betapa efek grafis 3-dimensi yang bagus bisa dibuat dengan MATLAB. Sekarang, nikmati waktu Anda dengan menjalankan berbagai demo yang lain!

1.4 Mendapatkan Help

MATLAB memiliki sistem "help" yang ekstensif, memuat dokumentasi detil dan informasi "help" meliputi semua *command* dan fungsi di MATLAB. Sistem ini akan sangat membantu kita, baik yang pemula maupun ahli, untuk memahami fungsionalitas MATLAB yang belum pernah kita gunakan sebelumnya. Untuk mendapatkan *help*, terdapat 2 cara: melalui *command window*, dan melalui *help browser*.

1.4.1 Mendapatkan Help dari Command Window

Dari *command window*, kita bisa gunakan: **help**, **helpwin**, dan **doc**. Misalkan kita ingin mengetahui deskripsi dari *command* **plot**.

```
>> help plot
PLOT   Linear plot.
PLOT(X,Y) plots vector Y versus vector X. If X or Y is
a matrix, then the vector is plotted versus the rows
or columns of the matrix, whichever line up.  If X is
a scalar and Y is a vector, length(Y) disconnected
points are plotted.
       ....
       ....
See also SEMILOGX, SEMILOGY, LOGLOG, PLOTYY, GRID,
CLF, CLC, TITLE, XLABEL, YLABEL, AXIS, AXES, HOLD,
COLORDEF, LEGEND, SUBPLOT, STEM.
```

Output dari **help** juga merujuk ke *command* lain yang berhubungan. Dalam contoh ini: **semilogx**, **semilogy**, **loglog**, dan seterusnya. Untuk melihat deskripsinya bisa kita ketikkan `help semilogx`, `help loglog`, dan sebagainya.

Penting!

Nama fungsi atau *command* di dalam *help* ditampilkan dengan huruf kapital, tetapi ketika kita ketikkan di *command window* harus menggunakan huruf kecil.
Contohnya dalam **help plot** di atas, tertulis `PLOT(X,Y)`, tetapi ketika kita gunakan harus ditulis `plot(x,y)`

Dari *command window* Anda juga bisa menggunakan **helpwin**.

```
>> helpwin plot
```

Akan muncul *window* yang berisi deskripsi tentang fungsi atau *command* yang dimaksud.

Terlihat bahwa **help** ataupun **helpwin** menampilkan informasi yang sama, namun demikian terdapat kelebihan **helpwin**:

- Teks ditampilkan di *window* yang terpisah dengan *command window*
- Kita bisa langsung mengklik fungsi di "See also" untuk referensi, jadi tidak usah mengetik lagi lewat *command window*.
- Terdapat *link* **Default Topics** yang berisi daftar semua kategori fungsi MATLAB, sehingga kita bisa mengetahui semua fungsi yang terdapat dalam suatu kategori. Misalkan kita ingin mengetahui fungsi apa saja untuk plot grafik 2-dimensi, maka pilihlah *link* **matlab\graph2d**.

Cara yang lain untuk mendapatkan dokumentasi yang lengkap ialah menggunakan **doc**.

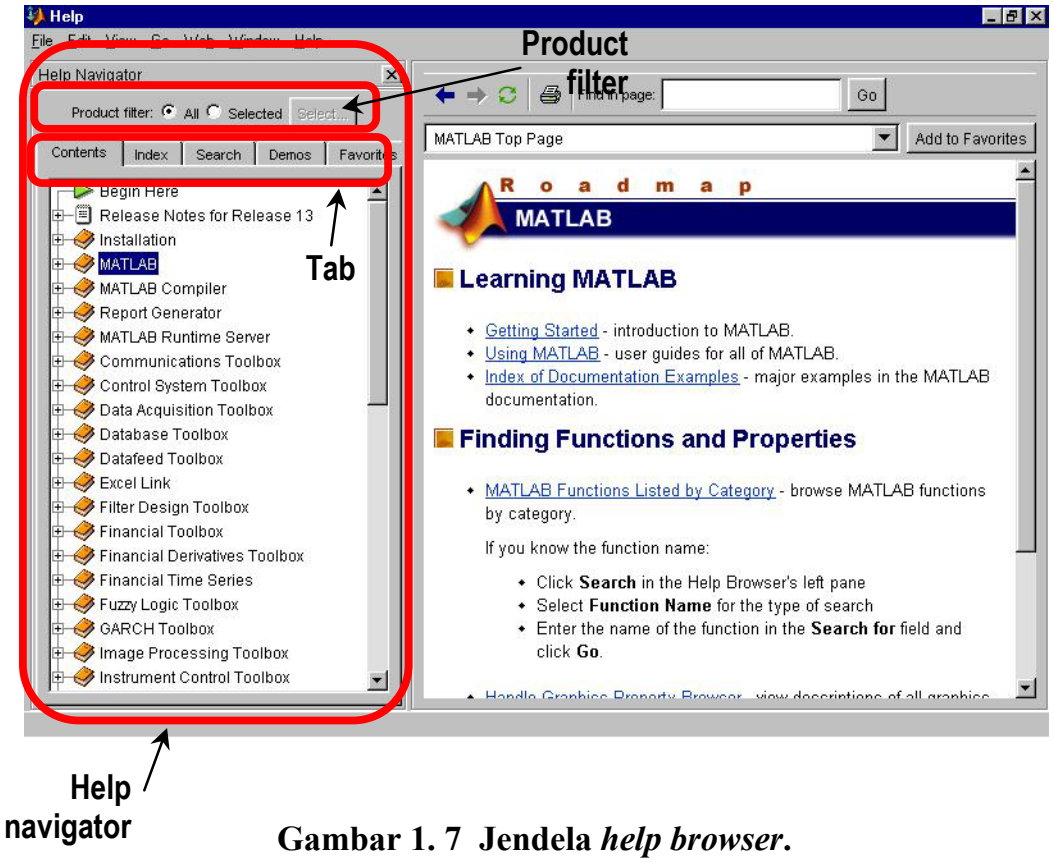
```
>> doc plot
```

Keluaran *command* **doc** inilah yang paling lengkap, bahkan menyediakan contoh lengkap yang bisa dipelajari dan dieksekusi.

Sekarang cobalah Anda lihat *help* untuk *command* lainnya: **plot3**, **polyfit**, dan **trapz.**

1.4.2 Mendapatkan Help dari Help Browser

Sumber *help* lainnya ialah *help browser*. Anda bisa mengetikkan **helpbrowser** di *command window*, atau dari menu
Help → MATLAB Help.



Gambar 1. 7 Jendela *help browser*.

Help browser memiliki dua bagian utama: **Help Navigator**, dan layar tampilan di sisi kanan. Cara penggunaan *help browser* mirip dengan Windows Explorer; apa yang kita pilih di daftar navigator akan ditampilkan di layar sisi kanan. **Help Navigator** ini memiliki sejumlah komponen:

- **Product filter** : mengaktifkan filter untuk memperlihatkan dokumentasi hanya pada produk yang Anda inginkan
- Tab **Contents** : melihat judul dan daftar isi dokumentasi
- Tab **Index** : mencari entri indeks tertentu (dengan kata kunci) di dalam dokumentasi
- Tab **Demos** : melihat dan menjalankan demo
- Tab **Search** : untuk mencari dokumentasi yang mengandung kata / potongan kata tertentu. Untuk mendapatkan *help* dari suatu fungsi tertentu, pilihlah **Search type: Function Name**
- Tab **Favorites** : melihat daftar *link* ke dokumen yang telah ditandai sebagai favorit.

Di antara tab tersebut, yang paling sering digunakan ialah **Contents** dan **Search**. Sebagai latihan, cobalah mencari dokumen mengenai "sound" dengan help browser. Pilih tab **Search**, **Search type**: Full Text, **Search for**: sound.

Penggunaan kaca kunci untuk pencarian mirip dengan mesin pencari di internet (google, yahoo, altavista, dll). Misalkan Anda ingin mencari "filter digital", maka ketikkan dalam **Search for:** filter AND digital.

BAB 2

VARIABEL DAN OPERASI DASAR

2.1 Kalkulator Sederhana

Dalam mode penggunaan dasar, MATLAB dapat digunakan sebagai fungsi kalkulator. Sebagai contoh, kita bisa lakukan perhitungan berikut pada *command window*.

```
>> 3+12
ans =
       15

>> 25*10-16
ans =
       234

>> (9+18)/3^2
ans =
       3
```

Operator aritmatik dasar yang didukung oleh MATLAB ialah sebagai berikut:

Tabel 2. 1

+, -, *, /	: tambah, kurang, kali, bagi
(,)	: kurung
\	: pembagian terbalik
^	: pangkat

Hirarki operator mengikuti standar aljabar yang umum kita kenal:
1. Operasi di dalam kurung akan diselesaikan terlebih dahulu
2. Operasi pangkat
3. Operasi perkalian dan pembagian
4. Operasi penjumlahan dan pengurangan

Sekarang kita coba contoh berikut ini.

```
>> 2.5+0.6
ans =
       3.1000

>> 3*4+3/4
ans =
       12.7500

>> 5\(15+35)
ans =
       10

>> 169^(1/2), (6+14)\10^2
ans =
       13
ans =
       5
```

Dalam contoh di atas kita menemui variabel **ans**, singkatan dari "answer", yang digunakan MATLAB untuk menyimpan hasil perhitungan terakhir.

Tips	Kita bisa melakukan beberapa operasi sekaligus dalam satu baris dengan menggunakan tanda koma sebagai pemisah Gunakan panah atas/bawah ↑↓ berulang-ulang untuk memunculkan lagi *command* yang pernah ditulis sebelumnya.

Penting!	format bilangan "floating point" di MATLAB digambarkan dalam contoh berikut: 2.5×10^7 dituliskan 2.5e7 0.02×10^{-16} dituliskan 0.02e-16 atau .02e-16 10^8 dituliskan 1e8 dan sebagainya

2.2 Menciptakan Variabel

Kita juga bisa menciptakan variabel untuk menyimpan nilai, baik berupa bilangan ataupun teks. Contoh berikut ini untuk menciptakan variabel:

```
>> a=100
a =
       100
>> b=200
b =
       200
>> c=300;
>> d=400;

>> total=a+b+c+d
total =
       1000

>> rata_rata=total/4;
```

Untuk melihat hasil **rata_rata**, kita bisa panggil variabel tersebut.

```
>> rata_rata
rata_rata =
       250
```

Penting!

Jika kita tidak menambahkan tanda titik-koma (;) di akhir *command*, maka MATLAB akan menampilkan variabel dan bilangan yang baru kita masukkan, atau hasil perhitungan yang baru dikerjakan. Jika terdapat titik-koma, maka perhitungan tetap dilakukan tanpa menuliskan hasilnya.

Berikutnya, kita bisa melihat daftar variabel apa saja yang sedang aktif di dalam MATLAB menggunakan *command* **whos**.

```
>> whos
  Name            Size          Bytes  Class

  a               1x1               8  double array
  b               1x1               8  double array
  c               1x1               8  double array
  d               1x1               8  double array
  rata_rata       1x1               8  double array
  total           1x1               8  double array

Grand total is 6 elements using 48 bytes
```

Atau kita juga bisa melihat daftar ini di *window* **Workspace**, di sebelah kiri *command window* (silakan lihat kembali Gambar 1.2).

Untuk menghapus beberapa atau semua variabel kita gunakan *command* **clear**. Misalkan untuk menghapus variabel **total.**

```
>> clear total
```

dan untuk menghapus semua variabel sekaligus

```
>> clear
```

Penamaan Variabel

Pemberian nama variabel mengikuti rambu-rambu berikut ini:

- Gunakan karakter alfabet (A s/d Z, a s/d z), angka, dan garis bawah (_), sebagai nama variabel. Perlu diingat bahwa MATLAB peka terhadap besar-kecilnya huruf.
  Misalkan:
  jumlah, **x1**, **x2**, **S_21**, **H_2_in**; merupakan nama variable yang valid
  sinyal1, **Sinyal1**, **SINYAL1**; dianggap sebagai 3 variabel yang berbeda.
- Jangan gunakan spasi, titik, koma, atau operator aritmatik sebagai bagian dari nama.

Selain berisi bilangan, variabel juga bisa berisi teks. Dalam mendefinisikan variabel teks gunakanlah tanda petik tunggal.

```
>> baca_ini = 'Contoh variabel berisi teks!';

>> baca_ini
baca_ini =
Contoh variabel berisi teks!
```

Kita tidak boleh salah memperlakukan variabel berisi bilangan dengan yang berisi teks, sebab variabel teks juga bisa terlibat dalam operasi perhitungan. Misalkan:

```
>> clear
>> a=7;
>> b='7';

>> a/b
ans =
       0.1273

>> a+b
ans =
       62
```

Terlihat bahwa mengoperasikan variabel berisi teks bisa memunculkan hasil perhitungan yang "salah".

2.3 Variabel Terdefinisi di MATLAB

Di dalam MATLAB telah terdapat beberapa variabel yang telah terdefinisi, sehingga kita bisa langsung pergunakan tanpa perlu mendeklarasikannya lagi. Variabel tersebut ialah:

Tabel 2. 2

ans	"answer", digunakan untuk menyimpan hasil perhitungan terakhir
eps	bilangan sangat kecil mendekati nol yang merupakan batas akurasi perhitungan di MATLAB.
pi	konstanta π, 3.1415926...
inf	"infinity", bilangan positif tak berhingga, misalkan `1/0`, `2^5000`, dsb.
NaN	"not a number", untuk menyatakan hasil perhitungan yang tak terdefinisi, misalkan `0/0` dan `inf/inf`.
i, j	unit imajiner, √-1, untuk menyatakan bilangan kompleks.

2.4 Fungsi Matematika

Berbagi fungsi matematika yang umum kita pergunakan telah terdefinisi di MATLAB, meliputi fungsi eksponensial, logaritma, trigonometri, pembulatan, dan fungsi yang berkaitan dengan bilangan kompleks.

Tabel 2. 3

abs(x)	menghitung nilai absolut dari x, yaitu $\|x\|$
sign(x)	fungsi "signum": bernilai +1 jika x positif, -1 jika x negatif, dan 0 jika x sama dengan nol.
Fungsi eksponensial dan logaritma:	
sqrt(x)	akar kuadrat dari x
exp(x)	pangkat natural dari x, yaitu e^x
log(x)	logaritma natural dari x, yaitu $ln\ x$
log10(x)	logaritma basis 10 dari x, yaitu $log_{10}\ x$
log2(x)	logaritma basis 2 dari x, yaitu $log_2\ x$
Fungsi trigonometri:	
sin(x), cos(x), tan(x), cot(x), sec(x), csc(x)	fungsi trigonometri sinus, cosinus, tangent, cotangent, secant, dan cosecant. (x dalam satuan radian)
asin(x), acos(x), atan(x), acot(x), asec(x), acsc(x)	fungsi arcus trigonometri
sinh(x), cosh(x), tanh(x), coth(x), sech(x), csch(x)	fungsi trigonometri-hiperbolik
asinh(x), acosh(x), atanh(x), acoth(x), asech(x), acsch(x)	fungsi arcus trigonometri-hiperbolik
Fungsi pembulatan:	
round(x)	pembulatan x ke bilangan bulat terdekat
floor(x)	pembulatan ke bawah dari x ke bilangan bulat terdekat
ceil(x)	pembulatan ke atas dari x ke bilangan bulat terdekat
fix(x)	pembulatan ke bawah untuk x positif, dan ke atas untuk x negatif
rem(x,y)	sisa pembagian dari x/y

Fungsi bilangan kompleks:	
real(z)	menghitung komponen riil dari bilangan kompleks *z*
imag(z)	menghitung komponen imajiner dari bilangan kompleks *z*
abs(z)	menghitung magnitude dari bilangan kompleks *z*
angle(z)	menghitung argumen dari bilangan kompleks *z*
conj(z)	menghitung konjugasi dari bilangan kompleks *z*

Bagi Anda yang belum familiar dengan sistem bilangan kompleks, tutorial singkat mengenai topik ini terdapat di Lampiran 2.

Untuk memperdalam pemahaman dari subbab 2.3 dan 2.4, cobalah contoh berikut dan amatilah hasilnya:

```
>> a=pi/2, b=1000, c=-0.5, d=13, e=4

>> sign(a)

>> sqrt(10*b), exp(c), exp(b)
>> log(exp(c)), log10(b), log2(b+24)

>> sin(a), cos(a), tan(a/2)
>> asin(c), acos(c)

>> round(d/e), floor(d/e), ceil(d/e), rem(d,e)

>> A=3+4i, B = sqrt(2) - i*sqrt(2)

>> real(A), imag(A), real(B), imag(B)
>> abs(A), angle(A), abs(B), angle(B)
>> abs(A)*cos(angle(A)), abs(A)*sin(angle(A))
```

Soal Latihan

1. Hitunglah dengan MATLAB:
   12 / 3,5 $(3 + 5/4)^2$ $(0{,}25^2 + 0{,}75^2)^{1/2}$ 2 / (6/0,3)

2. Buatlah empat variabel berikut:
   A = 25 B = 50 C = 125 D = 89
   Hitunglah dan simpan dalam variabel baru:
   X = A + B +C Y = A / (D+B)
   $Z = D^{A/B} + C$

3. Manakah di antara nama-nama variabel berikut yang valid ?
   luas, **kel_1**, **2_data**, **diff:3**, **Time**, **time_from_start**, **10_hasil_terakhir, nilai-awal**

4. Misalkan: $x = \pi/6$, $y = 0{,}001$; hitunglah:
   $\sqrt{y}$ e^{-x} $\sin x$ $\cos 2x$ $\tan 3x$
   $\log_{10} y$ $\log_2 y$ $\ln y$

5. Misalkan: $p = 9+16i$ dan $q = -9+16i$; hitunglah:
   $r = pq$ $s = \dfrac{p}{q}$ $p - r$ $r + s$ p^2 $\sqrt{q}$
   $|p|$ $\angle p$ $|q|$ $\angle q$ $|r|$ $\angle r$ $|s|$ $\angle s$

BAB 3

MATRIKS

3.1 Skalar, Vektor, dan Matriks

Terdapat tiga jenis format data di MATLAB, yaitu skalar, vektor, dan matriks.

- **Skalar**, ialah suatu bilangan tunggal
- **Vektor**, ialah sekelompok bilangan yang tersusun 1-dimensi. Dalam MATLAB biasanya disajikan sebagai vektor-baris atau vektor-kolom
- **Matriks**, ialah sekelompok bilangan yang tersusun dalam segi-empat 2-dimensi. Di dalam MATLAB, matriks didefinisikan dengan jumlah baris dan kolomnya. Di MATLAB terdapat pula matriks berdimensi 3, 4, atau lebih, namun dalam buku ini kita batasi hingga 2-dimensi saja.

Sebenarnya, semua data bisa dinyatakan sebagai matriks. Skalar bisa dianggap sebagai matriks satu baris – satu kolom (matriks 1×1), dan vektor bisa dianggap sebagai matriks 1-dimensi: satu baris – n kolom, atau n baris – 1 kolom (matriks 1×n atau n×1). Semua perhitungan di MATLAB dilakukan dengan matriks, sehingga disebut MATrix LABoratory.

Matriks didefinisikan dengan kurung siku ([ ]) dan biasanya dituliskan baris-per-baris. Tanda koma (,) digunakan untuk memisahkan kolom, dan titik-koma (;) untuk memisahkan baris. Kita juga bisa menggunakan spasi untuk memisahkan kolom dan menekan Enter ke baris baru untuk memisahkan baris.

Perhatikan cara mendefinisikan skalar dengan ataupun tanpa kurung siku.

```
>> skalar1 = 3.1415
skalar1 =
       3.1415

>> skalar2 = [2.71828]
skalar2 =
       2.7183
```

Contoh vektor-baris dan vektor-kolom

```
>> vektor1=[3,5,7]
vektor1 =
       3      5      7

>> vektor2=[2;4;6]
vektor2 =
       2
       4
       6
```

Berikutnya kita coba contoh berikut untuk mendefinisikan matriks 3×3.

```
>> matriks1=[10 20 30
40 50 60
70 80 90]

>> matriks2=[10 20 30; 40 50 60; 70 80 90]
```

Terlihat bahwa **matrix1** dan **matrix2** isinya sama, karenanya kita bisa menekan Enter untuk membuat baris baru, ataupun menggunakan titik-koma.
Kita juga bisa mendefinisikan matriks elemen per elemen.

```
>> mat(1,1)=100; mat(1,2)=200; mat(2,1)=300;
>> mat(2,2)=400
mat =
       100    200
       300    400
```

Kita sekarang akan mencoba menggabungkan variabel yang ada untuk membentuk matriks baru.

```
>> gabung1=[vektor2 matriks1]
gabung1 =
       2      10     20     30
       4      40     50     60
       6      70     80     90

>> gabung2=[vektor1; matriks2]
gabung2 =
        3      5      7
       10     20     30
       40     50     60
       70     80     90
```

Kita harus ingat bahwa matriks gabungan harus memiliki jumlah baris dan kolom yang valid sehingga membentuk persegi panjang.

Sekarang cobalah menghitung matriks gabungan berikut.

```
>> gabung3=[vektor2 vektor2 vektor2]
>> gabung4=[vektor1;vektor1;vektor1]
>> gabung5=[gabung3 gabung4]
```

3.2 Ukuran Matriks

Untuk mengetahui ukuran atau dimensi dari matriks yang ada, kita bisa gunakan *command* **size** dan **length**. **size** umumnya digunakan untuk matriks 2-dimensi, sementara **length** untuk vektor.

```
>> length(vektor1)
ans =
       3

>> size(matrix1)
ans =
       3      3
```

Menunjukkan panjang **vektor1** ialah 3 elemen, dan ukuran **matrix1** ialah 3-baris 3-kolom (3×3). Kita juga bisa menyimpan keluaran *command* dalam variabel baru.

```
>> panjang=length(vektor2)
panjang =
       3

>> [jml_baris,jml_kolom]=size(gabung5)
jml_baris =
       3
jml_kolom =
       6
```

Sementara itu, untuk menghitung jumlah elemen dari suatu matriks, kita pergunakan *command* **prod**. Misalkan untuk matriks **gabung5**, jumlah elemennya ialah;

```
>> jml_elemen=prod(size(gabung5))
jml_elemen =
       18
```

3.3 Matriks Khusus

MATLAB menyediakan berbagai *command* untuk membuat dan memanipulasi matriks secara efisien. Di antaranya ialah *command* untuk membuat matriks-matriks khusus, manipulasi indeks matriks, serta pembuatan deret. Mari kita bahas terlebih dahulu mengenai matriks khusus.

Berbagai matriks khusus yang kerap kita pergunakan dalam perhitungan bisa dibuat secara efisien dengan *command* yang telah ada di MATLAB.

Tabel 3. 1

ones(n)	membuat matriks satuan (semua elemennya berisi angka 1) berukuran n×n.
ones(m,n)	membuat matriks satuan berukuran m×n.
zeros(n)	membuat matriks nol (semua elemennya berisi angka 0) berukuran n×n.
zeros(m,n)	membuat matriks nol berukuran m×n.
eye(n)	membuat matriks identitas berukuran n×n (semua elemen diagonal bernilai 1, sementara lainnya bernilai 0).
rand(n), rand(m,n)	membuat matriks n×n, atau m×n, berisi bilangan random terdistribusi uniform pada selang 0 s.d. 1.
randn(n), randn(m,n)	membuat matriks n×n, atau m×n, berisi bilangan random terdistribusi normal dengan mean = 0 dan varians = 1. *Command* ini kerap kita gunakan untuk membangkitkan derau putih gaussian.
[]	matriks kosong, atau dengan kata lain matriks 0×0; biasa digunakan untuk mendefinisikan variabel yang belum diketahui ukurannya.

Untuk memperdalam pemahaman, mari kita lihat contoh di bawah ini.

```
>> clear
```

```
>> mat_1=5*ones(2,4)
mat_1 =
       5      5      5      5
       5      5      5      5

>> mat_2=zeros(2,4)
mat_2 =
       0      0      0      0
       0      0      0      0

>> mat_3=[eye(4) -ones(4)]
mat_3 =
1      0      0      0      -1     -1     -1     -1
0      1      0      0      -1     -1     -1     -1
0      0      1      0      -1     -1     -1     -1
0      0      0      1      -1     -1     -1     -1

>> bil_acak_uniform=rand(1,10)
bil_acak_uniform =
Columns 1 through 7
0.9501  0.2311  0.6068  0.4860  0.8913  0.7621  0.4565

Columns 8 through 10
0.0185  0.8214  0.4447

>> gaussian_noise=randn(5,1)
gaussian_noise =
    -0.4326
    -1.6656
     0.1253
     0.2877
    -1.1465
```

Misalkan kita ingin membangkitkan 20 buah bilangan acak gaussian dengan mean = 5 dan varians = 3.

```
>> mu=5;      %Nilai mean
>> varians=3; %Nilai variansi

>> bil_acak_gaussian= sqrt(varians)*randn(1,20) + mu
bil_acak_gaussian =
```

Tips

Setiap kali kita menggunakan *command* **rand** dan **randn**, kita akan selalu mendapatkan nilai keluaran yang berbeda. Hal ini merupakan salah satu sifat bilangan acak.

3.4 Manipulasi Indeks Matriks

Dalam vektor ataupun matriks, indeks digunakan untuk menunjuk satu/beberapa elemen dari vektor/matriks. Indeks dituliskan di dalam tanda kurung ( ) dengan pola umum sebagai berikut.

Untuk vektor:

nama_vektor(indeks)

Untuk matriks:

nama_matriks(indeks_baris , indeks_kolom)

Dalam suatu vektor, elemen pertama diberi indeks = 1, sementara dalam matriks, indeks menunjukkan nomor baris dan nomor kolom dari elemen yang ingin ditunjuk. Untuk lebih jelasnya perhatikan contoh berikut ini.

```
>> clear
>> vektor_ini = [1 3 5 7 9];
>> vektor_itu = [9; 8; 7; 6; 5];
>> matrix = [10 20 30; 40 50 60; 70 80 90];

>> vektor_ini(1)
ans =
       1

>> vektor_itu(2)
ans =
       8

>> matrix(1,2)
ans =
       20

>> [matrix(1,1) matrix(1,2) matrix(1,3)]
ans =
       10     20     30
```

Operator-Titik Dua

Kita juga bisa mengambil beberapa baris dan kolom sekaligus dari suatu matriks dengan operator titik-dua (:). Dalam hal ini tanda titik-dua berarti **"sampai dengan"**.

Misalkan untuk mengambil elemen ke-1 sampai ke-3 dari **vektor_ini**

```
>> vektor_ini(1:3)
ans =
       1      3      5
```

Mengambil elemen ke-3 sampai ke-5 dari **vektor_itu**

```
>> vektor_itu(3:5)
ans =
       7
       6
       5
```

Mengambil elemen baris ke-1 sampai ke-2, kolom ke-2 sampai ke-3 dari **matrix**

```
>> matrix(1:2,2:3)
ans =
       20     30
       50     60
```

Dalam hal lain tanda titik-dua bisa berarti **"seluruhnya"**.
Misalkan untuk mengambil seluruh elemen dari **vektor_ini**

```
>> vektor_ini(:)
ans =
       1      3      5      7      9
```

Mengambil seluruh baris dan kolom dari **matrix**

```
>> matrix(:,:)
ans =
       10     20     30
       40     50     60
       70     80     90
```

Mengambil seluruh elemen di baris ke-1 dari **matrix**

```
>> matrix(1,:)
ans =
       10     20     30
```

Mengambil seluruh elemen di kolom ke-2 dari **matrix**

```
>> matrix(:,2)
```

```
ans =
       20
       50
       80
```

Mengambil seluruh elemen di kolom ke-2 dan ke-3 dari **matrix**

```
>> matrix(:,2:3)
ans =
       20     30
       50     60
       80     90
```

Dengan menggunakan indeks, kita bisa mengubah nilai elemen matriks yang telah ada.

```
>> vektor_ini(1)=1000
vektor_ini =
       1000   3      5      7      9

>> vektor_itu(2:4)=[-1; -1; -1]
vektor_itu =
        9
       -1
       -1
       -1
        5

>> matrix(3,:)=100*ones(1,3)
matrix =
        10     20     30
        40     50     60
       100    100    100
```

3.5 Membuat Deret

Deret bilangan merupakan hal yang kerap kita temui dalam pengolahan data, terutama berkaitan dengan plot data dan proses iterasi (perhitungan berulang-ulang). Misalkan kita memiliki data tegangan suatu baterai pada setiap menit selama 1 jam. Dalam menyajikan data "waktu", kita harus membuat vektor berisi deret. Kita tentunya bisa melakukannya secara manual seperti ini:

```
>> time=[1, 2, 3, 4, …, 60]
```

Tetapi akan lebih efisien jika deret diciptakan menggunakan

operator titik-dua. Formulanya ialah:

deret = nilai_awal : inkremen : nilai_akhir

Inkremen harus bilangan bulat positif atau negatif
Khusus untuk inkremen = 1:

deret = nilai_awal : nilai_akhir

Sehingga kita bisa tuliskan

```
>> time=1:60
```

Sekarang kita akan berlatih menggunakan operator titik-dua untuk membuat deret berikut:

x = 0, 100, 200, 300, 400, … , 2200, 2300
y = -10, -9.5, -9, -8.5, … -0.5, 0, 0.5, … , 9, 9.5, 10
z = 10, 9.95, 9.9, 9.85, 9.8, 9.75, … , 1, 0.95, 0.9, … , 0.05, 0

```
>> x=0:100:2300;
>> y=-10:0.5:10;
>> z=10:-0.05:0;
```

Penting! Bedakan operator titik-dua untuk manipulasi indeks matriks dengan operator titik-dua untuk membuat deret. Untuk membedakannya ingatlah selalu bahwa indeks selalu berada di dalam tanda kurung ( )

Di dalam MATLAB, pembuatan deret juga bisa dilakukan dengan *command* berikut ini.

Tabel 3. 2

linspace(a,b,n)	membuat vektor baris berisi **n** titik yang terpisah merata secara linier antara **a** dan **b**.
logspace(a,b,n)	membuat vektor baris berisi **n** titik yang terpisah merata secara logaritmik antara **10^a** dan **10^b**. *Command* ini biasa digunakan untuk menghitung respon frekuensi suatu sistem.

Contoh:

```
>> linspace(0,10,11)
ans =
      0   1   2   3   4   5   6   7   8   9  10

>> logspace(0,2,10)
ans =
Columns 1 through 7
1.0000  1.6681  2.7826  4.6416  7.7426 12.9155 21.5443

Columns 8 through 10
35.9381 59.9484 100.0000
```

3.6 Membentuk-Ulang Matriks

Terdapat beberapa *command* yang bisa digunakan untuk menukar, merotasi, dan menyusun kembali elemen matriks.

Tabel 3. 3

fliplr(A)	menukar posisi elemen matriks **A** secara melintang, yaitu sebelah kiri ditukar dengan sebelah kanan.
flipud(A)	menukar posisi elemen matriks **A** secara membujur, yaitu sebelah atas ditukar dengan sebelah bawah.
rot90(A)	merotasi posisi elemen matriks **A** berlawanan arah jarum jam sejauh 90^o.
reshape(A,m,n)	menyusun ulang elemen matriks **A** menjadi berukuran **m×n**. Harus diingat bahwa jumlah elemen **A** harus sama dengan **m×n**

Contoh:

```
>> A=[0:3; 4:7]
A =
     0     1     2     3
     4     5     6     7

>> fliplr(A)
ans =
     3     2     1     0
     7     6     5     4
```

```
>> flipud(A)
ans =
     4     5     6     7
     0     1     2     3

>> rot90(A)
ans =
     3     7
     2     6
     1     5
     0     4

>> reshape(A,1,8)
ans =
     0     4     1     5     2     6     3     7

>> reshape(A,4,2)
ans =
     0     2
     4     6
     1     3
     5     7
```

Soal Latihan

1. Definisikan vektor dan matriks berikut ini di dalam MATLAB:

$$\begin{pmatrix}10 & 20 & 30 & 40\end{pmatrix} \quad \begin{pmatrix}-5\\-15\\-40\end{pmatrix} \quad \begin{pmatrix}1 & 3 & 5 & 0\\3 & 1 & 3 & 5\\5 & 3 & 1 & 3\\0 & 5 & 3 & 1\end{pmatrix}$$

2. Gabungkan matriks **A** dan **B** berikut ini:

$A = \begin{pmatrix}4 & 8\\2 & 4\end{pmatrix}$ $B = \begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$ menjadi:

$$C = \begin{pmatrix}A & B\end{pmatrix} \quad W = \begin{pmatrix}B & B\\B & -B\end{pmatrix}$$

3. Hitunglah:
   a. Masing-masing ukuran vektor/matriks pada soal no.1 dan no. 2 di atas
   b. Masing-masing jumlah elemen vektor/matriks pada soal no.1 dan no.2 di atas.

4. Buatlah matriks-matriks berikut dengan *command* **ones**, **zeros**, dan **eye**:

$$\begin{pmatrix}5 & 0 & 0 & 0\\0 & 5 & 0 & 0\\0 & 0 & 5 & 0\\0 & 0 & 0 & 5\end{pmatrix} \quad \begin{pmatrix}5 & 5 & 0 & 0\\5 & 5 & 0 & 0\\-5 & 0 & 0 & 5\\0 & -5 & 5 & 0\end{pmatrix}$$

5. Buatlah vektor berukuran 100 berisi bilangan acak gaussian dengan mean = 1 dan variansi = 0,2.

6. Buatlah matriks **M** berikut ini:

$$M = \begin{pmatrix}1 & 5 & 10 & 15 & 20\\1 & 2 & 4 & 8 & 16\\-3 & 0 & 3 & 6 & 9\\32 & 16 & 8 & 4 & 2\\5 & -5 & 5 & -5 & 5\end{pmatrix}$$

Buatlah vektor / matriks baru berisi:
- baris pertama dari **M**
- kolom ketiga dari **M**
- baris ketiga hingga kelima, kolom kedua hingga keempat dari **M**
- elemen pada diagonal utama dari **M**

7. Buatlah deret berikut ini dengan operator titik-dua, **linspace**, dan **logspace**:
   x = -10, -9, -8, ... , 8, 9, 10
   y = 7,5 , 7,0 , 6,5 , 6,0 , ... , 0,5 , 0
   z = 1, 4, 7, 10, 13, ... , 100
   w = 0,001 , 0,01 , 0,1 , 1 , 10 , ... , 10^6

8. Buatlah matriks **N** yang berisi kolom pertama hingga keempat dari matriks **M** pada no.6 di atas. Bentuk-ulang matriks **N** tersebut menjadi matriks baru seperti berikut ini:
   - kolom pertama ditukar dengan kolom keempat, kolom kedua ditukar dengan kolom ketiga
   - baris pertama ditukan dengan baris kelima, baris kedua ditukar dengan baris keempat
   - matriks berukuran 10×2
   - matriks berukuran 4×5

BAB 4

OPERASI MATRIKS

Ketika kita bekerja dengan matriks di dalam MATLAB, operasi ataupun manipulasi yang kita lakukan terhadap matriks tersebut bisa berupa: operasi (aljabar) matriks, dan operasi elemen-per-elemen. Operasi matriks di MATLAB sama seperti yang kita temui di aljabar matriks, misalkan penjumlahan/pengurangan, perkalian matriks, invers, transpose, dot product, cross product, dan sebagainya. Sementara operasi elemen-per-elemen, yang merupakan ciri khas MATLAB, mengoperasikan satu per satu elemen matriks seperti operasi skalar, meliputi penjumlahan/pengurangan, perkalian/pembagian, dan pangkat. Dalam bab ini, operasi matriks dibahas terlebih dahulu, dan kemudian operasi elemen-per-elemen.

4.1 Penjumlahan dan Pengurangan

Penjumlahan dua matriks, **A+B**, dan selisih dua matriks, **A–B**, terdefinisi jika **A** dan **B** berukuran sama. Namun demikian, penjumlahan/pengurangan juga bisa dilakukan antara matriks dengan skalar. Untuk jelasnya mari kita praktekkan contoh berikut ini.

```
>> A=[0 1;2 3];
>> B=[4 5;6 7];

>> Jumlah=A+B, Selisih=A-B, Tambah50=A+50
Jumlah =
     4     6
     8    10
Selisih =
    -4    -4
    -4    -4
Tambah50 =
    50    51
    52    53
```

4.2 Perkalian Matriks

Perkalian matriks, misalkan **C = AB**, terdefinisi jika jumlah kolom di **A** sama dengan jumlah baris di **B**. Selain itu, perkalian juga bisa dilakukan antara matriks dengan skalar.
Kita akan lanjutkan contoh sebelumnya.

```
>> A,B
A =
     0     1
     2     3
B =
     4     5
     6     7

>> MultAB=A*B, MultBA=B*A
MultAB =
     6     7
    26    31
MultBA =
    10    19
    14    27
```

Tips

Ketika mengalikan dua matriks, maka matriks hasil perkalian dihitung berdasarkan formula baku. Misalkan **C=AB**; **A** dan **B** matriks 2×2,
sehingga hasilnya **C** juga 2×2.

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

di mana:
$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$
$$c_{12} = a_{11}b_{12} + a_{12}b_{22}$$
$$c_{21} = a_{21}b_{11} + a_{22}b_{21}$$
$$c_{22} = a_{21}b_{12} + a_{22}b_{22}$$

Contoh berikutnya ialah perkalian dua vektor, yang juga mengikuti aturan perkalian matriks, karena vektor sesungguhnya sama dengan matriks 1-dimensi.

```
>> x=[3 2 1], y=[100;10;1]
x =
     3     2     1
```

```
y =
   100
    10
     1

>> z1=x*y, z2=y*x
z1 =
   321
z2 =
   300   200   100
    30    20    10
     3     2     1
```

Selain perkalian di atas, dikenal pula perkalian vektor, yaitu: "dot-product" (atau disebut juga *inner-product*), dan "cross-product".

Tabel 4. 1

dot(x,y)	menghitung *dot-product* dari vektor **x** dan **y**
cross(x,y)	menghitung *cross-product* dari vektor **x** dan **y**

Tips

Dot-product dan *cross-product* dihitung berdasarkan formula baku.
Misalkan terdapat dua vektor $\mathbf{x} = (x_1 \ x_2 \ x_3)$ dan $\mathbf{y} = (y_1 \ y_2 \ y_3)$, maka:

dot-product: $\mathbf{x} \bullet \mathbf{y} = x_1y_1 + x_2y_2 + x_3y_3$
cross-product: $\mathbf{x} \times \mathbf{y} = (x_2y_3 - x_3y_2 \quad x_3y_1 - x_1y_3 \quad x_1y_2 - x_2y_1)$

Perlu diingat bahwa hasil *dot-product* berupa skalar, sementara hasil *cross-product* berupa vektor.

4.3 Persamaan Linier dalam Matriks

Kita sering menemui persamaan linier dengan beberapa variabel. Di dalam aljabar, solusi persamaan tersebut bisa ditemukan, salah satunya dengan menggunakan matriks. Misalkan kita tinjau sistem persamaan linier dengan variabel x_1 dan x_2.

$$\begin{aligned} x_1 - 2\,x_2 &= 32 \\ 12\,x_1 + 5\,x_2 &= 7 \end{aligned}$$

Dalam bentuk matriks bisa kita tuliskan:

$$\begin{pmatrix} 1 & -2 \\ 12 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 32 \\ 7 \end{pmatrix} \quad \Leftrightarrow \quad \mathbf{AX = B}$$

$\mathbf{X = A^{-1}B}$; di mana $\mathbf{A^{-1}}$ ialah invers matriks **A**

Dalam MATLAB kita tuliskan:

```
>> A=[1 -2;12 5]; B=[32;7];

>> X=inv(A)*B
X =
    6.0000
  -13.0000
```

Sehingga kita dapatkan solusi $x_1 = 6$ dan $x_2 = -13$.
Atau kita juga bisa mendapatkan solusi tersebut dengan operator pembagian terbalik:

```
>> X=A\B
X =
    6.0000
  -13.0000
```

Sebagai bahan latihan, cobalah Anda pecahkan persamaan linier dengan tiga variabel berikut ini.

$$\begin{aligned} x + 2y + 3z &= 2 \\ 4x + 5y + 6z &= -5{,}5 \\ 7x + 8y - 9z &= -49 \end{aligned}$$

4.4 Transposisi

Salah satu operasi yang penting dalam matriks ialah transposisi, dituliskan dalam MATLAB dengan operator petik tunggal (‘) dan titik-petik (.’). Operasi ini mempertukarkan baris dan kolom dari suatu matriks atau vektor.

Tabel 3. 1

petik tunggal (‘)	operasi transposisi untuk matriks berisi bilangan riil, atau transposisi dan konjugasi untuk matriks kompleks.
titik-petik (.’)	operasi transposisi tanpa konjugasi. Untuk matriks riil, operator ini memberi hasil yang sama dengan petik tunggal

Mari kita praktekkan contoh berikut ini untuk memahami kedua operator di atas.

```
>> Mat_riil=[1 0; 3 5], Mat_kompleks=[1+2i 3i; 1 2+3i]
Mat_riil =
     1     0
     3     5
Mat_kompleks =
   1.0000 + 2.0000i         0 + 3.0000i
   1.0000              2.0000 + 3.0000i

>> Transp_riil=Mat_riil',Transp_kompleks=Mat_kompleks'
Transp_riil =
     1     3
     0     5
Transp_kompleks =
   1.0000 - 2.0000i   1.0000
        0 - 3.0000i   2.0000 - 3.0000i

>> Transp_riil2=Mat_riil.'
Transp_riil2 =
     1     3
     0     5

>> Transp_kompleks2=Mat_kompleks.'
Transp_kompleks2 =
   1.0000 + 2.0000i   1.0000
        0 + 3.0000i   2.0000 + 3.0000i
```

4.5 Operasi Elemen-per-Elemen

Di dalam MATLAB, operasi matematik juga bisa dilakukan elemen-per-elemen. Dalam hal ini matriks atau vektor yang terlibat harus berukuran sama. Operasi yang bisa dilakukan ialah perkalian/pembagian, penjumlahan/pengurangan, serta pangkat. Operator yang digunakan diawali dengan tanda “titik” (kecuali penjumlahan/pengurangan), yaitu:

Tabel 3. 2

+ –	Tambah dan kurang (elemen-per-elemen)
.* ./ .	Kali, bagi, bagi terbalik (elemen-per-elemen)
.^	Pangkat (elemen-per-elemen)

Operasi penjumlahan/pengurangan matriks secara definit sudah dilakukan elemen-per-elemen, sehingga + dan – tidak diawali "titik".

Sekarang kita coba praktekkan contoh di bawah ini.

```
>> A=[1 -2;1 5]; B=[7 5; 2 0];

>> A+B
ans =
     8     3
     3     5

>> A.*B
ans =
     7   -10
     2     0

>> B./A
ans =
    7.0000   -2.5000
    2.0000         0

>> B.^2
ans =
    49    25
     4     0

>> A.^B
ans =
     1   -32
     1     1

>> 2.^B
ans =
   128    32
     4     1
```

Perhatikan bahwa hasil operasi juga berupa matriks berukuran sama dengan **A** dan **B**.

Pada contoh berikutnya kita coba operasi antar vektor.

```
>> a = [3 2 1]; b = [4 5 6];
>> c = [10 20 30]'; d = [5 10 15]';

>> a.*b
ans =
    12    10     6

>> c.*d
ans =
    50
   200
   450

>> a.*c
??? Error using ==> .*
Matrix dimensions must agree.
```

Perhatikan bahwa ukuran **a** dan **c** tidak cocok sehingga muncul pesan error (**a** berukuran 1×3 sementara **c** 3×1).

```
>> b.^a, c./d+2
ans =
    64    25     6
ans =
     4
     4
     4

>> c./2.*d.^2
ans =
         125
        1000
        3375
```

Ingat, operasi pangkat selalu dilakukan lebih dulu, diikuti perkalian/pembagian, kemudian penjumlahan/pengurangan.

4.6 Fungsi Elemen-per-Elemen

Semua fungsi matematik yang berlaku pada skalar (lihat kembali subbab 2.4), berlaku pula untuk matriks/vektor secara elemen-per-elemen. Pada contoh kali ini, kita akan mencoba beberapa contoh sederhana, kemudian kita coba pula dua kasus perhitungan dengan memanfaatkan berbagai fungsi yang telah kita pelajari.

```
>> n=-3:3
n =
    -3    -2    -1     0     1     2     3

>> abs(n), sign(n)
ans =
     3     2     1     0     1     2     3
ans =
    -1    -1    -1     0     1     1     1

>> round(n./2), floor(n./2), ceil(n./2)
ans =
    -2    -1    -1     0     1     1     2
ans =
    -2    -1    -1     0     0     1     1
ans =
    -1    -1     0     0     1     1     2

>> rem(n,3)
ans =
     0    -2    -1     0     1     2     0
```

Contoh Kasus

Berikutnya, kita pelajari contoh kasus pertama:

Misalkan Anda ditugasi untuk mencari solusi persamaan logaritmik:

$y = ln(x^2)$

di mana x bernilai antara –100 hingga +100. Setelah itu, Anda harus menampilkan nilai pada rentang $x = –2$ hingga $x = 2$ saja.

```
>> clear
>> inkremen = 0.5;
>> x = -100:inkremen:100;  %Di sini kita definisikan x,
>> y = log(x.^2);          %kemudian kita hitung y
Warning: Log of zero.
```

Warning muncul karena terdapat perhitungan $y = log(0)$ ketika $x=0$. Untuk menghindari *warning*, kita bisa buat angka di dalam logaritma tidak pernah bernilai nol dengan cara menambahkan bilangan "amat kecil" **eps**.

```
>> y = log(x.^2+eps);
```

Nilai x telah didefinisikan, dan y telah dihitung. Sekarang, kita

harus melokalisasi data pada rentang –2 hingga +2. Untuk melakukannya, kita harus tahu panjang vektor *x*, dan pada nomor indeks berapa saja *x* bernilai –2 hingga +2.

```
>> panjang = length(x)
panjang =
   401

>> titik_tengah = round(panjang/2)
titik_tengah =
   201
```

Pada **titik_tengah** ini, *x* bernilai 0. Sekarang kita ambil nilai *x* di kiri dan kanan **titik_tengah** sebanyak 4 titik untuk mendapatkan *x* = –2 hingga *x* = 2.

```
>> x_baru = x(titik_tengah-4:titik_tengah+4)
x_baru =
Columns 1 through 7
-2.0000 -1.5000 -1.0000 -0.5000    0   0.5000   1.0000

Columns 8 through 9
1.5000   2.0000
```

Lalu kita tampilkan nilai *y* pada rentang tersebut.

```
>> y_baru = y(titik_tengah-4:titik_tengah+4)
y_baru =
Columns 1 through 7
1.3863 0.8109 0.0000 -1.3863 -36.0437 -1.3863   0.0000

Columns 8 through 9
0.8109  1.3863
```

Berikutnya pada contoh kasus kedua:

Anda ditugasi membuat tabel trigonometri: sinus dan cosinus untuk sudut-sudut istimewa: 0^o, 30^o, 45^o, 60^o, 90^o, ... , 360^o. Dalam tugas ini akan digunakan pula command **sort** untuk mengurutkan data dan **disp** untuk menampilkan isi variabel di layar.

Mula-mula, kita definisikan **x** sebagai sudut-sudut istimewa, berupa sudut kelipatan 30^o mulai 0^o hingga 360^o. Kemudian kita tambahkan empat sudut istimewa: 45^o, 135^o, 225^o, dan 315^o, lalu kita urutkan isi vektor **x**.

```
>> clear
```

```
>> x=0:30:360;
>> x=[x 45 135 225 315];

>> x=sort(x)
x =
Columns 1 through 13
0  30  45  60  90 120 135 150 180 210 225 240 270

Columns 14 through 17
300 315 330 360
```

x dalam satuan derajat kita ubah menjadi **t** (radian), karena perhitungan trigonometri dilakukan dalam satuan radian.

```
>> t=x.*pi/180;
>> y1=sin(t); y2=cos(t);
```

Selanjutnya kita buat matriks tiga kolom bernama **tabel** berisi: sudut, sin, dan cos.

```
>> tabel=[x;y1;y2]';
>> judul='     sudut     sin        cos';
```

Ingat, vektor **x**, **y1**, dan **y2** berupa satu baris; padahal kita ingin menampilkannya memanjang ke bawah berupa kolom, jadi perlu dilakukan transposisi.

```
>> disp(judul), disp(tabel)
     sudut    sin       cos
         0         0    1.0000
   30.0000    0.5000    0.8660
   45.0000    0.7071    0.7071
   60.0000    0.8660    0.5000
   90.0000    1.0000    0.0000
  120.0000    0.8660   -0.5000
  135.0000    0.7071   -0.7071
  150.0000    0.5000   -0.8660
  180.0000    0.0000   -1.0000
  210.0000   -0.5000   -0.8660
  225.0000   -0.7071   -0.7071
  240.0000   -0.8660   -0.5000
  270.0000   -1.0000   -0.0000
  300.0000   -0.8660    0.5000
  315.0000   -0.7071    0.7071
  330.0000   -0.5000    0.8660
  360.0000   -0.0000    1.0000
```

Soal Latihan

1. Operasikan matriks **M** dan **N** berikut ini:

   $$M = \begin{pmatrix} 10 & 20 \\ 5 & 8 \end{pmatrix} \quad N = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

   M + **N**, **M** – **N**, **N** + 9
   MN, **NM**

2. Hitunglah dot-product dan cross-product dari dua vektor berikut ini:

   $\vec{a} = (0 \quad 5 \quad 5) \quad \vec{b} = (1 \quad 1 \quad 1)$

   $\vec{a} \bullet \vec{b} \quad \vec{a} \times \vec{b} \quad \vec{b} \times \vec{a}$

3. Pecahkanlah persamaan linier tiga variabel berikut ini:

   $x + 2y - 3z = -7$
   $4x + 5y + 6z = 11$
   $7x + 8y + 9z = 17$

4. Carilah solusi dari persamaan lingkaran berikut ini:

   $y = \sqrt{25 - x^2}$ untuk $-5 \le x \le 5$, dengan inkremen *x* sebesar 0,05. Setelah itu, tampilkanlah nilai *y* pada rentang *x* = 0 hingga *x* = 1 saja.

5. Buatlah tabel hiperbolik-trigonometri: sinh, cosh, dan tanh untuk rentang $-5 \le x \le 5$, dengan inkremen *x* sebesar 0,1.

BAB 5

GRAFIK DAN SUARA

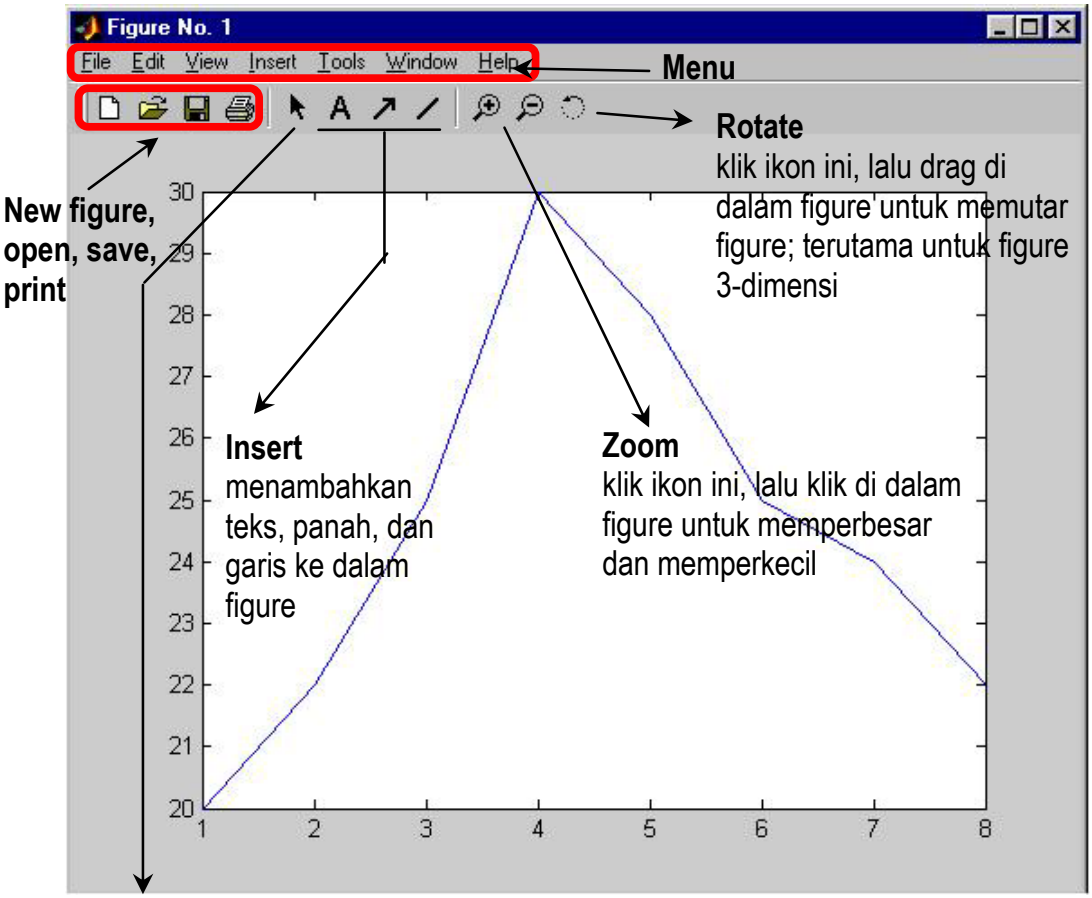
Salah satu keunggulan MATLAB ialah kemampuannya dalam menampilkan/mengolah grafik dan suara dengan *command* yang sederhana dan fleksibel. Pada bab ini ini kita akan belajar mengenai visualisasi data (plot grafik 2-dimensi dan 3-dimensi), serta penyuaraan.

5.1 Plot 2-Dimensi

Untuk memvisualisasi data secara 2-dimensi ataupun 3-dimensi, kita menggunakan berbagai command plotting; di mana command yang paling dasar ialah **plot**. Anda bisa praktekan contoh berikut ini.

```
>> x = 1:8; y=[20 22 25 30 28 25 24 22];
>> plot(x,y)
```

Akan muncul *window* baru berisi figure hasil plotting. Perhatikan kegunaan dari ikon yang ada.



Gambar 5. 1 Jendela figure.

Seperti yang Anda lihat, titik (1,20), (2,22), (3,25), (4,30), dst... terhubung dengan garis lurus. Sekarang Anda bisa coba untuk membalik urutan sintaks dan mengamati grafik yang dihasilkan!

```
>> plot(y,x)
```

Setiap gambar di *figure window*, bisa Anda print melalui menu **File→Print** (Ctrl+P), atau Anda simpan sebagai file FIG dengan **File→Save** (Ctrl+S), ataupun Anda ekspor sebagai file JPG, EMF, BMP, dsb dengan **File→Export**.

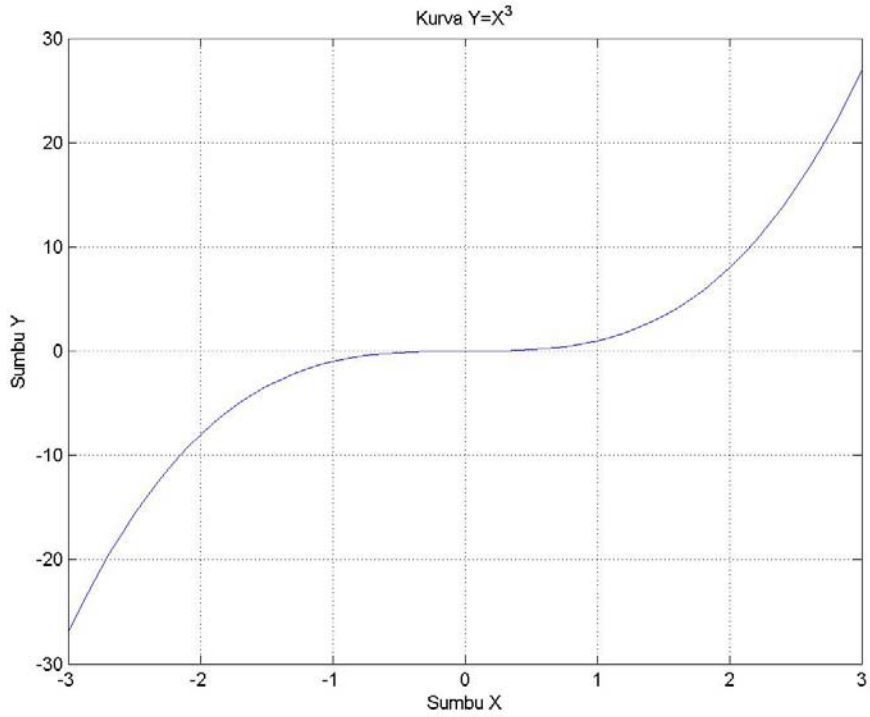
Untuk menambahkan judul, label, dan grid ke dalam hasil plot Anda, digunakan *command* berikut ini.

Tabel 5. 1

xlabel	memberi label pada sumbu-x
ylabel	memberi label pada sumbu-y
title	memberi judul di atas area plot
grid on	memunculkan grid di dalam area plot
grid off	menghapus grid

Sekarang mari kita lihat contoh plot yang lain. Kita akan memplot kurva $y = x^3$ pada rentang x = -3 hingga x = +3.

```
>> clear
>> x=-3:0.1:3; %inkremen=0.1 agar kurva terlihat mulus
>> y=x.^3;
>> plot(x,y)
>> xlabel('Sumbu X'), ylabel('Sumbu Y')
>> title('Kurva Y=X^3')
>> grid on
```



Gambar 5. 2 Contoh plot: kurva Y = X^3

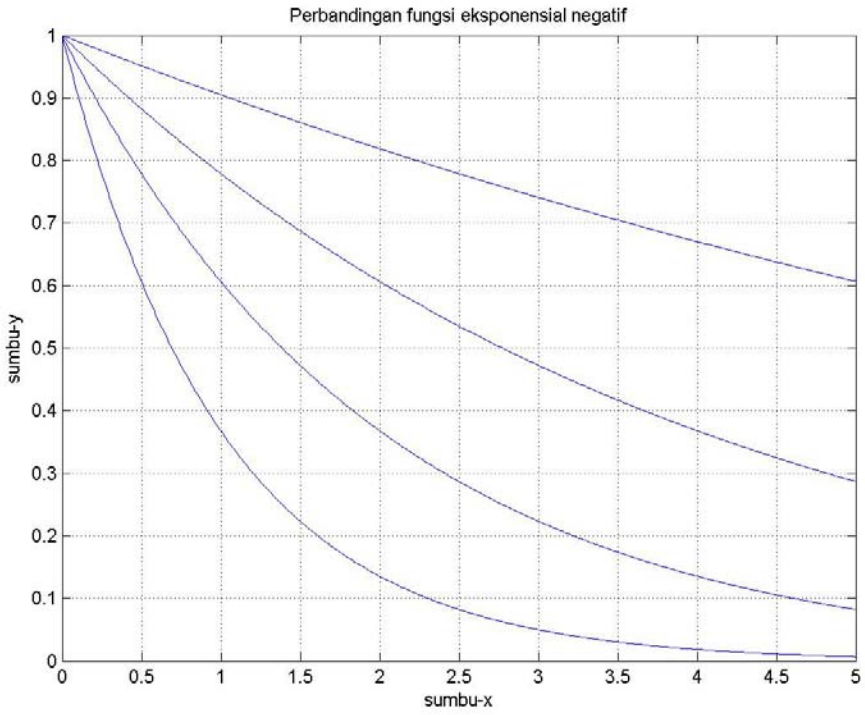
Ketika Anda menggunakan command **plot**, gambar sebelumnya di *figure window* akan terhapus. Lalu bagaimana jika kita ingin memplot beberapa fungsi dalam satu figure sekaligus? Dalam hal ini kita bisa gunkan command **hold**.

Tabel 5. 2

hold on	untuk 'menahan' gambar sebelumnya supaya tak terhapus ketika ditimpa gambar baru
hold off	untuk menonaktifkan *command* **hold**

Berikut ini contoh memplot beberapa kurva eksponensial negatif sekaligus.

```
>> clear
>> x=linspace(0,5,500);
>> y1=exp(-x); plot(x,y1);
>> grid on
>> hold on
>> y2=exp(-0.5*x); plot(x,y2);
>> y3=exp(-0.25*x); plot(x,y3);
>> y4=exp(-0.1*x); plot(x,y4);
>> xlabel('sumbu-x'), ylabel('sumbu-y')
>> title('Perbandingan fungsi eksponensial ...
negatif')
```



Gambar 5. 3 Hasil plot dengan "hold on"

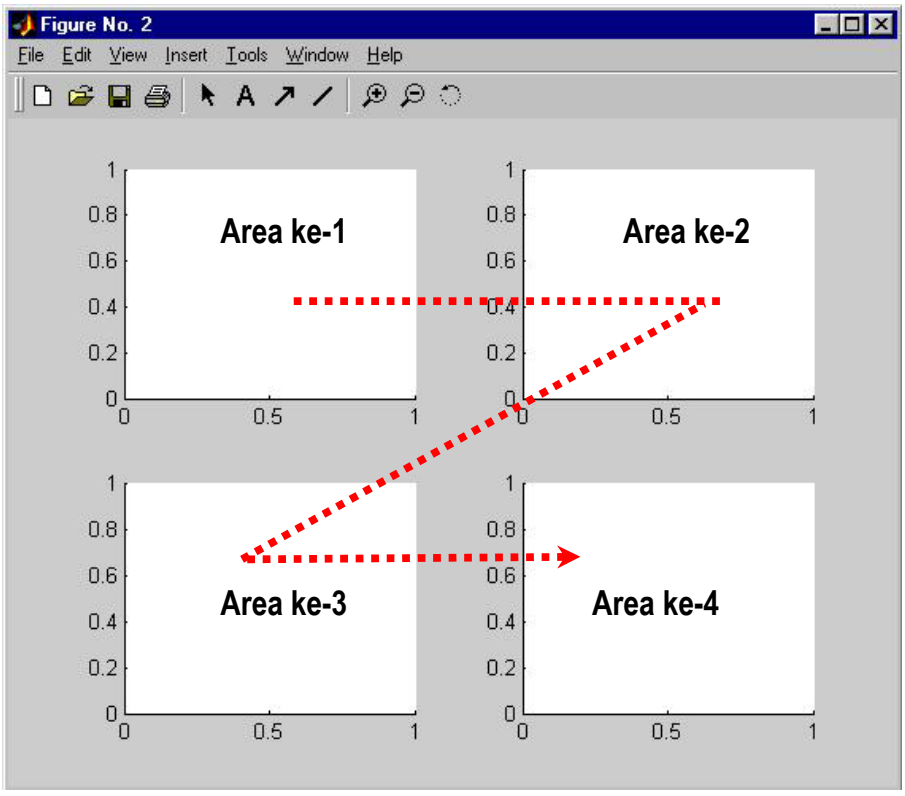
5.2 Lebih Jauh Mengenai Plot

Anda mungkin ingin memplot beberapa fungsi dalam beberapa *figure window* yang terpisah, atau membagi satu *window* menjadi sejumlah area plot, ataupun mengatur *properties* dari plot yang akan digambar. Beberapa *command* di bawah ini bisa digunakan untuk tujuan tersebut.

Tabel 5. 3

figure	menciptakan *figure window* baru yang kosong dan siap untuk di-plot
figure(k)	untuk 'menduduki' *figure window* nomor-k
subplot(m,n,k)	membagi figure window menjadi m-baris × n-kolom area plot yang terpisah, dan menduduki area ke-k
clf	"clear figure", mengosongkan *figure window* yang sedang 'diduduki'

Misalkan *figure window* berikut dibagi menjadi 2-baris × 2-kolom dengan **subplot**. Perhatikan urutan nomor area dari kiri-atas ke kanan-bawah.



Gambar 5. 4 Pembagian area plot dengan "subplot"

Tabel 5. 3 (lanjutan)

plot(x,y,'string') menciptakan plot 2-dimensi dari vektor **x** versus vektor **y**, dengan *property* yang ditentukan oleh **string**, sebagai berikut:

Warna		Jenis Garis		Jenis Point	
b	biru	**-**	utuh	**.**	titik
g	hijau	**:**	titik-titik	**o**	lingkaran
r	merah	**-.**	titik-strip	**x**	tanda ×
c	biru muda	**--**	putus-putus	**+**	tanda +
m	ungu			*****	tanda *
y	kuning			**s**	bujur sangkar
k	hitam			**d**	permata
w	putih			**v**	segitiga ke bawah
				^	segitiga ke atas
				<	segitiga ke kiri
				>	segitiga ke kanan
				p	segilima
				h	segienam

Misalkan:

plot(x,y,'r-') memplot **x** versus **y** dengan garis utuh warna merah

plot(x,y,'k*') menempatkan tanda * warna hitam untuk setiap titik **x** versus **y**.

plot(x,y,'g--s') memplot dengan garis putus-putus warna hijau dan menempatkan tanda bujur sangkar di setiap titik **x** versus **y**.

Perlu diingat bahwa **'string'** dalam plot bersifat opsional. Apabila tidak dituliskan maka digunakan garis utuh warna biru.

Tabel 5. 3 (lanjutan)

plot(x1,y1,'string1',x2,y2,'string2',x3,y3,'string3', ...)

menciptakan sejumlah plot sekaligus dalam satu area plot: **x1** versus **y1** dengan property **string1**, **x2** versus **y2** dengan property **string2**, dan seterusnya

legend('ket1','ket2','ket3', ...)

menambahkan legenda ke dalam plot yang telah dibuat; **ket1** untuk plot pertama, **ket2** untuk plot kedua, dan seterusnya

axis off	menghilangkan tampilan sumbu koordinat pada plot
axis on	menampakkan kembali sumbu koordinat

axis([x_awal x_akhir y_awal y_akhir])
membuat tampilan area plot pada batas-batas nilai x = **x_awal** hingga **x_akhir**, dan nilai y = **y_awal** hingga **y_akhir**

axis equal	mengubah skala sumbu-x dan sumbu-y menjadi sama
axis square	mengubah bentuk area plot menjadi bujur sangkar

Berbagai fungsi yang berkaitan dengan plot di atas, berlaku pula untuk plot diskrit, plot logaritmik dan plot dalam koordinat polar.

Tabel 5. 4

stem(...)	sama dengan **plot(...)**, tetapi menampilkan **y** sebagai data diskrit
semilogy(...)	sama dengan **plot(...)**, kecuali sumbu-y menggunakan skala logaritmik (basis 10)
semilogx(...)	sama dengan **plot(...)**, kecuali sumbu-x menggunakan skala logaritmik
loglog(...)	sama dengan **plot(...)**, tetapi sumbu-x dan sumbu-y menggunakan skala logaritmik

polar(theta,rho,'string')
membuat plot dalam koordinat polar dari sudut **theta** (satuan radian) versus radius **rho**, dengan *property* ditentukan oleh **string**

Kini saatnya mencoba berbagai *command* di atas dalam contoh berikut ini.

Pertama, kita akan mencoba memplot kurva eksponensial negatif seperti pada contoh subbab 5.1 secara lebih efisien.

```
>> clear
>> x=linspace(0,5,500);
>> y1=exp(-x); y2=exp(-0.5*x); y3=exp(-0.25*x);
>> y4=exp(-0.1*x);
>> plot(x,y1,x,y2,x,y3,x,y4)
>> grid on
>> xlabel('sumbu-x'), ylabel('sumbu-y')
```

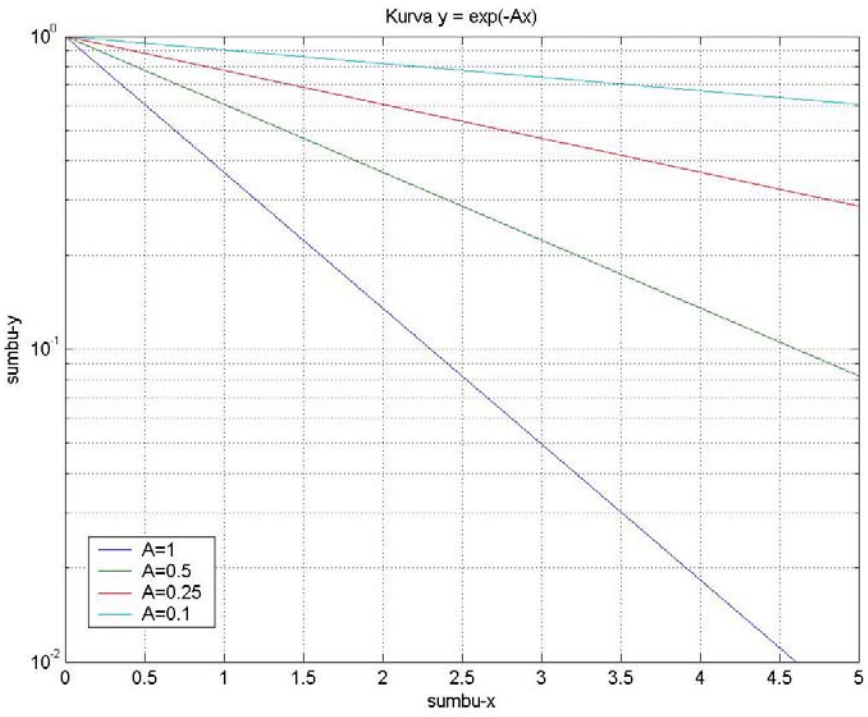
```
>> title('Kurva y = exp(-Ax)')
>> legend('A=1','A=0.5','A=0.25','A=0.1')
```

Kemudian, kita coba memplot kurva tersebut dalam skala semilogaritmik

```
>> figure
>> semilogy(x,y1,x,y2,x,y3,x,y4)
>> grid on
>> xlabel('sumbu-x'), ylabel('sumbu-y')
>> title('Kurva y = exp(-Ax)')
>> legend('A=1','A=0.5','A=0.25','A=0.1')
```

Misalkan kita ingin menyempitkan area plot pada y = 1 hingga 10^{-2} saja, maka:

```
>> axis([0 5 1e-2 1])
```

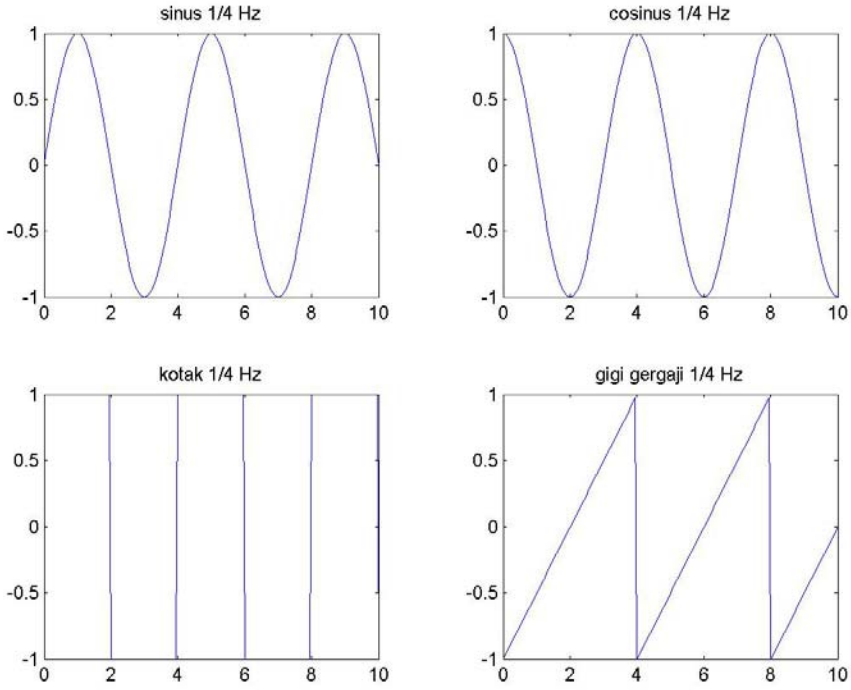


Gambar 5. 5 Contoh plot semi-logaritmik

Dalam contoh kedua, kita akan memplot gelombang sinus, cosinus, kotak, dan gigi gergaji dengan melibatkan *command* **subplot**.

```
>> figure
>> t=0:0.05:10;
```

```
>> sinus=sin(2*pi*0.25*t);
>> cosinus=cos(2*pi*0.25*t);
>> kotak=square(2*pi*0.25*t);
>> gigi=sawtooth(2*pi*0.25*t);

>> subplot(2,2,1);
>> plot(t,sinus), title('sinus 1/4 Hz')

>> subplot(2,2,2);
>> plot(t,cosinus), title('cosinus 1/4 Hz')

>> subplot(2,2,3);
>> plot(t,kotak), title('kotak 1/4 Hz')

>> subplot(2,2,4);
>> plot(t,gigi), title('gigi gergaji 1/4 Hz')
```



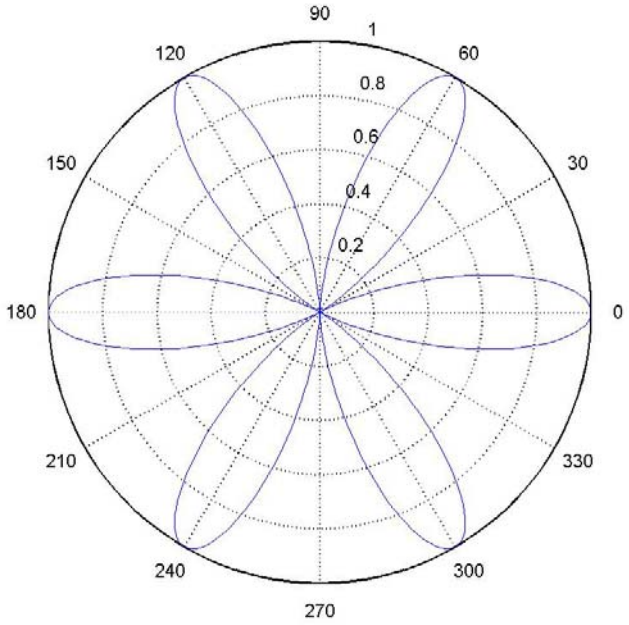
Gambar 5. 6 Contoh penggunaan subplot

Dalam contoh ketiga, kita akan mencoba memplot suatu fungsi matematis dalam koordinat polar. Diinginkan plot fungsi:

$\rho = sin^2(3\theta)$

dalam MATLAB dituliskan

```
>> figure
>> theta=linspace(0,2*pi,500);
>> rho=(cos(theta.*3)).^2;
>> polar(theta,rho);
```



Gambar 5. 7 Contoh plot dengan *command* "polar"

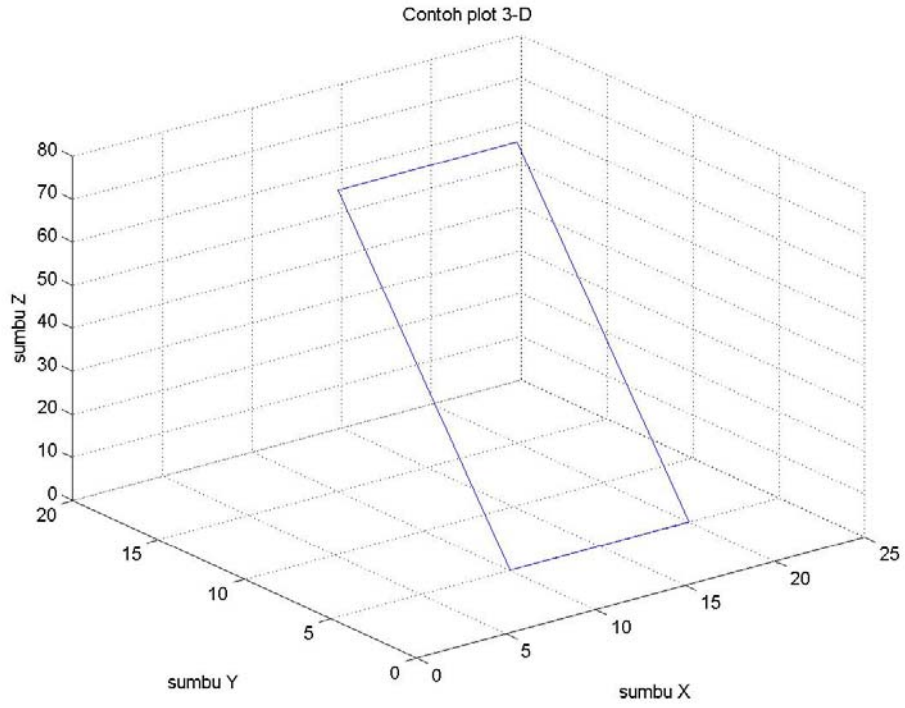
5.3 Plot 3-Dimensi

Dalam subbab ini akan dibahas tiga macam plot 3-dimensi: plot garis, plot permukaan (*surface*), dan plot kontur.

5.3.1 Plot Garis

Mari kita mulai dengan plot garis di dalam ruang 3-dimensi. Ini mirip dengan plot 2-dimensi, tetapi kali ini kita gunakan *command* **plot3(...)**, dan dibutuhkan vektor **z**, untuk dimensi ketiga.

```
>> X = [10 20 20 10 10];
>> Y = [5 5 15 15 5];
>> Z = [0 0 70 70 0];
>> plot3(X,Y,Z); grid on;
>> xlabel('sumbu X'); ylabel('sumbu Y');
>> zlabel('sumbu Z');
>> title ('Contoh plot 3-D');
>> axis([0 25 0 20 0 80])
```

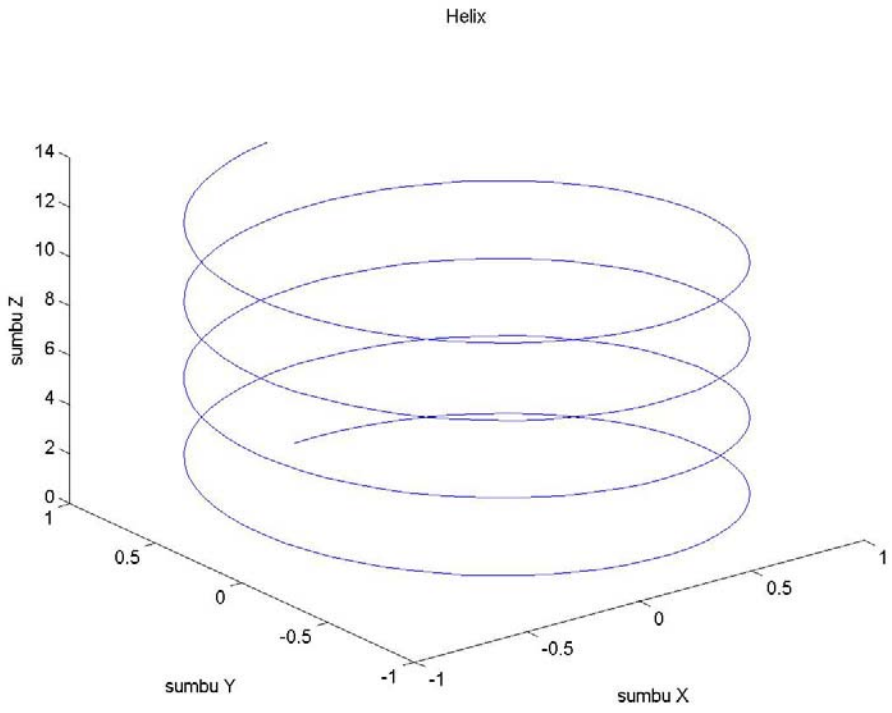


Gambar 5. 8 Contoh plot 3-dimensi dengan *command* "plot3"

Perhatikan bahwa *command* **label**, **title**, **grid**, **axis**, **hold**, dan **subplot** juga berlaku di sini. Anda juga bisa merotasi gambar 3-dimensi tersebut dengan cara men-klik ikon rotate dan *dragging* mouse di atas gambar.

Sekarang kita coba contoh yang lain untuk menggambarkan helix.

```
>> t=0:0.1:25;
>> X=sin(t); Y=cos(t); Z=0.5*t;
>> plot3(X,Y,Z)
>> xlabel('sumbu X'); ylabel('sumbu Y');
>> zlabel('sumbu Z');
>> title ('Helix');
```



Gambar 5. 9 Contoh penggunaan "plot3"

5.3.2 Plot Permukaan

Sementara itu, untuk plot permukaan (*surface*) dalam ruang 3-dimensi digunakan *command* **mesh** atau **surf**. Contoh berikut ini menggambarkan fungsi dua variabel $z = x^2 + y^2$.

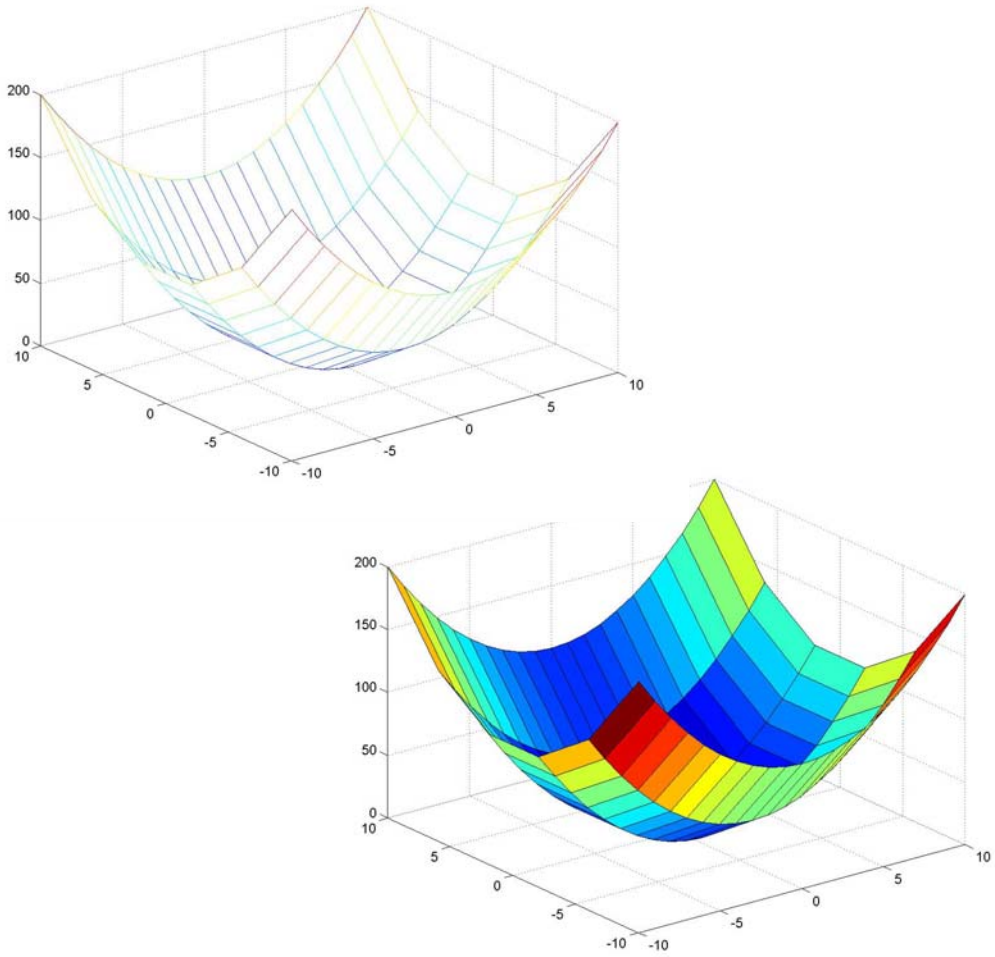
Caranya ialah:
1) Definisikan batas-batas nilai *x* dan *y* yang akan diplot
2) Gunakan *command* **meshgrid** untuk "mengisi" bidang-XY dengan jalinan titik
3) Hitunglah fungsi 3-dimensi untuk jalinan titik tersebut
4) Buatlah plot dengan *command* **mesh** atau **surf**.

Sebagai contoh:

```
>> batas_x = -10:1:10; batas y = -10:4:10;
>> [X,Y] = meshgrid(batas_x,batas_y);
>> Z = X.^2 + Y.^2;
>> mesh(X,Y,Z);
```

Kini Anda mendapatkan plot 3-dimensi. Kini cobalah

```
>> surf(X,Y,Z);
```



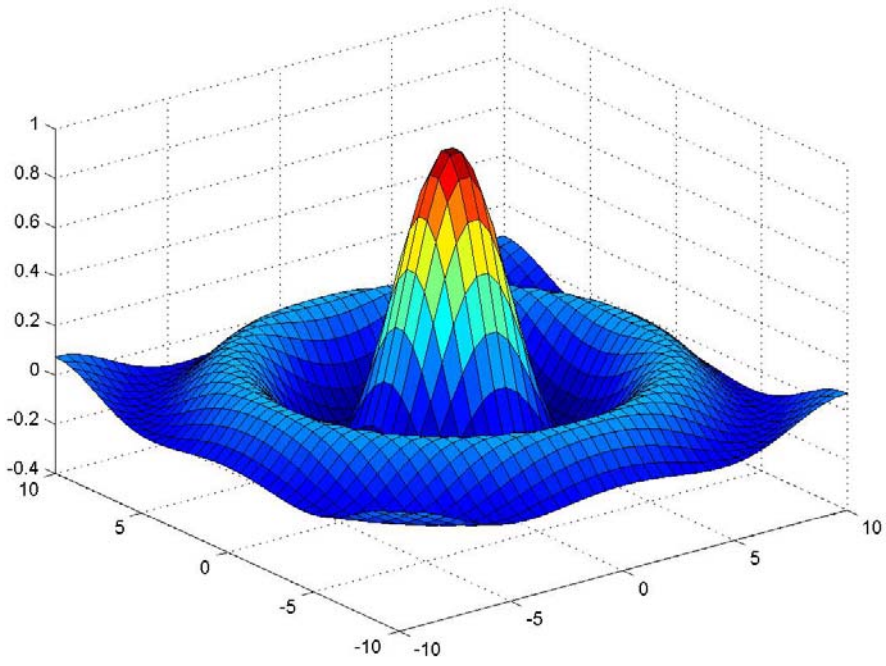
Gambar 5. 10 Hasil plot dengan "mesh" dan "surf"

Amatilah perbedaan hasil antara **mesh** dan **surf** ! Anda juga bisa menambahkan "label" dan "title" seperti plot pada umumnya.

Sekarang kita coba contoh yang lain untuk memplot fungsi 3-dimensi

$$z = \frac{\sin(r)}{r}, \qquad \text{di mana } r = \sqrt{x^2 + y^2}\ .$$

```
>> x = linspace(-10,10,40); y = x;
>> [X,Y] = meshgrid(x,y);
>> R = sqrt(X.^2+Y.^2);
>> Z = sin(R)./(R+eps);
>> surf(X,Y,Z);
```



Gambar 5. 11 Plot 3-dimensi dari fungsi sin(r) / r

di sini kita menggunakan variabel **eps**, untuk mencegah perhitungan 0/0 ketika **R** = 0.

5.3.3 Plot Kontur

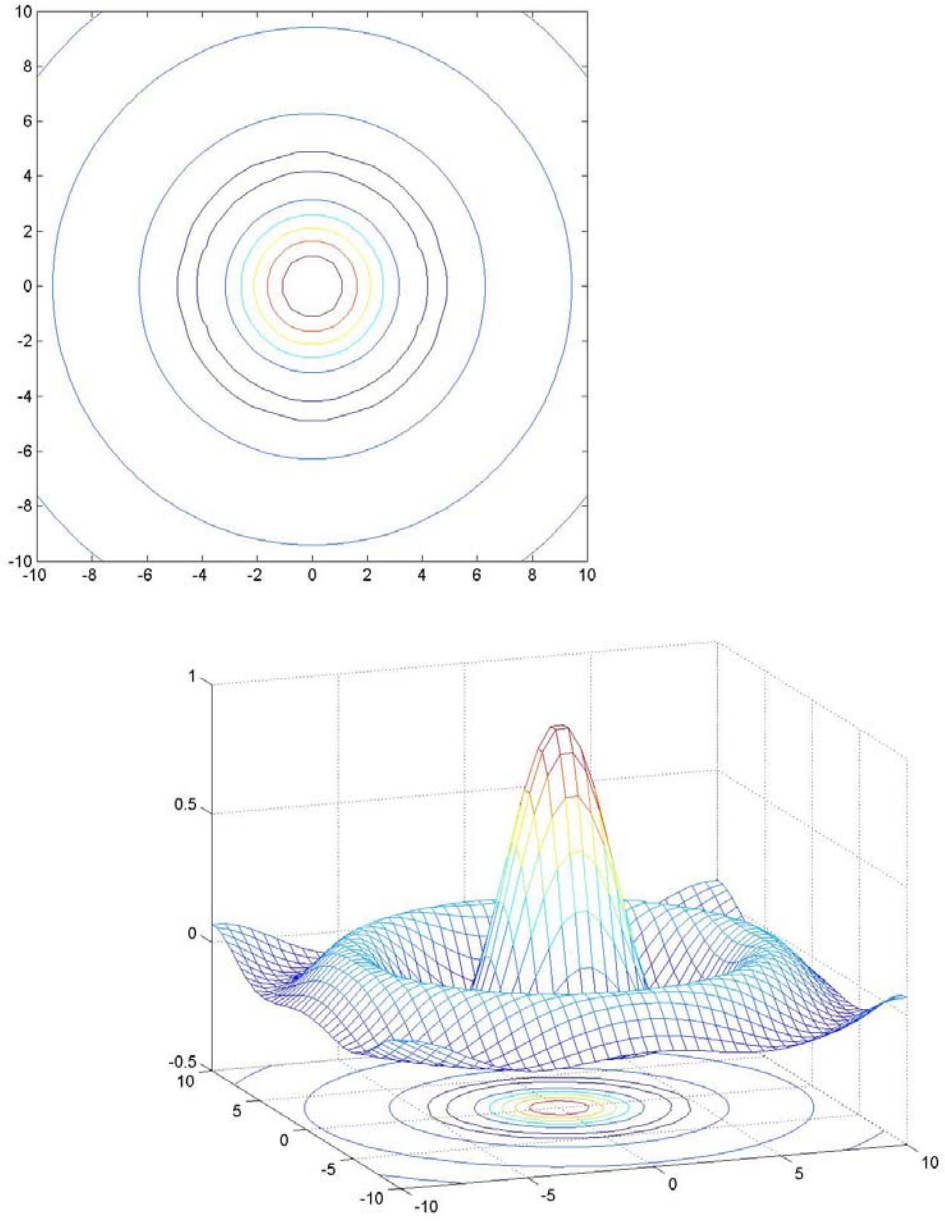
Fungsi dua variabel, misalkan $z = f(x,y)$ bisa kita gambarkan konturnya dalam dua dimensi dengan *command* berikut ini:

Tabel 5. 5

contour(X,Y,Z)	menggambar kontur dari nilai di **Z** dengan 10 level. Elemen **Z** diterjemahkan sebagai level-level di atas bidang (*x*,*y*)
C = contour(X,Y,Z)	menghitung matriks kontur **C**
contour(X,Y,Z,n)	menggambar kontur dengan *n* level
contour(... , 'string')	menggambar kontur dengan *property* yang ditentukan oleh string (lihat Tabel 5.3)
clabel(C)	menuliskan angka pada garis-garis kontur untuk menunjukkan level
meshc(X,Y,Z)	menggambar permukaan seperti pada *command* **mesh**, dan juga memplot kontur pada dasar grafik.

Mari kita gambarkan kontur dari fungsi sin(*r*)/*r* di atas, lalu bandingkan dengan plot permukaannya:

```
>> figure; contour(X,Y,Z);
>> figure; meshc(X,Y,Z);
```



Gambar 5. 12 Contoh plot kontur

5.4 Suara

Untuk menyuarakan suatu vektor, ataupun membaca dan menyimpan file audio berformat WAV, digunakan *command* berikut ini:

[x,Fs] = wavread('nama_file')
membaca file WAV dan menyimpannya dalam vektor **x**, serta mengembalikan frekuensi sampling **Fs** dari file tersebut. *Command* ini juga bisa membaca file WAV multi kanal

wavwrite(x,Fs,'nama_file')
menuliskan file WAV dari vektor **x** dengan frekuensi sampling **Fs**

sound(x,Fs)
menyuarakan vektor **x** dengan frekuensi sampling **Fs**

soundsc(x,Fs)
sama seperti sintaks sebelumnya, namun vektor x terlebih dahulu diskalakan pada selang $-1 \leq \mathbf{x} \leq +1$

File yang akan dibaca harus tersimpan di direktori **Matlab\work**, atau Anda harus merinci drive, direktori dan nama file jika file tersimpan di direktori lain.

Sebagai gambaran, marilah kita dengarkan suara berikut ini.

Pertama, suara pitch 400 Hz berdurasi 2 detik.

```
>> Fs=8000;          %frekuensi sampling 8 kHz
>> t=0:1/Fs:2;       %sinyal berdurasi 2 detik
>> frek=400;         %frekuensi sinyal 400 Hz
>> m=cos(2*pi*frek*t);
>> sound(m,Fs);             %suara dari m
>> wavwrite(m,Fs,'tone_400Hz.wav'); ...
%Menyimpan vektor m ke dalam file
```

Berikutnya, memperdengarkan suara helikopter yang ada di file **heli.wav**.

```
>> [x,Fs]=wavread('heli.wav'); %Membaca file heli.wav
>> sound(x,Fs);             %Suara helikopter
```

Soal Latihan

1. Gambarkan kurva $y = x^4 - 9x^2$ pada rentang $-6 \le x \le 6$. Buatlah inkremen x cukup kecil sehingga kurva terlihat mulus.

2. Gambarkan kurva-kurva berikut pada rentang $-10 \le x \le 10$ dalam satu *figure* sekaligus!
   $y = \sqrt{100 + x^2}$ $y = \sqrt{100 + 2x^2}$ $y = \sqrt{100 + 4x^2}$
   $y = \sqrt{100 + 16x^2}$

3. Suatu filter memiliki respon frekuensi sebagai berikut:
   $\frac{V_o}{V_i} = \frac{1}{1 + j2\pi fF}$ di mana F = 4kHz ialah frekuensi *cut-off* dari filter. Buatlah plot semilogaritmis pada sumbu frekuensi: respon amplituda, $\left|\frac{V_o}{V_i}\right|$ versus f, dan plot respon fasa, $\angle\left(\frac{V_o}{V_i}\right)$ versus f, pada rentang frekuensi 0 hingga 50 kHz.
   Gambarkan kedua plot tadi pada satu *window* saja, setengah bagian atas untuk plot amplituda, dan setengah bagian bawah untuk plot fasanya.

4. Sebuah antena diketahui memiliki pola radiasi dalam koordinat polar sebagai berikut:
   $$U(\phi) = \begin{cases} \cos^3\phi & -\frac{\pi}{2} \le \phi \le \frac{\pi}{2} \\ 0 & \textit{selainnya} \end{cases}$$
   Gambarkan pola radiasi ini!

5. Gambarkan kurva berikut ini di dalam ruang 3-D:
   $$\left.\begin{aligned} x &= 1 + \cos t \\ y &= 2 + \sin t \\ z &= 1 - \cos 2t \end{aligned}\right\} 0 \le t \le 2\pi$$

6. Plot fungsi dua variabel berikut ini: $z = x^2 - y^2$, untuk rentang $-5 \le x \le 5$, $-5 \le y \le 5$

7. Plot kontur dari fungsi dua variabel berikut ini:
   $f(x, y) = \cos x \sin 2y$, untuk $0 \le x \le 4\pi$, $0 \le y \le 4\pi$

8. Buatlah suatu file suara WAV berisi urut-urutan tone DO-RE-MI-FA-SOL-LA-TI-DO dengan frekuensi berikut ini:

DO	**RE**	**MI**	**FA**	**SOL**	**LA**	**TI**	**DO**
262	294	330	349	392	440	495	524

BAB 6

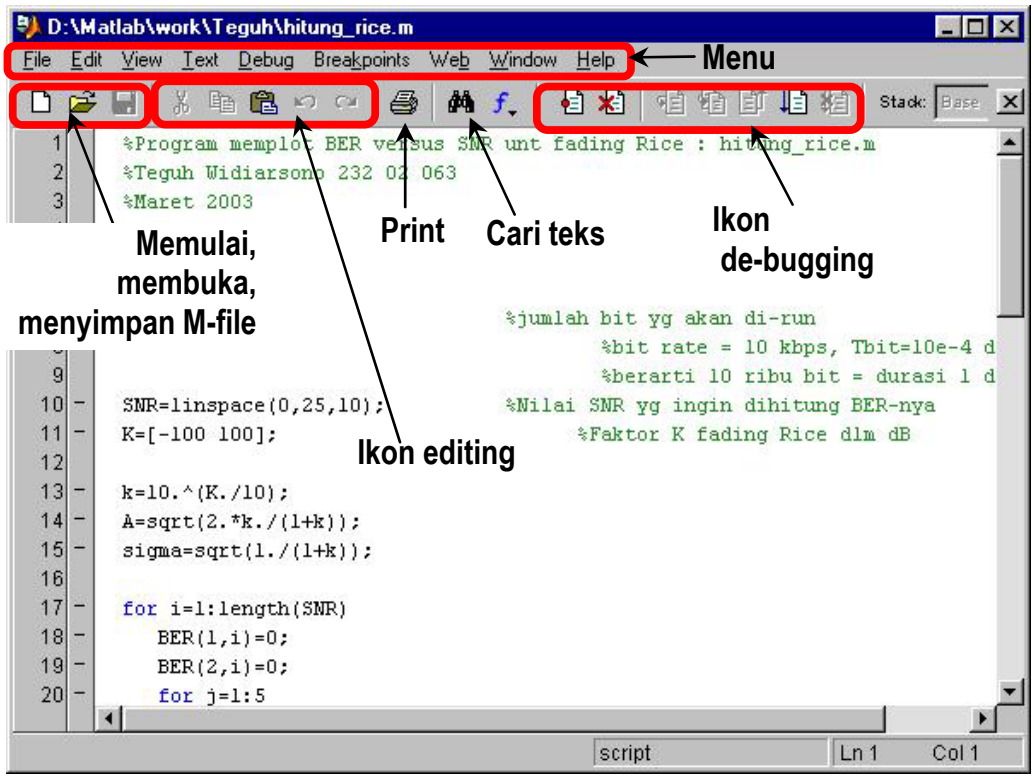
M-FILE DAN PEMROGRAMAN MATLAB

Pada bab-bab yang lalu, Anda telah belajar berinteraksi dengan MATLAB menggunakan *command window*. Sekarang, katakanlah Anda harus mempergunakan sederetan *command* secara berulang-ulang di dalam sesi MATLAB yang berbeda. Akan sangat repot jika Anda harus mengetikkan *command* tersebut secara manual di *command window* setiap kali Anda butuhkan. Namun dengan M-file, deretan *command* tersebut bisa Anda simpan dalam bentuk skrip teks. Kapan saja Anda butuhkan, skrip tersebut bisa dijalankan/dieksekusi secara otomatis dengan cara mengetikkan nama M-file yang bersangkutan di *command window*.

Kali ini kita akan belajar mengenal M-file dengan contoh sederhana. Namun demikian perlu diketahui bahwa MATLAB sebenarnya merupakan bahasa pemrograman umum, seperti halnya Basic, C, Java, Pascal, Fortran, dll. Sehingga dalam bab ini kita akan menitikberatkan pada pelajaran pemrograman komputer.

6.1 Membuat M-File

Untuk menuliskan skrip M-file, Anda bisa mulai dengan membuka file baru. Caranya ialah melalui menu di *main window*: **File→Open** atau **File→New→M-file**; atau dengan mengklik ikon yang ada di jendela utama. Sebuah jendela editor akan terbuka seperti gambar berikut ini.



Gambar 6. 1 Jendela editor M-file

Dengan editor ini, kita bisa membuka sejumlah M-file, melakukan editing, ataupun mencoba menjalankannya dan melakukan debuging (mencari kesalahan di dalam skrip).

Sementara itu, untuk menyimpan M-file, Anda bisa lakukan dengan menu: **File→Save** atau **File→Save As**; ataupun dengan mengklik ikon yang ada.

Namun demikian, sebenarnya Anda juga bisa menuliskan M-file dengan sebarang editor teks, seperti MS Word, Notepad, dll.; yang penting Anda menyimpan file tersebut dengan ekstensi ***.m**.

6.2 M-File Sebagai Skrip Program

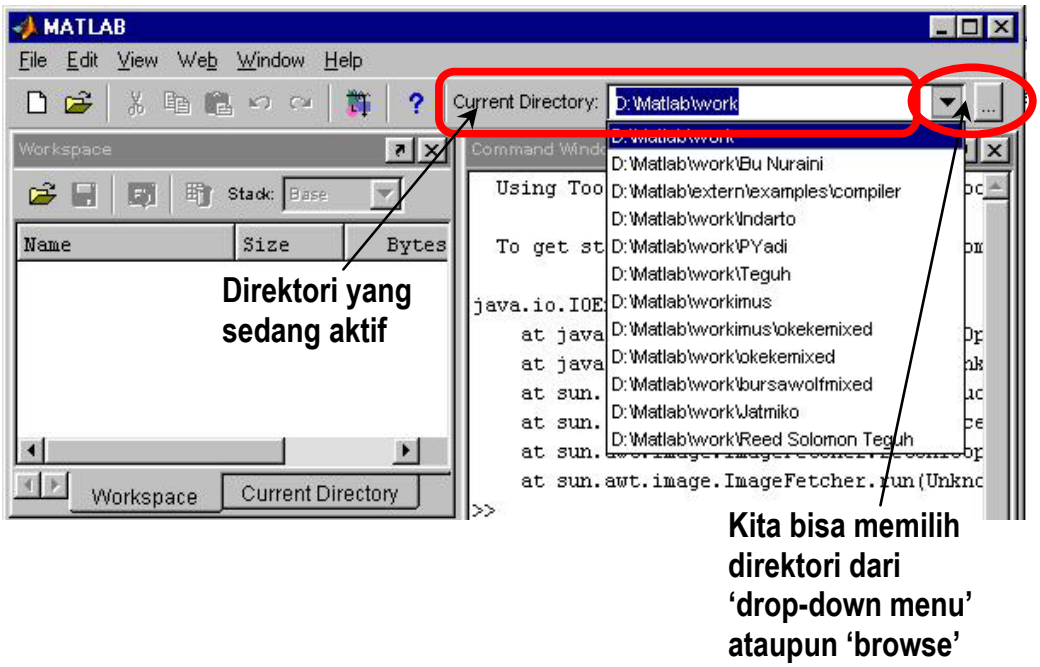
Pada bagian ini, kita akan menggunakan M-file untuk menjalankan sederetan *command* yang kita tuliskan sebagai skrip. Mari kita mulai dengan skrip sederhana untuk menghitung rata-rata dari lima bilangan. File ini kita namakan **rata_rata.m**.

Bukalah M-file baru lalu ketikkan skrip berikut ini.

```
% Program sederhana untuk menghitung
% rata-rata 5 bilangan:
% rata_rata.m
a = 50;
b = 100;
c = 150;
d = 200;
e = 250;

% Menghitung dan menampilkan rata-rata
hasil = (a + b + c + d + e)/5;
hasil
```

Teks yang diawali tanda "%" menunjukkan komentar, dan tidak akan dieksekusi oleh MATLAB.

Simpanlah file ini di dalam direktori **Matlab\work** dengan nama **rata_rata.m**. Sekarang cobalah jalankan dari *command window*. Sebelumnya pastikan bahwa direktori menunjuk ke **Matlab\work**. Perhatikan "Current Directory" yang ada di jendela utama MATLAB. Kita bisa mengubah direktori yang sedang aktif melalui *drop-down menu* ataupun melalui *browse*.



Gambar 6. 2 Memilih direktori untuk menjalankan M-file

```
>> clear
>> rata_rata
hasil =
       150
>> whos
Name        Size          Bytes  Class

  a           1x1             8  double array
  ans         1x1             8  double array
  b           1x1             8  double array
  c           1x1             8  double array
  d           1x1             8  double array
  e           1x1             8  double array
  hasil       1x1             8  double array

Grand total is 7 elements using 56 bytes
```

Perhatikan bahwa:

- Di dalam M-file, setiap *command* diakhiri dengan titik-koma supaya hasil perhitungan di tiap baris tidak ditampilkan di *command window*. Kecuali pada hasil perhitungan yang ingin kita tampilkan, tidak diakhiri titik-koma.
- Variabel yang didefinisikan di dalam M-file akan disimpan oleh MATLAB ketika M-file telah dieksekusi.

Di dalam editor, skrip yang kita tuliskan akan memiliki warna tertentu:

- hijau untuk komentar
- hitam untuk variabel dan command
- biru untuk statement pemrograman.

Sekarang, marilah kita mencoba M-file lain untuk menghitung sisi miring suatu segi tiga siku-siku dengan formula phytagoras, menghitung luasnya, dan kelilingnya.

```
% Program menghitung segi-3 siku-siku: segi3.m
% Untuk menghitung sisi miring, luas, dan keliling

% Mendefinisikan sisi siku-siku segitiga
Sisi_A = 3;
Sisi_B = 4;

% Menghitung sisi miring
Sisi_C = sqrt(Sisi_A^2 + Sisi_B^2)

% Menghitung luas segitiga
Luas = 1/2* Sisi_A * Sisi_B
```

```
% Menghitung keliling
Keliling = Sisi_A + Sisi_B + Sisi_C
```

Lalu simpan dengan nama **segi3.m**.

Sekarang kita panggil M-file tersebut

```
>> segi3
Sisi_C =
       5
Luas =
       6
Keliling =
       12
```

Sekarang Anda bisa mencoba sendiri membuat program yang lebih menantang, seperti menghitung dan memplot fungsi 2 ataupun 3-dimensi dengan M-file.

6.3 M-File Sebagai Fungsi

Sebagai skrip program, jika kita ingin mengubah/mengatur parameter masukan program, maka harus kita lakukan di dalam editor. Padahal seringkali kita harus menjalankan satu program/algoritma berulang kali dengan nilai masukan yang berbeda-beda, misalkan dalam proses iterasi atau optimasi. Untuk keperluan ini, kita bisa menuliskan M-file sebagai suatu fungsi spesifik sesuai kebutuhan kita.

Dalam setiap fungsi terdapat tiga unsur:

1. Parameter masukan; dalam hal ini kita sebut sebagai "argumen input". Jumlah parameter (argumen) tersebut bisa sebarang (satu, dua, sepuluh, atau tidak ada argumen input sama sekali). Jenis argumen pun sebarang (variabel, bilangan ataupun teks).
2. Proses di dalam program; berupa sederetan command untuk menjalankan suatu algoritma tertentu.
3. Parameter keluaran; atau "argumen output" yang jumlah dan jenisnya sebarang.

Deklarasi fungsi di M-file harus dilakukan pada baris awal dengan sintaks:

```
function [argumen output] = nama_fungsi(argumen input)
```

Sebagai contoh awal, kita akan membuat fungsi untuk menghitung sisi miring, luas, dan keliling segitiga; seperti program yang ada pada contoh sebelumnya.

```
%Fungsi untuk menghitung segi-3 siku-siku: segitiga.m
%Untuk menghitung sisi miring, luas, dan keliling

function [Sisi_C,Luas,Kll] = segitiga(Sisi_A,Sisi_B)

% Menghitung sisi miring
Sisi_C = sqrt(Sisi_A^2 + Sisi_B^2);

% Menghitung luas segitiga
Luas = 1/2* Sisi_A * Sisi_B;

% Menghitung keliling
Kll = Sisi_A + Sisi_B + Sisi_C;
```

Lalu simpan dengan nama **"segitiga.m"**.

Sekarang Anda panggil fungsi tersebut.

```
>> clear
>> [Hyp,Area,Circum]=segitiga(12,16)
Hyp =
       20
Area =
       96
Circum =
       48
```

Dari contoh sederhana tersebut, ada beberapa hal yang perlu kita perhatikan:

- Dalam fungsi **segitiga**, terdapat dua argumen input (**Sisi_A**, **Sisi_B**), dan tiga argumen output (**Sisi_C**, **Luas**, **Kll)**.
- Ketika dipanggil di *command window*, kita bisa menggunakan nama argumen input/output yang berbeda dengan di M-file, namun urutannya tidak berubah. Di dalam contoh, argumen **Sisi_A** dan **Sisi_B** kita isi dengan bilangan, sementara argumen **Sisi_C**, **Luas**, dan **Keliling** kita panggil dengan **Hyp**, **Area**, dan **Circum**.

Sekarang kita lihat dengan command **whos**:

```
>> whos
Name          Size          Bytes  Class

Area          1x1               8  double array
Circum        1x1               8  double array
Hyp           1x1               8  double array

Grand total is 3 elements using 24 bytes
```

Terlihat bahwa variabel yang dideklarasikan di dalam fungsi tidak disimpan, melainkan dimusnahkan ketika suatu fungsi selesai dijalankan. Yang ada di sana hanyalah variabel yang telah dideklarasikan di *command window* untuk menyimpan nilai output. Hal ini merupakan salah satu perbedaan utama antara skrip program dengan fungsi.

Penting!

Ketika membuat fungsi dengan M-file, nama file harus sama dengan nama fungsi yang dideklarasikan dalam sintaks **`function ...`**
Aturan penamaan M-file sama dengan penamaan variabel! Lihat kembali aturan tersebut di subbab 2.2

Perlu diperhatikan bahwa fungsi yang telah kita buat pada dasarnya sama dengan fungsi yang telah ada di MATLAB, semisal fungsi **sin(x)** ataupun **sqrt(x)**. Misalkan kita memanggil fungsi tanpa menyebutkan argumen output, maka keluaran akan disimpan di **ans**.

6.4 Display dan Input

Adakalanya kita membutuhkan interaksi dengan pengguna program untuk memasukkan parameter tertentu di awal/tengah program. Dalam hal ini kita bisa pergunakan cara sederhana dengan *command* **input**. Sementara *command* **disp** digunakan untuk menampilkan teks di layar.

Misalkan kita akan membuat program untuk menghitung jumlah kombinasi team basket yang mungkin dari sejumlah mahasiswa.

```
% Program menghitung kombinasi : hit_komb.m
% untuk menghitung jumlah kombinasi
% dari sejumlah populasi

% Menampilkan judul program
clc;
disp('Menghitung Kombinasi');
disp('---------------------');

% Meminta masukan dari user
n = input('Berapa jumlah mahasiswa yang ada? : ');
r = input('Berapa jumlah personel satu team? : ');

% Menghitung kombinasi
kombinasi = factorial(n)/factorial(r)/factorial(n-r);

% Menampilkan keluaran
disp('Jumlah kombinasi yang ada = ',kombinasi);
```

Kita coba jalankan program tersebut:

```
>> hit_komb

Menghitung Kombinasi
---------------------
Berapa jumlah mahasiswa yang ada? : 8
Berapa jumlah personel satu team? : 5
Jumlah kombinasi yang ada =
    56
```

6.5 Control Statement

Seperti halnya bahasa program pada umumnya, kita bisa mengendalikan arah program dengan berbagai cara, berupa percabangan arah program berdasarkan kondisi tertentu, ataupun loop (perhitungan berulang) ketika kita melakukan iterasi.

6.5.1 Statement if … elseif … else … end

Ini merupakan statement untuk percabangan program berdasarkan satu/beberapa kondisi tertentu. Sintaks yang digunakan dalam MATLAB meliputi:

```
if kondisi
       Command yang dijalankan jika kondisi dipenuhi
end
```

```
if kondisi
       Command yang dijalankan jika kondisi dipenuhi
else
       Dijalankan jika kondisi tidak dipenuhi
end
```

```
if kondisi1
       Command yang dijalankan jika kondisi1 dipenuhi
elseif kondisi2
       Dijalankan jika kondisi2 dipenuhi
elseif kondisi3
       Dijalankan jika kondisi3 dipenuhi
elseif ...
       ...dst...
else
       Dijalankan jika kondisi manapun tidak dipenuhi
end
```

Selain itu, dimungkinkan pula membuat pernyataan **if** di dalam pernyataan yang lain (disebut *nested*-if), misalkan:

```
if kondisi1
       command1
       if kondisiA
              commandA
       else
              commandB
       end
else
       command2
end
```

Penting!

jangan keliru menuliskan **elseif** dan **else if**, karena keduanya berbeda. Yang pertama untuk menguji kondisi alternatif setelah kondisi di **if** terdahulu tak dipenuhi; tetapi yang kedua berarti **nested-if**.

6.5.2 Statement switch … case

Sebagai alternatif dari statement **if … elseif … else … end**, kita bisa menggunakan statement **switch**. Sintaksnya ialah:

```
switch nama_variabel
case{kondisi1,kondisi2,...}
       Dijalankan jika kondisi1 atau kondisi2 dst...
       dipenuhi
case{kondisiA,kondisiB,...}
       Dijalankan jika kondisiA atau kondisiB dst...
       dipenuhi
case{kondisiX,kondisiY,...}
       Dijalankan jika kondisiX atau kondisiY dst...
       dipenuhi
case{...}
       ...dst...
default
       Dijalankan jika kondisi manapun tidak dipenuhi
end
```

6.5.3 Statement for … end

Statement ini digunakan untuk loop/perhitungan berulang. Sintaks yang digunakan dalam MATLAB ialah:

```
for variabel = nilai_awal : inkremen : nilai_akhir
       Command untuk dijalankan
end
```

Adapun sintaks yang digunakan untuk membatasi loop mirip dengan yang kita pakai untuk membuat deret (lihat kembali subbab 3.5). Misalkan untuk menampilkan bilangan kelipatan 3 dari 30 sampai 100.

```
for k = 30:3:100
       k
end
```

Hasilnya ialah:

```
k =
    30
k =
    33
k =
    ...
k =
    99
```

Sementara untuk nilai inkeremen = 1, cukup dituliskan nilai awal dan akhir. Misalkan untuk mendaftar bilangan bulat dari –10 hingga 10 dan menyimpannya dalam satu vektor.

```
Vektor=[];
for k = -10:10      %dalam hal ini inkremen = 1
      Vektor = [Vektor k];
end
Vektor
```

Menghasilkan:

```
Vektor =
Columns 1 through 13
-10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2

Columns 14 through 21
3   4   5   6   7   8   9  10
```

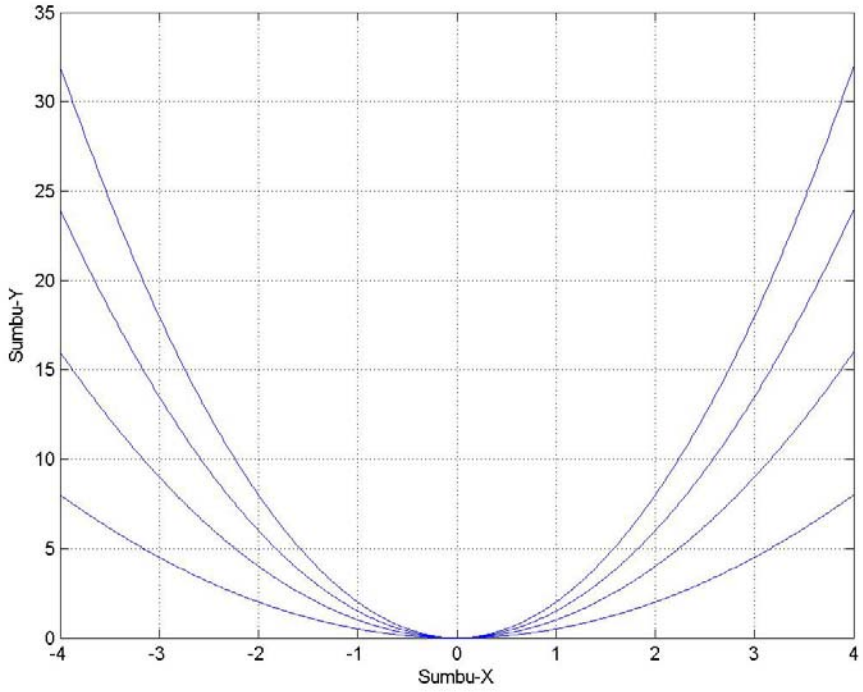
Atau untuk memplot kurva parabola:

$y = Ax^2$

dengan berbagai nilai parameter *A*, yaitu 0,5 , 1 , 1,5 , dan 2.
Dalam hal ini indeks vektor **A** kita iterasi dari 1 hingga indeks terakhir.

```
figure;
x = linspace(-4,4,500);    % mendefinisikan nilai x
A = 0.5:0.5:2;             % mendefinisikan vektor A
for i = 1:length(A)
      y = A(i)* x.^2;
      plot(x,y);
      hold on;
end
grid on;
```

Menghasilkan:



Gambar 6. 3 Contoh plot 4 kurva parabola dengan "for"

Perhatikan bahwa setiap selesai satu loop, **variabel** (dalam contoh di atas ialah `i`) akan otomatis mengalami inkremen. Demikian seterusnya hingga **nilai_akhir** (yaitu `length(A)`) tercapai dan program dilanjutkan ke baris selanjutnya.

6.5.4 Statement while … end

Alternatif dari sintaks loop ialah berikut ini

```
while kondisi
      Command untuk dijalankan jika kondisi dipenuhi
end   %keluar dari loop jika kondisi tidak dipenuhi
```

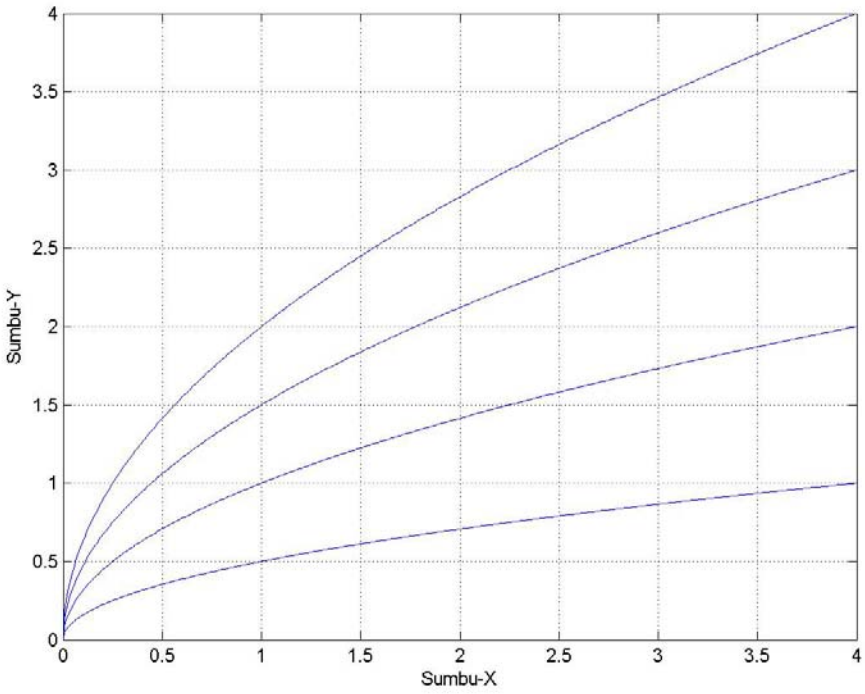
Misalkan untuk memplot fungsi akar kuadrat

$y = B\,x^{1/2}$

dengan berbagai nilai parameter B.

```
figure;
x=linspace(0,4,500);
A=0.5:0.5:2;
i=1;
while i <= length(A)
       y = A(i)* x.^(1/2);
       plot(x,y); hold on;
       i=i+1;
end
grid on;
```

Menghasilkan:



Gambar 6. 4 Contoh plot 4 kurva dengan "while"

6.5.5 Statement break dan return

Ketika kita sudah berada dalam suatu loop, kita bisa keluar dengan **break** tanpa menunggu **nilai_akhir** tercapai, atau tanpa menunggu kondisi loop tidak dipenuhi lagi. Sementara, **return** digunakan untuk keluar dari fungsi yang sedang berjalan. Berikut ini gambarannya dalam kasus penentuan apakah suatu bilangan bersifat prima atau tidak.

Algoritma yang akan digunakan ialah sebagai berikut:

- User memasukkan satu bilangan bulat positif **N** sebagai argumen input.

- Apabila **N** bukan bilangan bulat positif, maka perhitungan tidak dilanjutkan, dan digunakan **return** untuk keluar.
- **N** kita coba-coba bagi dengan 2, 3, 4, 5, … dst. dengan loop. Apabila satu waktu ditemukan **N** habis terbagi, berarti **N** bukan bilangan prima. Selanjutnya kita langsung keluar loop dengan **break** dan menampilkan hasilnya di layar.
- Apabila **N** tidak pernah habis dibagi oleh 2, 3, 4, … , **N**/2 (sampai loop selesai), maka **N** pasti bilangan prima. Selanjutnya kita tampilkan di layar dan program selesai.
- Untuk mengetahui apakah N habis terbagi atau tidak, kita bisa menggunakan fungsi **rem(N,pembagi)**.

```
% Fungsi untuk menentukan sifat prima suatu bilangan:
% apa_prima.m
%
function apa_prima(N)

% N : bil. bulat positif yang dimasukkan oleh user

if N <= 0     %Jika N bilangan negatif
    disp('Masukan harus bilangan bulat positif');
    return;   %Perhitungan tidak dilanjutkan
end

% Membulatkan N kalau-kalau N bukan bil. bulat
N = round(N);

prima = 1;    %Kita anggap N bil prima pd awal program
              %flag 'prima' kita set jadi satu.

for i = 2:floor(N/2)
       if rem(N,i) == 0
              prima=0;
                     % ternyata N tidak prima,
                     % flag 'prima' kita set jadi nol
              break; % Keluar dari loop
       end
end

% Menampilkan hasil:
if prima == 0
    disp(N), disp('bukan bilangan prima!');
else
    disp(N), disp('adalah bilangan prima!');
end
```

Simpanlah fungsi ini dengan nama **apa_prima.m** di dalam direktori **Matlab\work**.

Kita coba jalankan fungsi di atas pada *command window*:

```
>> apa_prima(37)
    37
adalah bilangan prima!

>> apa_prima(27)
    27
bukan bilangan prima!

>> apa_prima(-27)
Masukan harus bilangan bulat positif
```

Perlu diingat bahwa fungsi **apa_prima** di atas tidak memiliki argumen keluaran, karena hasil perhitungan langsung kita tampilkan di layar menggunakan **disp**, sehingga hasil tersebut tidak bisa disimpan dalam variabel.

6.5.6 Statement continue

Statement continue digunakan untuk memaksa program untuk langsung menuju iterasi berikutnya dari suatu loop, tanpa mengeksekusi *command* yang masih ada di bawahnya.

Sebagai contoh, kita akan membuat fungsi untuk mengumpulkan bilangan tak nol dari suatu vektor.

```
% Fungsi untuk mengumpulkan bilangan
% tak nol di dalam vektor
% hit_taknol.m

function y = hit_taknol(x)

% x : vektor masukan
% y : vektor berisi bilangan tak nol dari x

y = [];
for i=1:length(x)
    if x(i)==0
        continue
    else
        y=[y x(i)];
    end
end
```

Sekarang kita coba:

```
>> x = [0 0 2 -3.6 0 0 0 3 0 -0.6 10 0 0 0];
```

```
>> y = hit_taknol(x)
y =
    2.0000   -3.6000    3.0000   -0.6000   10.0000
```

6.6 Operator Perbandingan dan Logika

Seperti yang kita lihat pada subbab 6.5 Control Statement, kita harus bisa menuliskan kondisi dalam bahasa MATLAB untuk menciptakan percabangan program ataupun loop. Untuk keperluan ini kita mungkin harus membandingkan dua variabel (sama atau tidak, lebih besar atau lebih kecilkah?), mengevaluasi apakah suatu variabel memenuhi satu dari sejumlah syarat, dan sebagainya.

Untuk membandingan dua variabel digunakan operator berikut ini:

Tabel 6. 1

<	>	lebih kecil, lebih besar
<=	>=	lebih kecil atau sama dengan, lebih besar atau sama dengan
==	~=	sama dengan, tidak sama dengan

Sementara untuk mengevaluasi logika, digunakan fungsi dan operator:

Tabel 6. 2

and(A,B) atau **A & B**	operasi logika AND antara **A** dan **B**
or(A,B) atau **A | B**	operasi logika OR
xor(A,B)	operasi logika XOR
not(A) atau **~A**	operasi logika NOT pada **A**

A dan **B** di sini bisa berupa skalar, vektor, maupun matriks, asalkan ukuran **A** dan B sama.

Adapun tabel kebenaran yang digunakan pada setiap operasi logika tersebut ialah sebagai berikut:

Tabel 6. 3

A	B	A & B	A | B	xor(A,B)	~A
nol	nol	0	0	0	1
nol	bukan nol	0	1	1	1
bukan nol	nol	0	1	1	0
bukan nol	bukan nol	1	1	0	0

Perlu diperhatikan bahwa operasi logika memiliki prioritas untuk dihitung lebih dahulu, kemudian diikuti operasi aritmatika, lalu operasi perbandingan.

Untuk menambah pemahaman, mari kita praktekkan contoh di bawah ini di *command window*:

```
>> A = [1 2 0 -1 -2]; B = [1 0 0 0 5];
>> C = and(A,B)
C =
     1     0     0     0     1

>> D=A|B|C
D =
     1     1     0     1     1

>> E = xor(~A,B)
E =
     1     0     1     0     1
```

Sekarang, mari kita mencoba membuat fungsi untuk menentukan suatu tahun termasuk kabisat atau tidak. Jangkauan tahun yang bisa dihitung ialah 1900 hingga 2500. Kita ketahui bahwa tahun kabisat terjadi pada tahun-tahun berkelipatan 4, kecuali tahun akhir abad; namun untuk tahun akhir abad berkelipatan 400 termasuk kabisat pula.

```
% Fungsi untuk mengetahui tahun kabisat atau tidak
% iskabisat.m

function hasil = iskabisat(thn)

% thn : merupakan masukan bilangan bulat positif
% hasil = 1 jika kabisat, 0 jika tidak

if thn<1900 | thn>2500
    disp('Tahun yang valid: 1900 - 2500');
    hasil=[];
    return
end
```

```
if rem(thn,4)==0 & (rem(thn,100)~=0|rem(thn,400)==0)
    hasil=1;
else
    hasil=0;
end
```

Pada fungsi tersebut, terdapat dua *control statement* " if ":

- `if thn<1900 | thn>2500`
  Berarti jika variabel **thn** kurang dari 1900 ATAU lebih dari 2500, *command* di dalam " if " tersebut akan dijalankan.
- `if rem(thn,4)==0 & ...`
  `(rem(thn,100)~=0|rem(thn,400)==0)`
  Berarti jika variabel **thn** habis dibagi 4 DAN logika `(rem(thn,100)~=0|rem(thn,400)==0)` bernilai 1 (true), maka *command* setelah " if " akan dijalankan.

Perlu diperhatikan bahwa logika `(rem(thn,100)~=0|rem(thn,400)==0)` akan bernilai 1 bila **thn** bukan tahun abad (kelipatan 100); ataupun kalau tahun abad haruslah kelipatan 400.

Sekarang kita bisa coba:

```
>> iskabisat(2005), iskabisat(1972)
ans =
       0
ans =
       1
```

Fungsi ini hanya bisa mengolah masukan skalar. Lalu bagaimana kalau diinginkan masukan berupa vektor atau matriks? Kita bisa ubah fungsinya menjadi berikut ini:

```
% Fungsi untuk mengetahui tahun kabisat atau tidak
% iskabisat.m

function hasil = iskabisat(thn)

% thn : merupakan masukan bilangan bulat positif
% hasil = 1 jika kabisat, 0 jika tidak

if sum(sum(thn<1900 | thn>2500))~=0
    disp('Tahun yang valid: 1900 - 2500');
    hasil=[];
    return
end
```

```
hasil = rem(thn,4)==0 & ...
(rem(thn,100)~=0|rem(thn,400)==0);
```

Sekarang kita bisa coba untuk menentukan tahun kabisat antara 1980 hingga 1990.

```
>> iskabisat(1980:1990)
ans =
1    0    0    0    1    0    0    0    1    0    0
```

Soal Latihan

1. Buatlah program dengan M-file untuk menghitung volume dan luas permukaan balok bila diketahui:
   panjang = 5, lebar = 3, tinggi = 6,5.
   Beri nama program ini dengan **prog_balok.m**

2. Buatlah suatu fungsi dengan M-file untuk menghitung volume dan luas permukaan balok dengan spesifikasi:
   masukan fungsi : panjang, lebar, dan tinggi balok
   keluaran fungsi : volume, dan luas permukaan balok.
   Beri nama fungsi ini dengan **hitung_balok.m**

3. Buatlah suatu fungsi dengan M-file untuk menghitung volume dan luas permukaan dari suatu prisma segiempat dengan spesifikasi:
   masukan fungsi : panjang dan lebar alas prisma, serta tinggi prisma
   keluaran fungsi : volume, dan luas permukaan prisma
   Beri nama fungsi ini dengan **hitung_prisma.m**

4. Buatlah suatu program untuk menampilkan segitiga Pascal. Pengguna harus memasukkan jumlah level segitiga yang ingin ditampilkan melalui *command* **input**. Apabila pengguna menginginkan segitiga 4 level maka akan tampil keluaran:
   1
   1 1
   1 2 1
   1 3 3 1
   Beri nama program ini dengan **prog_pascal.m**

5. Buatlah sebuah fungsi untuk menghitung jumlah hari di antara dua tanggal. Spesifikasi dari fungsi tersebut ialah:
   masukan : tanggal, bulan, dan tahun awal, serta tanggal, bulan, dan tahun akhir.
   keluaran : jumlah hari di antara dua tanggal tersebut.
   Beri nama fungsi ini dengan **hitung_hari.m**.
   Misalkan kita ingin menghitung jumlah hari antara 2 Januari 2004 hingga 5 November 2006, maka ketikkan:

```
>> jml_hari = hitung_hari(2,1,2004,5,11,2006)
jml_hari =
    1038
```

BAB 7

ANALISIS DATA

Dalam bab ini, kita akan belajar bagaimana menganalisis dan memanipulasi data mempergunakan MATLAB, terutama untuk perhitungan statistik: rentang data, maksimum/minimum, rata-rata, deviasi, jumlah kumulatif, dan sebagainya. Di MATLAB fungsi-fungsi statistik semacam ini telah ada dan bisa digunakan secara fleksibel.

Dalam penjelasan bab ini, **x** dan **y** kita misalkan sebagai vektor (baris ataupun kolom), dan **A** dan **B** sebagai matriks m×n.

7.1 Maksimum dan Minimum

Nilai maksimum dan minimum diperoleh dengan **command** berikut ini:

Tabel 7. 1

max(x)	menghitung nilai maksimum dari elemen vektor **x**. Jika **x** bernilai kompleks maka dihitung **max(abs(x))**
max(A)	menghitung nilai maksimum dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
max(max(A))	menghitung nilai maksimum dari elemen matriks **A**
max(A,B)	menghitung matriks berukuran sama dengan **A** dan **B** dengan elemen berisi nilai terbesar di antara elemen **A** dan **B** pada posisi yang sama
min(...)	sama dengan sintaks **max(...)** di atas, tetapi untuk mencari minimum

Mari kita praktekkan beberapa contoh untuk menambah pemahaman terhadap sintaks di atas. Misalkan **x** ialah data tinggi badan dari 10 orang, dan **A** ialah data indeks prestasi (IP) dari 4 mahasiswa dalam 3 semester.

Data tinggi badan (dalam cm)
175 177 173 165 160 170 174 177 168 170

Data IP mahasiswa			
Nama	**IP sem-1**	**IP sem-2**	**IP sem-3**
Agus	3,3	2,8	3,3
Dedy	3,9	4,0	3,8
Tanjung	3,8	3,5	2,9
Vijay	2,9	3,2	3,1

```
>> x=[175 177 173 165 160 170 174 177 168 170];
>> A=[3.3  2.8  3.3;
3.9  4.0  3.8;
3.8  3.5  2.9;
2.9  3.2  3.1];

>> max(x)
ans =
   177

>> max(A), max(A')
ans =
    3.9000    4.0000    3.8000
ans =
    3.3000    4.0000    3.8000    3.2000

>> max(max(A))
ans =
     4
```

Kita bisa melihat bahwa **max(x)** menghitung tinggi maksimum dari 10 orang yang ada, **max(A)** menghitung IP tertinggi pada setiap semester, sedangkan **max(A')** menghitung IP tertinggi dari setiap mahasiswa. Sementara itu, **max(max(A))** menghitung IP tertinggi yang pernah dicapai mahasiswa selama 3 semester.

7.2 Jumlah dan Produk

Beberapa jenis operasi penjumlahan bisa dilakukan dengan *command* **sum** dan **cumsum**.

Tabel 7. 2

sum(x)	menjumlahkan nilai elemen vektor **x**
sum(A)	menjumlahkan nilai elemen dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
sum(sum(A))	menjumlahkan nilai semua elemen matriks **A**
cumsum(x)	menghitung vektor berukuran sama dengan **x** berisi jumlah kumulatif elemen **x**; yaitu elemen kedua ialah jumlah dari elemen pertama dan kedua dari **x**, dan seterusnya
cumsum(A)	menghitung matriks berukuran sama dengan **A** di mana kolom-kolomnya merupakan jumlah kumulatif dari kolom di **A**

Sebagai contoh, kita akan definisikan vektor **y** dan matriks **B** sebagai berikut:

$$y = \begin{pmatrix} 1 & 4 & 9 & 16 & 25 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
>> y=[1:5].^2;
>> B=[1:3 ; 4:6 ; 7:9];
>> jml_y = sum(y)
jml_y =
       55
>> jml_B = sum(B)
jml_B =
       12   15   18
>> total_B = sum(sum(B))
total_B =
       45

>> kumulasi_y = cumsum(y)
kumulasi_y =
       1    5    14   30   55
>> kumulasi_B = cumsum(B)
kumulasi_B =
     1    2    3
     5    7    9
    12   15   18
```

Sementara itu, produk (perkalian elemen-elemen) vektor dan matriks bisa diperoleh dengan cara yang mirip.

Tabel 7. 3

prod(x)	mengalikan nilai elemen vektor **x**
prod(A)	mengalikan nilai elemen dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
prod(prod(A))	mengalikan nilai semua elemen matriks **A**
cumprod(x)	menghitung vektor berukuran sama dengan **x** berisi produk kumulatif elemen **x**; yaitu elemen kedua ialah perkalian dari elemen pertama dan kedua dari **x**, dan seterusnya
cumprod(A)	menghitung matriks berukuran sama dengan **A** di mana kolom-kolomnya merupakan produk kumulatif dari kolom di **A**

Sebagai contoh kita gunakan vektor **y** dan matriks **B** seperti sebelumnya.

```
>> pdk_y = prod(y)
pdk_y =
       14400
>> pdk_B = prod(B)
pdk_B =
    28    80   162
>> tot_pdk_B = prod(prod(B))
tot_pdk_B =
      362880
>> kumulasi_pdk_y = cumprod(y)
kumulasi_pdk_y =
      1    4    36   576  14400
>> kumulasi_pdk_B = cumprod(B)
kumulasi_pdk_B =
     1     2     3
     4    10    18
    28    80   162
```

7.3 Statistika

Pada subbab sebelumnya, telah disajikan *command* operasi vektor dan matriks untuk menghitung maksimum, minimum, jumlah,

serta produk. Sekarang kita akan belajar *command* untuk analisis data statistik.

Tabel 7. 4

mean(x)	menghitung rata-rata aritmatik dari elemen vektor **x**
mean(A)	menghitung rata-rata aritmatik dari elemen setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
median(...)	sama seperti sintaks **mean(...)**, tetapi untuk menghitung median (nilai tengah)
std(...)	sama seperti sintaks **mean(...)**, tetapi untuk menghitung deviasi standar (simpangan baku)
var(...)	sama seperti sintaks **mean(...)**, tetapi untuk menghitung variansi

Sebagai contoh, kita gunakan kembali data tinggi badan dan nilai IP mahasiswa seperti sebelumnya.

```
>> x=[175 177 173 165 160 170 174 177 168 170];
>> A=[3.3  2.8  3.3;
3.9  4.0  3.8;
3.8  3.5  2.9;
2.9  3.2  3.1];

>> rataan_IP_sem = mean(A)
rataan_IP_sem =
    3.4750    3.3750    3.2750
>> rataan_IP_mhs = mean(A')
rataan_IP_mhs =
    3.1333    3.9000    3.4000    3.0667
>> rataan_IP_total = mean(mean(A))
rataan_IP_total =
    3.3750

>> nilai_tengah = median(x), deviasi = std(x), ...
variansi = var(x)
nilai_tengah =
  171.5000
deviasi =
    5.4661
variansi =
   29.8778
```

7.4 Sortir

Kita bisa mengurutkan data (sortir) di MATLAB dengan *command* berikut ini:

Tabel 7. 5

sort(x)	menghitung vektor dengan elemen **x** telah tersortir secara *ascending* (dari kecil ke besar). Jika **x** bernilai kompleks maka dihitung **sort(abs(x))**
[y,ind] = sort(x)	menghitung vektor **y** berisi sortiran elemen **x**, dan vektor **ind** berisi indeks sehingga **y = x(ind)**
[B,Ind] = sort(A)	menghitung matriks **B** berisi sortiran kolom-kolom matriks **A**. Setiap kolom pada matriks **Ind** berisi indeks seperti halnya kasus vektor di atas

Mari kita coba *command* tersebut pada data tinggi badan dan IP mahasiswa.

Kita urutkan data tinggi badan dari kecil ke besar (ascending).

```
>> sort(x)
ans =
160  165  168  170  170  173  174  175  177  177
```

Atau kita urutkan disertai indeks yang menunjukkan nomor urut elemen pada vektor **x** sebelum disortir.

```
>> [y,ind]=sort(x)
y =
160  165  168  170  170  173  174  175  177  177
ind =
 5    4    9    6   10    3    7    1    2    8
```

Untuk mengurutkan tinggi badan dari besar ke kecil (descending).

```
>> fliplr(sort(x))
ans =
177  177  175  174  173  170  170  168  165  160
```

Demikian pula untuk mengurutkan elemen matriks: secara

ascending pada kolom per kolom:

```
>> sort(A)
ans =
    2.9000    2.8000    2.9000
    3.3000    3.2000    3.1000
    3.8000    3.5000    3.3000
    3.9000    4.0000    3.8000
```

Atau secara *descending* pada kolom per kolom:

```
>> flipud(sort(A))
ans =
    3.9000    4.0000    3.8000
    3.8000    3.5000    3.3000
    3.3000    3.2000    3.1000
    2.9000    2.8000    2.9000
```

Ataupun melakukan sortir dengan indeks. Perhatikan bahwa kolom-kolom dalam **IND** berisi nomor urut elemen pada matriks **A** sebelum disortir.

```
>> [Y,IND]=sort(A)
Y =
    2.9000    2.8000    2.9000
    3.3000    3.2000    3.1000
    3.8000    3.5000    3.3000
    3.9000    4.0000    3.8000
IND =
     4     1     3
     1     4     4
     3     3     1
     2     2     2
```

Command **fliplr** dan **flipud** telah dibahas pada subbab 3.6 (Tabel 3.3).

7.5 Histogram

Histogram dan diagram batang yang kerap digunakan untuk menggambarkan data statistik juga bisa ditampilkan dengan MATLAB dengan *command* berikut ini:

Tabel 7. 6

hist(x)	memplot histogram dari data di **x** dalam 10 interval
hist(x,n)	memplot histogram dari data di **x** dalam *n* interval
hist(x,y)	memplot histogram dari data di **x** dengan interval yang dinyatakan oleh **y**. Elemen vektor **y** harus terurut secara ascending.
bar(x)	memplot diagram batang dari data di **x**
bar(z,x)	memplot diagram batang dari data di **x** pada posisi yang didefinisikan oleh **z**
bar(z,x,'string')	memplot diagram batang dengan property ditentukan oleh **'string'**, seperti pada Tabel 5.3.
stairs(x)	memplot diagram tangga
stairs(z,x)	memplot diagram tangga dari data di **x** pada posisi yang didefinisikan oleh **z**
stem(y)	memplot data diskrit dari data di **y**
stem(x,y)	memplot data diskrit dari data di **y** pada posisi yang didefinisikan oleh **x**

Pada *command* **hist**, **bar**, dan **stairs**, data bisa juga disimpan untuk penggunaan selanjutnya.

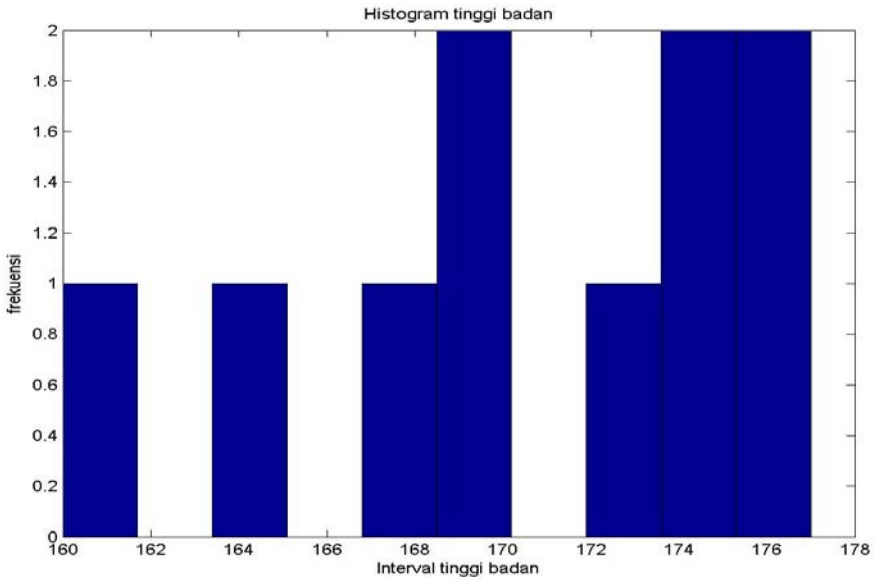
Tabel 7. 7 (lanjutan)

[m,y] = hist(x) membuat histogram dengan 10 interval seragam antara minimum **x** dan maximum **x**. Vektor **y** berisi 10 nilai antara **min(x)** dan **max(x)** yang terpisah seragam; vektor **m** berisi jumlah pada setiap interval. Histogram bisa diplot dengan **bar(y,m,'string')**

[m,y] = hist(x,n)	membuat histogram dengan *n* interval seragam
[m,y] = hist(x,y)	membuat histogram dengan interval didefinisikan oleh vektor y
[xb,yb] = bar(y)	membuat diagram batang dari nilai di **y**. Diagram bisa diplot dengan **plot(xb,yb)**
[xb,yb] = bar(x,y)	membuat diagram batang dari nilai **y** dengan posisi yang didefinisikan oleh **x**
[xb,yb] = stairs(y)	membuat diagram tangga dari nilai di **y**
[xb,yb] = stairs(x,y)	membuat diagram tangga dari nilai **y** dengan posisi yang didefinisikan oleh **x**

Mari kita coba gunakan data tinggi badan yang ada. Pertama, kita plot menjadi histogram dengan 10 interval.

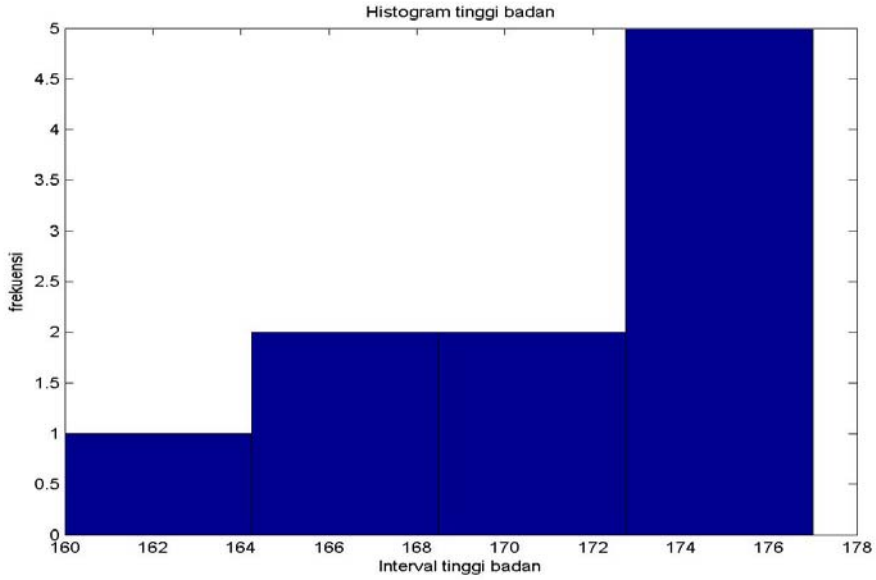
```
>> x=[175 177 173 165 160 170 174 177 168 170];
>> hist(x); title('Histogram tinggi badan');
>> xlabel('Interval tinggi badan');
>> ylabel('frekuensi');
```



Gambar 7. 1 Membuat histogram dengan "hist"

Jika kita hanya menginginkan 4 interval, maka:

```
>> hist(x,4); ); title('Histogram tinggi badan');
>> xlabel('Interval tinggi badan');
>> ylabel('frekuensi');
```



Gambar 7. 2 Membuat histogram dengan 4 interval

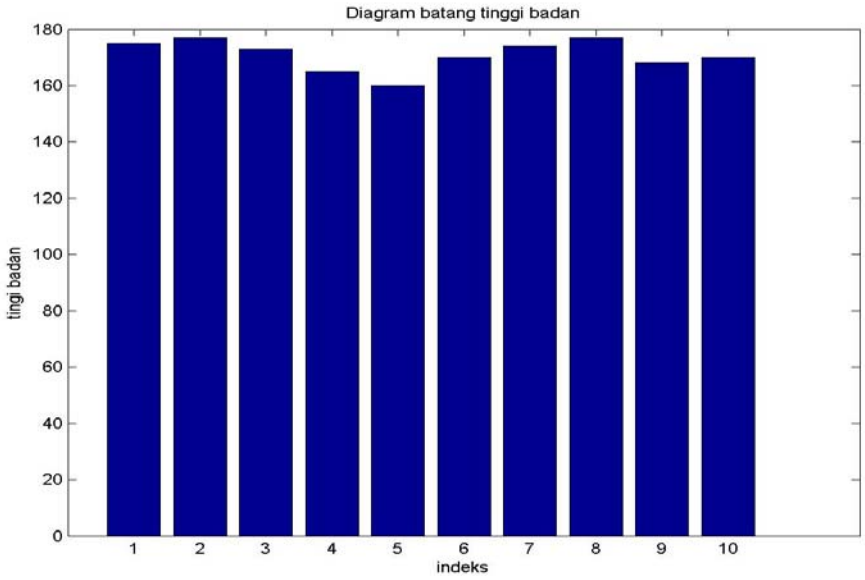
Perhatikan bahwa histogram di atas menggambarkan distribusi dari tinggi badan, dikelompokkan dalam sejumlah interval yang lebarnya seragam.

Sementara itu, untuk menggambar data tinggi badan itu sendiri dengan diagram batang, caranya mudah:

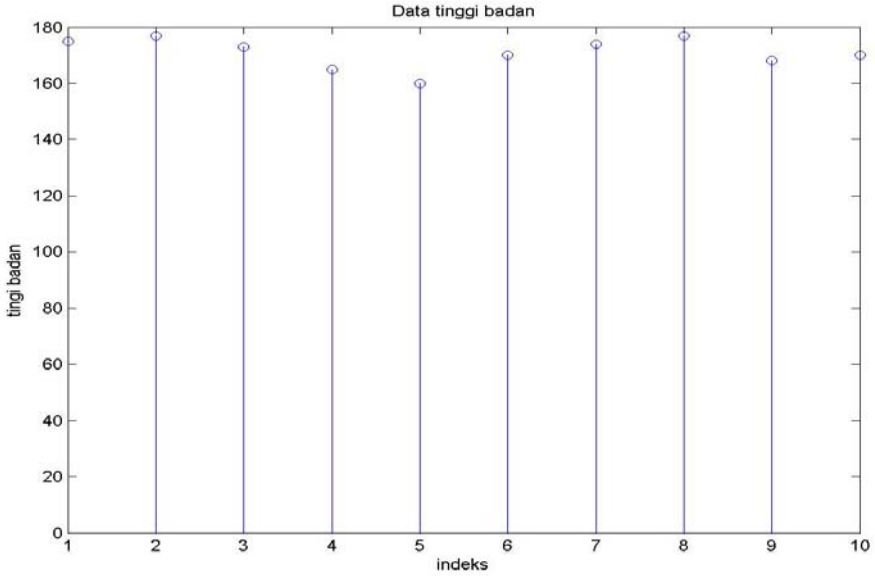
```
>> bar(x); title('Diagram batang tinggi badan');
```

Atau kita bisa juga memplot vektor **x** tersebut sebagai data diskrit.

```
>> stem(x)
```



Gambar 7. 3 Membuat diagram batang dengan "bar"



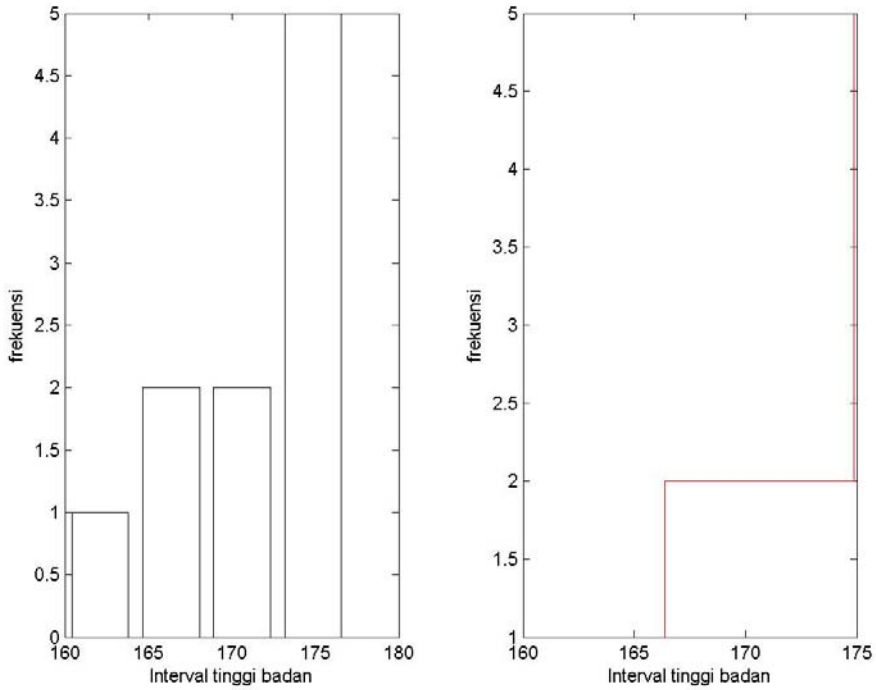
Gambar 7. 4 Memplot data diskrit dengan "stem"

Sekarang kita coba membuat histogram dan disimpan dalam variabel dengan *command* yang ada, kemudian kita plot diagram batangnya dan beri warna putih.

```
>> [m,y]=hist(x);
>> subplot(1,2,1); bar(y,m,'w')
>> xlabel('Interval tinggi badan')
>> ylabel('frekuensi')
```

Data tadi juga bisa kita plot sebagai diagram tangga berwarna merah:

```
>> subplot(1,2,2); stairs(y,m,'r')
>> xlabel('Interval tinggi badan')
>> ylabel('frekuensi')
```



Gambar 7. 5

7.6 Analisis Frekuensi : Transformasi Fourier

Analisis frekuensi terhadap suatu data ataupun sinyal umumnya dilakukan dengan transformasi Fourier. Dengan transformasi ini, kita bisa mengamati dan mengukur komponen frekuensi berapa saja yang menyusun data / sinyal tersebut. Untuk melakukan analisis frekuensi di dalam MATLAB, telah tersedia *command* "Fast Fourier Transform" (FFT) sebagai berikut:

Tabel 7. 8

fft(x)	menghitung "Transformasi Fourier Diskrit" dengan metode FFT dari vektor **x**. Apabila x berupa matriks, operasi akan dilakukan per kolom
fft(x,n)	menghitung FFT *n*-titik. Jika panjang **x** lebih dari *n* maka sisanya akan diisi nol; jika panjang **x** lebih dari *n* maka akan dipotong
ifft(X)	menghitung invers-FFT dari **X**
ifft(X,n)	menghitung invers-FFT *n*-titik

X = fft(x) dan **x = ifft(X)** dihitung dengan formula "Transformasi Fourier Diskrit" untuk *N*-titik sebagai berikut:

$$X(k) = \sum_{n=1}^{N} x(n) e^{-j2\pi(k-1)\frac{n-1}{N}} \qquad \text{untuk} \quad 1 \le k \le N$$

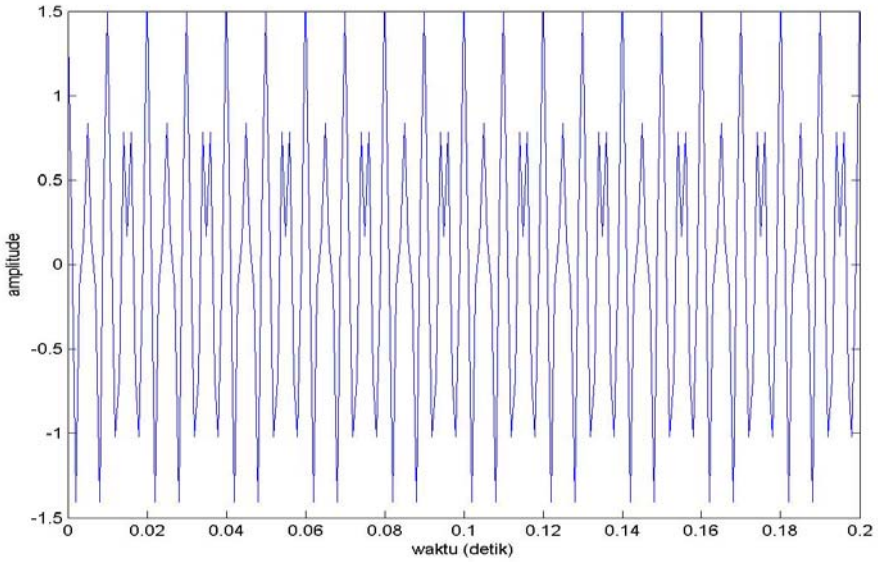
$$x(n) = \sum_{n=1}^{N} X(k) e^{j2\pi(k-1)\frac{n-1}{N}} \qquad \text{untuk} \quad 1 \le n \le N$$

Sebagai contoh, kita memiliki suatu sinyal seperti berikut ini:

```
>> clear;
>> Fs = 1000; % frekuensi sampling 1000Hz
>> t = 0:1/Fs:1.5; % durasi sinyal 1,5 detik
>> tone1 = 200;
>> tone2 = 300;
>> tone3 = 450; % 3 frekuensi tone dalam Hz
>> sinyal = cos(2*pi*tone1.*t) + ...
1/2*cos(2*pi*tone2.*t) + 1/3*sin(2*pi*tone3.*t);
```

Kita bisa lihat bentuk "time-domain" dari sinyal tersebut, kemudian kita dengarkan:

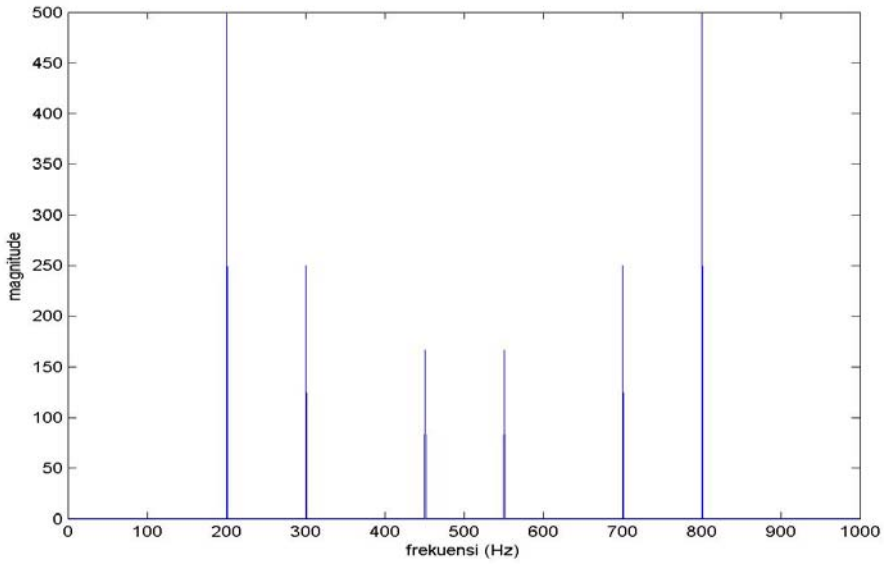
```
>> plot(t,sinyal); axis([0 0.2 -1.5 1.5]);
>> xlabel('waktu (detik)'); ylabel('amplitude')
>> sound(sinyal,Fs);
```



Gambar 7. 6 Bentuk "time-domain" dari sinyal

Kemudian kita lihat bentuk "frequency-domain" dari sinyal untuk mengetahui kandungan frekuensinya:

```
>> S = fft(sinyal,Fs);
>> plot(abs(S));
>> xlabel('frekuensi (Hz)'); ylabel('magnitude')
```



Gambar 7. 7 Bentuk "frequency-domain" dihitung dengan "fft"

Pada contoh di atas, vektor **S**, hasil operasi FFT, berisi bilangan kompleks, sehingga yang diplot adalah "magnitude" dari vektor **S**

dengan *command* **plot(abs(S))**.

Command yang berkaitan dengan bilangan kompleks telah dibahas pada subbab 2.4, Tabel 2.3.

Perhatikan bahwa hasil plot terlihat simetris kiri-kanan, hal ini merupakan ciri khas dari transformasi Fourier. Dalam hal ini yang perlu kita perhatikan ialah plot pada frekuensi 0 s.d. Fs/2 saja, yaitu 0-500 Hz. Pada rentang ini terlihat 3 komponen frekuensi yang tajam, yaitu: 200, 300, dan 450 Hz dengan magnitude masing-masing 500, 250, dan 167. Magnitude ini proporsional dengan amplituda dari tiga tone komponen **sinyal** yaitu: 1, 1/2 , dan 1/3.

Soal Latihan

1. Berikut ini data pendudukan kanal pada suatu "trunk" (saluran transmisi antar-sentral) pada setiap jam selama dua belas jam:

Data pendudukan kanal trunk			
Waktu	**Pendudukan**	**Waktu**	**Pendudukan**
6:00-7:00	100	12:00-13:00	958
7:00-8:00	350	13:00-14:00	1008
8:00-9:00	824	14:00-15:00	897
9:00-10:00	1056	15:00-16:00	921
10:00-11:00	1525	16:00-17:00	958
11:00-12:00	1247	17:00-18:00	215

   Hitunglah dan gambarlah:
   a) Maksimum pendudukan per jam dari trunk tersebut
   b) Total pendudukan trunk selama 12 jam
   c) Mean dan median dari pendudukan per jam selama 12 jam
   d) Simpangan baku dan variansi dari pendudukan per jam selama 12 jam
   e) Urutkan data pendudukan trunk tersebut secara ascending dan descending.
   f) Tampilkan data pendudukan trunk tersebut dengan diagram diskrit (*command* **stem**).
   g) Tampilkan distribusi nilai pendudukan dengan histogram warna merah dalam 6 interval.

2. Berikut ini data pengukuran temperatur suatu ruang penyimpanan yang dilakukan tiga kali selama tiga hari berturut-turut Dalam setiap seri dilakukan pengukuran per jam selama 8 jam:

Data pengukuran temperatur (dalam ^{o}C)			
Waktu	**Hari ke-1**	**Hari ke-2**	**Hari ke-3**
9:00	27,0	26,8	27,1
10:00	28,2	28,1	28,8
11:00	29,5	30,3	29,0
12:00	29,6	30,6	29,1
13:00	30,0	30,0	31,2
14:00	30,5	31,0	31,3
15:00	29,8	29,6	30,2
16:00	28,9	27,5	26,8

Hitunglah dan gambarlah:

a) Rata-rata temperatur pada masing-masing hari
b) Rata-rata temperatur pada jam 11:00, 12:00, dan 13:00.
c) Rata-rata temperatur pada hari pertama dan kedua
d) Rata-rata temperatur selama tiga hari pada pukul 9:00-12:00.
e) Temperatur tertinggi dan terendah selama tiga hari
f) Rata-rata, median, simpangan baku, dan variansi dari temperatur selama tiga hari tersebut.
g) Histogram dari temperatur selama tiga hari, digambarkan dalam 4 interval.

3. Keluarkan file suara WAV yang sudah dibuat pada Soal Latihan bab 5 berisi urut-urutan tone DO-RE-MI-FA-SOL-LA-TI-DO. Lakukanlah analisis frekuensi dengan Transformasi Fourier dengan *command* **fft** *n*-titik, di mana *n* ialah panjang sinyal WAV tadi. Plot magnitude dari hasil transformasi tersebut pada rentang frekuensi 0 hingga 1 kHz.

BAB 8

ANALISIS FUNGSI DAN INTERPOLASI

Berbagai fungsi matematis bisa dievaluasi dan dianalisis dengan berbagai *command* yang ada di MATLAB. Salah satu fungsi matematis yang sering digunakan, yaitu polinomial, penanganan dan evaluasinya akan dibahas pula dalam bagian ini. Berikutnya akan disajikan juga analisis fungsi, misalkan mencari nol, maksimum, dan minimum. Di samping itu, interpolasi dan *curve-fitting* menggunakan MATLAB akan dibahas pula. Pada bagian akhir akan dikenalkan "Function Tool, yaitu sebuah tool analisis fungsi yang ada di MATLAB.

8.1 Polinomial di MATLAB

Suatu polinomial, *p(x)*, berderajat *n* dinyatakan sebagai sebuah vektor baris **p** berukuran *n*+1. Elemen vektor menunjukkan koefisien dari polinomial yang diurutkan dari orde tertinggi ke terendah.

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

dinyatakan sebagai:

p = (a_n a_{n-1} ... a_1 a_0)

Command berikut digunakan untuk menangani polinomial:

Tabel 8. 1

polyval(p,x)	mengevaluasi polinonial **p** pada nilai **x**. **x** bisa berupa skalar maupun vektor
poly(x)	menghitung vektor sepanjang *n*+1 yang mewakili suatu polinomial orde-*n*. Vektor **x** sepanjang *n* berisi akar-akar dari polinom tersebut
roots(p)	menghitung vektor berisi akar-akar dari polinomial **p**
conv(p,q)	menghitung produk (hasil perkalian) dari polinomial **p** dan **q**. Bisa juga dianggap sebagai konvolusi antara **p** dan **q**
[k,r] = deconv(p,q)	membagi polinomial **p** dengan **q**. Hasil pembagian disimpan dalam polinom **k** dan sisa pembagian dalam polinom **r**. Bisa juga dianggap sebagai dekonvolusi antara **p** dan **q**
polyder(p)	menghitung vektor sepanjang *n* berisi turunan pertama dari polinom **p**

Misalkan kita memiliki dua polinomial sebagai berikut:

$$g(x) = 2x^3 + 5x - 1 \qquad h(x) = 6x^2 - 7$$

Dalam MATLAB kedua polinomial ini dinyatakan dengan:

```
>> g = [2 0 5 -1];
>> h = [6 0 -7];
```

Untuk mengevaluasi polinomial pada x = 10 kita tuliskan:

```
>> nilai1 = polyval(g,10), nilai2 = polyval(h,10)
nilai1 =
        2049
nilai2 =
   593
```

Namun bisa pula **x** berbentuk vektor:

```
>> x = -3:3
x =
    -3    -2    -1     0     1     2     3

>> nilai1 = polyval(g,x), nilai2 = polyval(h,x)
nilai1 =
   -70   -27    -8    -1     6    25    68
```

```
nilai2 =
    47    17    -1    -7    -1    17    47
```

Jika kita kalikan kedua polinomial tersebut, akan diperoleh sebuah polinomial baru:

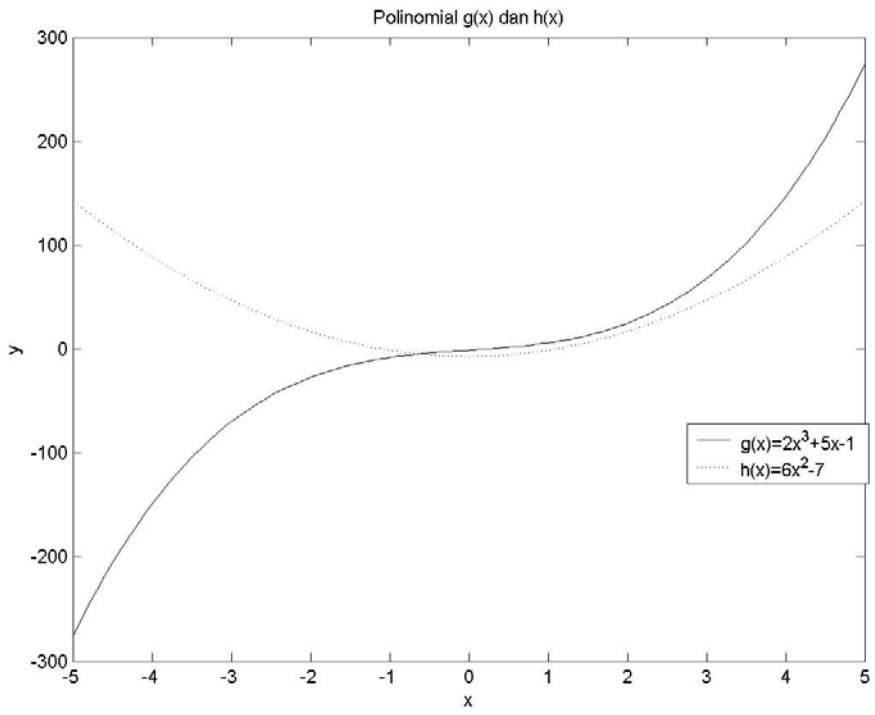
```
>> p = conv(g,h)
p =
    12     0    16    -6   -35     7
```

yang mewakili: $p(x) = 12x^5 + 16x^3 - 6x^2 - 35x + 7$

Akar-akar dari polinomial *g(x)* dan *h(x)* bisa kita hitung:

```
>> akar_g = roots(g), akar_h = roots(h)
akar_g =
  -0.0985 + 1.5903i
  -0.0985 - 1.5903i
   0.1969
akar_h =
    1.0801
   -1.0801
```

Perhatikan plot dari kedua polinomial tersebut.



Gambar 8. 1 Plot polinomial g(x) dan h(x)

Turunan pertama dan kedua dari g(x) bisa kita hitung pula:

```
>> g1=polyder(g),  g2=polyder(g1)
g1 =
     6     0     5
g2 =
    12     0
```

yang masing-masing mewakili

$g'(x) = 6x^2 + 5$ dan $g''(x) = 12x$

8.2 Nol dari Fungsi

Fungsi matematis bisa dinyatakan dalam bentuk M-file di MATLAB. Misalkan fungsi

$$f(x) = \frac{5x - 6.4}{(x-1.3)^2 + 0.002} + \frac{9x}{x^3 + 0.03} - \frac{x - 0.4}{(x-0.92)^2 + 0.005}$$

bisa kita tuliskan pada editor M-file (lihat kembali subbab 6.1)

```
function y = f(x)

y = (5.*x - 6.4)./((x-1.3).^2 + 0.002) + ...
(9.*x)./(x.^3 + 0.003) - ...
(x - 0.4)./((x-0.92).^2 + 0.005);
```

Fungsi **f** didefinisikan menggunakan operator elemen-per-elemen **.*** **./** **.^** **+** **–** (lihat kembali subbab 4.5 dan 4.6), sehingga apabila fungsi dipanggil dengan argumen vektor maka hasilnya juga berupa vektor. Semua fungsi MATLAB pada bab ini harus didefinisikan seperti contoh tersebut.

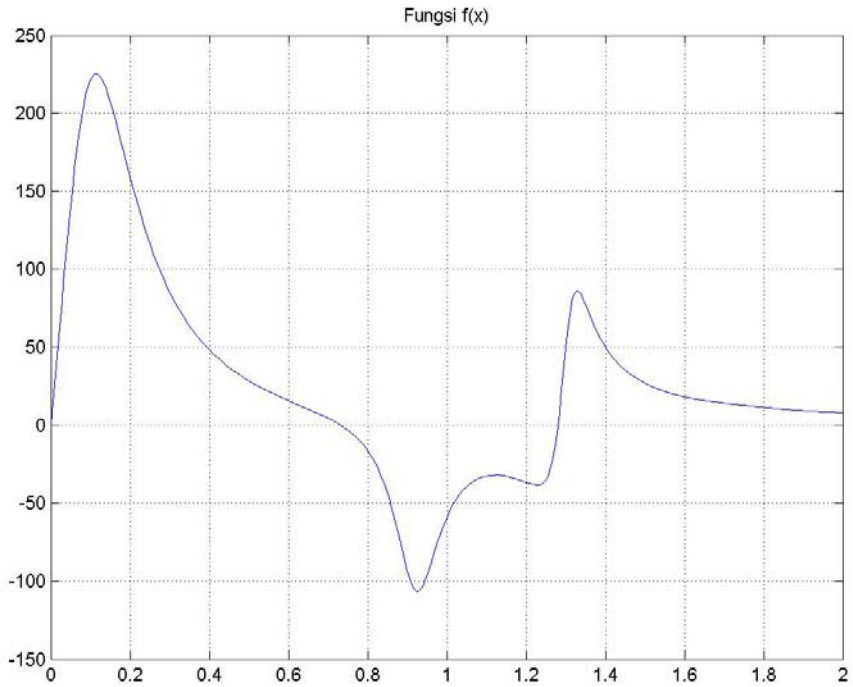
Fungsi tersebut bisa diplot dengan *command* **plot**:

```
>> x = linspace(0,2);       % membuat vektor x
>> plot(x,f(x));            % memplot grafik f(x)
>> grid on;
>> title('Fungsi f(x)');
```

Atau menggunakan *command* **fplot**:

```
>> fplot('f',[0 2]);        % memplot grafik f(x)
```

```
>> grid on;
>> title('Fungsi f(x)');
```



Gambar 8. 2 Plot fungsi rasional f(x)

Untuk mencari nol dari fungsi *f(x)*, sama saja dengan mencari solusi dari *f(x)* = 0. Nol dari suatu fungsi satu variabel bisa dicari dengan *command* **fzero**. Sementara untuk polinomial gunakanlah **roots** seperti pada subbab 8.1. Algoritma yang digunakan pada **fzero** bersifat iteratif, dan membutuhkan tebakan awal (*initial guess*) yang tidak terlalu jauh dari nol fungsi yang dicari.

Tabel 8. 2

fplot('fcn',lim,'string') memplot fungsi **fcn** pada interval **lim** dengan *property* yang didefinisikan oleh **string** (lihat Tabel 5.3). **fcn** berupa M-file yang berisi definisi fungsi. **lim** berupa vektor 2 elemen berisi batas interval x_{min} dan x_{max}.
fzero('fcn',x0) menghitung nol dari fungsi **fcn** dengan nilai tebakan awal **x0**.
fzero('fcn',x0,tol) menghitung nol dari fungsi **fcn** dengan nilai tebakan awal **x0**. **tol** menentukan toleransi error dari perhitungan pendekatan yang diinginkan

Command **zerodemo** akan memberikan demonstrasi dari topik ini.

Kita akan menghitung nol dari fungsi *f(x)* sebagai berikut:

```
>> x1=fzero('f',0), x2=fzero('f',0.5), x3=fzero('f',2)
x1 =
    0.0011
x2 =
    0.7320
x3 =
    1.2805
```

Misalkan kita ingin menghitung titik potong dari dua fungsi:
cos 2x dan *5x – 2*; atau dengan kata lain mencari solusi dari persamaan:

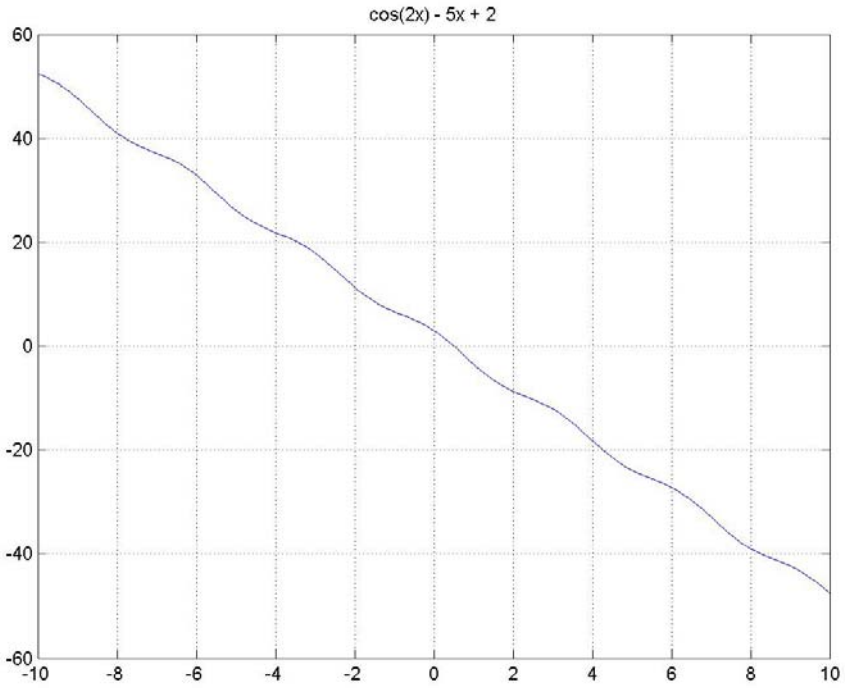
$$s(x) = \cos 2x - 5x + 2 = 0$$

Maka, pertama kita definisikan fungsi **cosm.m** dalam M-file.

```
function s = cosm(x)

s = cos(2.*x) - 5.*x + 2;
```

Kemudian kita plot untuk memudahkan mendapatkan tebakan awal:

```
>> fplot('cosm',[-10 10]);
>> grid on;
>> title('cos(2x) - 5x + 2');
```



Gambar 8. 3 Plot fungsi s(x) = cos 2x – 5x + 2

Kita lihat bahwa $x = 2$ merupakan tebakan awal yang bagus.

```
>> nol = fzero('cosm',2)
nol =
    0.5060
```

8.3 Minimum dan Maksimum dari Fungsi

Untuk melakukan optimisasi, yaitu mendapatkan solusi optimal, kita harus mendapatkan maksimum atau minimum dari fungsi pada suatu interval. Dalam hal ini MATLAB menggunakan metode numerik untuk menemukan minimum dari suatu fungsi. Algoritma yang digunakannya iteratif, yaitu suatu proses berulang.

Misalkan kita ingin mencari minimum $x_{\min}$ dari fungsi *f(x)*.

$$f(x_{\min}) = \min_x f(x)$$

Metode iteratif ini membutuhkan tebakan awal x_0. Dari nilai awal ini akan diperoleh nilai berikutnya, x_1, yang diharapkan semakin mendekati $x_{\min}$. Seberapa dekat x_1 ke $x_{\min}$ tergantung pada metode numerik yang digunakan. Proses iterasi ini berlanjut hingga nilai

x_i yang mendekati dengan akurasi tertentu diperoleh, di mana $|x_{min} - x_i|$ cukup kecil.

Dalam MATLAB tidak ada *command* untuk menentukan maksimum suatu fungsi *f(x)*, namun dalam hal ini bisa digunakan fungsi $g(x) = -f(x)$ untuk dicari minimumnya.

Tabel 8. 3

fmin('fcn',x1,x2) menghitung minimum dari fungsi satu variabel **fcn** pada interval **x1** < *x* < **x2**. Jika minimum-lokal tidak ditemukan, hasilnya ialah nilai *x* terkecil pada interval tadi.
fminbnd('fcn',x1,x2) sama dengan *command* **fmin**, tetapi untuk MATLAB versi terbaru.
fmins('fcn',x0) menghitung minimum dari fungsi multi variabel **fcn** dengan tebakan awal berupa vektor **x0**.
fminsearch('fcn',x0) sama dengan *command* **fmins**, tetapi untuk MATLAB versi terbaru.

Misalkan kita akan mencari minimum dari fungsi sinus pada interval $0 \le x \le 2\pi$.

```
>> minimum_sinus = fmin('sin',0,2*pi)
minimum_sinus =
    4.7124
```

Untuk fungsi yang lebih rumit, misalkan fungsi *f(x)* pada subbab 8.2, kita bisa temukan minimumnya pada interval $0 \le x \le 2$.

```
>> minimum_f1 = fmin('f',0,2)
minimum_f1 =
    1.2278
```

Perhatikan bahwa ini hanyalah satu "minimum-lokal" dan belum tentu merupakan minimum-global dari interval tadi. Jika kita lihat Gambar 8.2 maka terlihat bahwa minimum global terletak di interval yang lebih sempit $0 \le x \le 1$:

```
>> minimum_f2 = fmin('f',0,1)
minimum_f2 =
    0.9261
```

Untuk mencari maksimum dari fungsi *f(x)*, terlebih dahulu kita definisikan fungsi *–f(x)* dengan M-file, lalu simpanlah sebagai **minusf.m**.

```
function y = minusf(x)

y = -f(x);
```

Kemudian kita cari minimum dari fungsi tersebut yang merupakan maksimum dari *f(x)* :

```
>> maximum_f = fmin('minusf',0,2)
maximum_f =
    0.1144
```

Perhatikan kembali bahwa ini hanyalah satu "maksimum-lokal" yang ternyata kebetulan merupakan maksimum-global dari interval tadi.

Minimum dari Fungsi Multi Variabel

Misalkan kita definisikan suatu fungsi dua variabel:

$$g(x_1, x_2) = x_1^2 + x_2^2 - \frac{x_1 x_2}{4} - \sin x_1$$

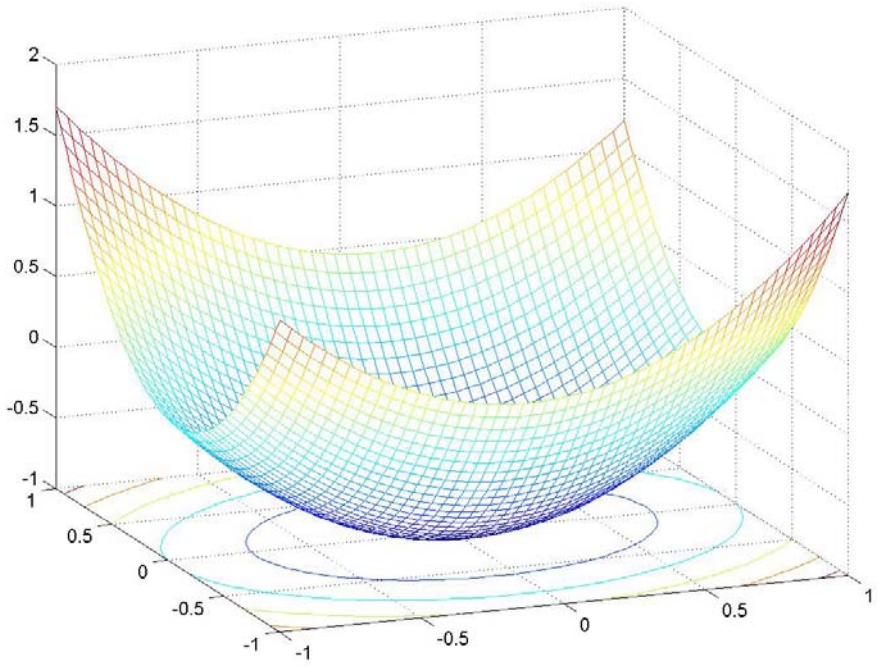
Kita tuliskan dalam M-file **gx1x2.m**

```
function g = gx1x2(x)

g = x(1).^2 + x(2).^2 - 0.25.*x(1).*x(2) - sin(x(1));
```

Kemudian kita coba plot fungsi ini beserta konturnya (penjelasan plot kontur lihat kembali subbab 5.3):

```
>> x=linspace(-1,1,50); % menciptakan vektor x
>> % asumsikan y = x
>> for i = 1:50  % menghitung gx1x2 pada setiap titik
       for j = 1:50
          Z(i,j) = gx1x2([x(i) x(j)]);
       end
   end
```

```
>> meshc(x,x,Z); % plot grafik 3-D plus kontur
```



Gambar 8. 4 Plot permukaan dan kontur dari fungsi dua variabel

Dari gambar tersebut, kita coba tebakan awal pada titik (1,0):

```
>> minimum_gx1x2 = fmins('gx1x2',[1,0])
minimum_gx1x2 =
  1.0e-004 *
    0.1467   -0.4034
```

8.4 Interpolasi

Pada fungsi yang memiliki sejumlah titik terbatas, dimungkinkan untuk menentukan titik-titik perantaranya dengan interpolasi. Cara termudah untuk menghitungnya ialah dengan menggunakan interpolasi linier untuk menghubungkan dua titik yang berdekatan.

Command **interp1** menggunakan algoritma khusus untuk interpolasi titik-titik data yang terpisah secara seragam. Untuk *command* ini, kita harus tambahkan tanda asteris '*****' di depan nama metoda yang diinginkan, misalkan **interp(x,y,xx,'*nearest')**.

Tabel 8. 4

yy = interp1(x,y,xx)

menghitung vektor **yy** yang panjangnya sama dengan vektor **xx**. Dalam hal ini **yy** fungsi dari **xx** merupakan interpolasi dari **y** fungsi dari **x**. Vektor **x** harus diurutkan secara ascending / descending

interp1(x,y,xx,'string')

menghitung interpolasi 1-dimensi; **string** menunjukkan metode yang digunakan, yaitu:

linear	interpolasi linier
nearest	interpolasi "nearest-neighbor"
spline	interpolasi "cubic-spline"
cubic	interpolasi kubik, membutuhkan jarak pisah seragam pada **x**

Apabila **string** tidak dituliskan, maka digunakan interpolasi linier. Untuk semua metode tersebut, **x** harus diurutkan ascending / descending.

interp1q(x,y,xx)

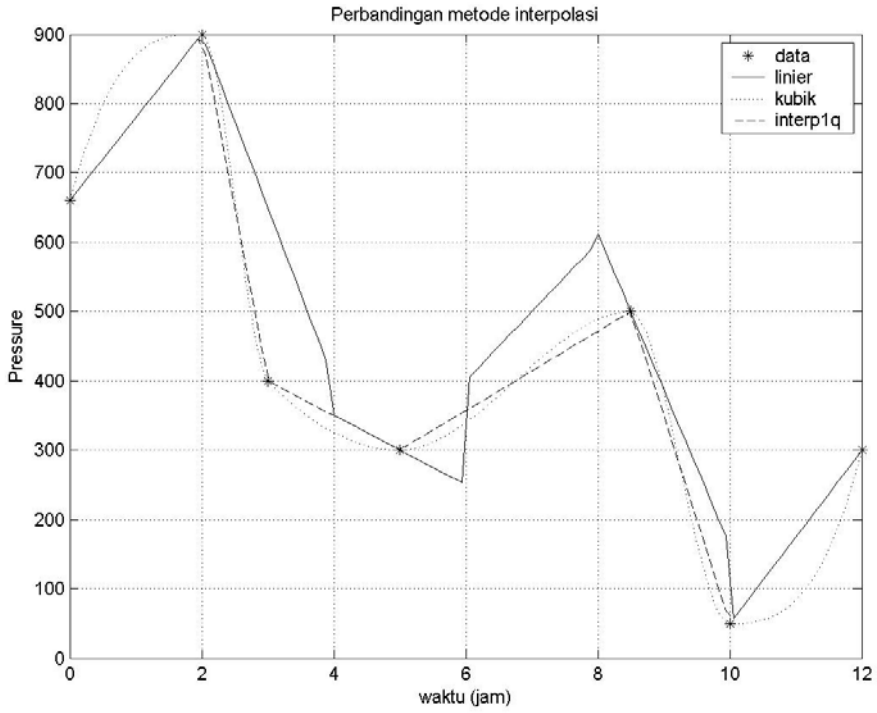
bekerja seperti **interp1** namun lebih cepat untuk titik-titik data yang terpisah tak seragam. **x**, **y**, dan **xx** harus berupa vektor kolom.

Misalkan kita memiliki data tekanan udara dalam suatu ruang tertutup yang diukur pada jam-jam tertentu sebagai berikut:

```
>> t = [0  2  3  5  8.5  10  12];
>> pres = [660 900 400 300 500 50 300];
```

Sekarang kita interpolasi dengan beberapa metode dan kita plot pada satu gambar sekaligus:

```
>> tt = linspace(0,12,100);
>> PP1 = interp1(t,pres,tt,'*linear');
>> PP2 = interp1(t,pres,tt,'*cubic');
>> PP3 = interp1q(t',pres',tt');

>> figure;
>> plot(t,pres,'k*',tt,PP1,'k-',tt,PP2,'k:', ...
tt,PP3,'k--')
>> grid on;
>> xlabel('waktu (jam)'), ylabel('Pressure')
>> legend('data','linier','kubik','interp1q')
>> title('Perbandingan metode interpolasi'
```



Gambar 8. 5 Perbandingan hasil interpolasi dengan tiga metode

8.5 Curve-Fitting

Pencocokkan kurva (*curve-fitting*) yang akan dibahas di sini ialah pencocokkan titik-titik data dengan suatu fungsi polinomial dengan metode pendekatan kuadrat terkecil (*least squares approximation*).

Tabel 8. 5

polyfit(x,y,n) menghitung vektor berisi koefisien polinomial orde-*n* yang mendekati titik-titik data di (x_i , y_i) **[p,E] = polyfit(x,y,n)** menghitung vektor polinomial **p** dan matriks **E** yang bisa digunakan oleh *command* polyval untuk mengestimasi error.

Mari kita coba dekati data tekanan udara seperti contoh sebelumnya dengan polinomial orde tiga, empat, dan lima.

```
>> t = [0  2  3  5  8.5  10  12];
>> pres = [660 900 400 300 500 50 300];
```

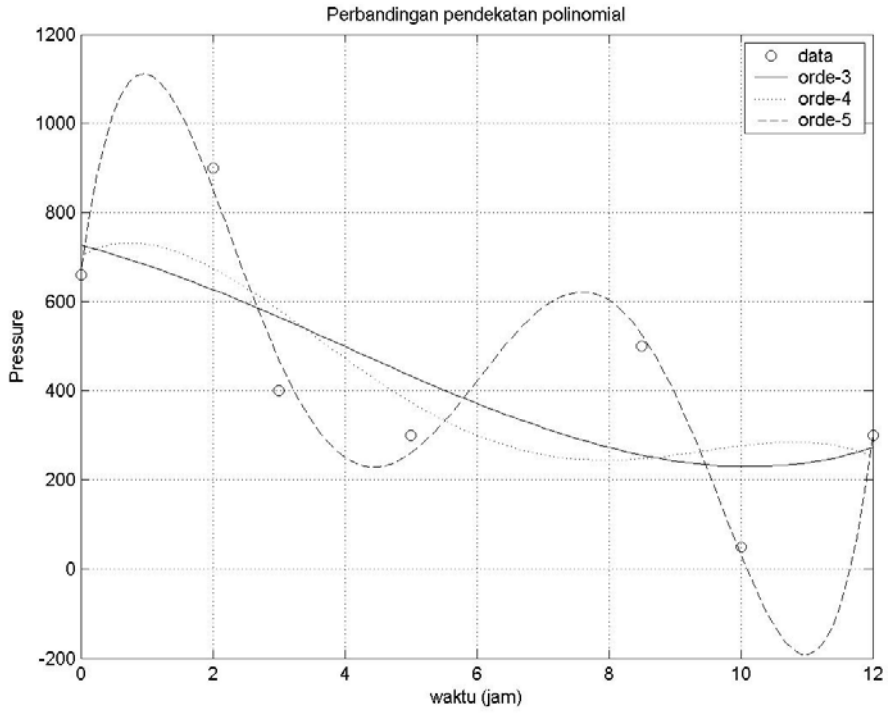
```
>> p3 = polyfit(t,pres,3)
p3 =
    0.5857   -6.9967  -38.3200  727.0393

>> p4 = polyfit(t,pres,4);
p4 =
   -0.3022    7.8645  -60.4717   77.6181  704.1170

>> p5 = polyfit(t,pres,5);
p5 =
  1.0e+003 *
0.0006  -0.0183   0.1908  -0.8055   1.0783   0.6648
```

Polinomial yang diwakili oleh **p3**, **p4**, dan **p5** ialah:

$$p_3(x) = 0{,}6x^3 - 7{,}0x^2 - 38{,}3x + 727{,}0$$
$$p_4(x) = -0{,}3x^4 + 7{,}9x^3 - 60{,}5x^2 + 77{,}6x + 704{,}1$$
$$p_5(x) = 0{,}6x^5 - 18{,}3x^4 + 190{,}8x^3 - 805{,}5x^2 + 1078{,}3x + 664{,}8$$

Berikutnya kita plot data dan ketiga kurva polinomial tersebut untuk dibandingkan.

```
>> tt = linspace(0,12,100);
>> kurva_p3 = polyval(p3,tt);
>> kurva_p4 = polyval(p4,tt);
>> kurva_p5 = polyval(p5,tt);

>> figure;
>> plot(t,pres,'ko',tt,kurva_p3,'k-', ...
tt,kurva_p4,'k:',tt,kurva_p5,'k--')
>> grid on;
>> xlabel('waktu (jam)'), ylabel('Pressure')
>> legend('data','orde-3','orde-4','orde-5')
>> title('Perbandingan pendekatan polinomial')
```



Gambar 8. 6 Perbandingan *curve-fitting* polinomial orde 3, 4, dan 5

8.6 Function Tool

Di dalam MATLAB telah terdapat perangkat (tool) untuk menggambar dan menganalisis fungsi secara praktis yang dikenal dengan "Function Tool". Untuk membuka perangkat ini, dari *command window* bisa kita ketikkan:

```
>> funtool
```

dan akan muncul tiga *window* berikut ini:

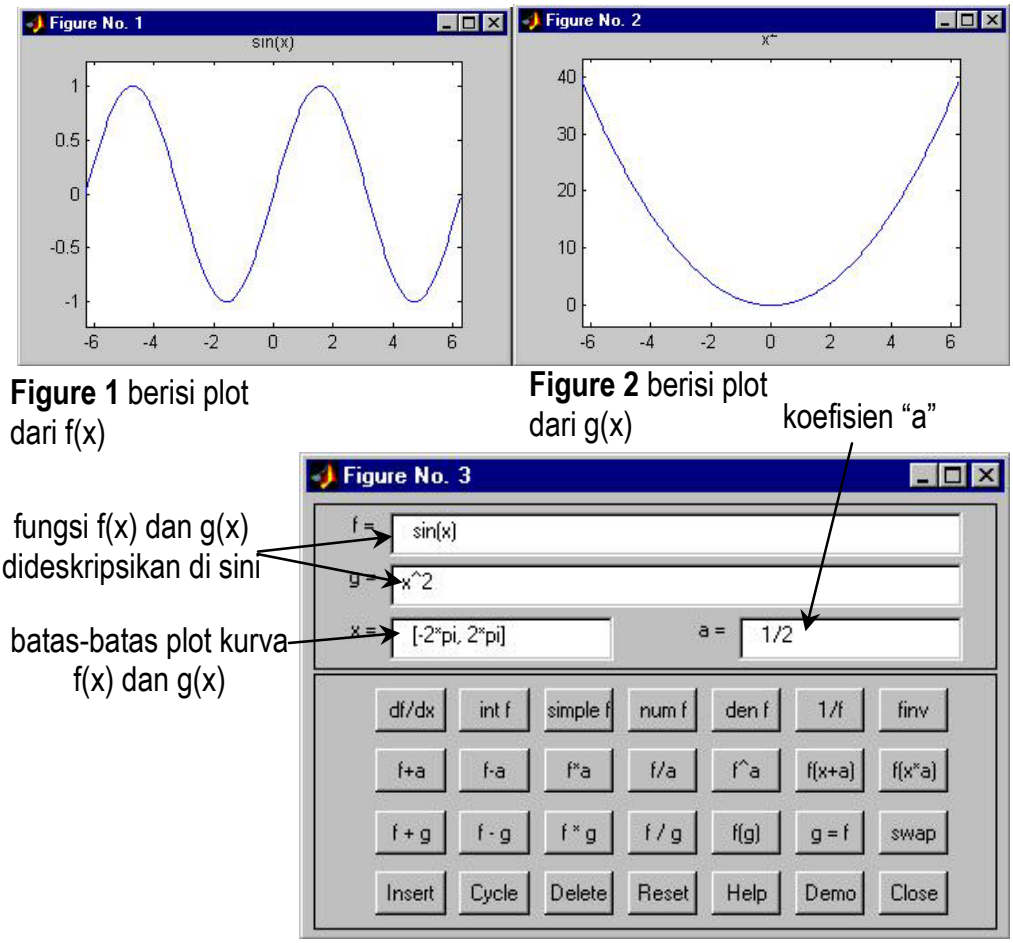


Figure 1 berisi plot dari f(x)

Figure 2 berisi plot dari g(x)

koefisien "a"

fungsi f(x) dan g(x) dideskripsikan di sini

batas-batas plot kurva f(x) dan g(x)

Figure 3 berfungsi sebagai papan kunci

Gambar 8. 7 Tiga *window* pada Function Tool

Berbagai operasi fungsi bisa kita lakukan dengan mengklik berbagai tombol yang ada di Figure3, misalkan:

Tabel 8. 6

df/dx	menghitung turunan $f(x)$ terhadap x: $\frac{df(x)}{dx}$
int f	menghitung integral tak tentu dari $f(x)$ terhadap x: $\int f(x)dx$
finv	menghitung fungsi invers dari $f(x)$
f+a, f-a, f*a, f/a, f^a	memanipulasi $f(x)$ dengan konstanta a
f(x+a), f(x*a)	memanipulasi variabel x dengan konstanta a
f+g, f-g, f*g, f/g	mengoperasikan fungsi $f(x)$ dan $g(x)$
f(g)	menghitung $f(g(x))$
g = f	menyalin $f(x)$ ke $g(x)$
swap	menukar antara $f(x)$ dengan $g(x)$

Soal Latihan

1. Nyatakanlah polinomial berikut dalam bentuk vektor baris:

$$p(x) = x^2 - 1 \qquad q(x) = x^4 - \frac{10}{9}x^2 + \frac{1}{9}$$

$$r(x) = \left(x^2 + \frac{3}{2}x + \frac{1}{2}\right)\left(x^3 - \frac{3}{2}x^2 + \frac{1}{2}x\right)$$

2. Evaluasilah ketiga polinomial pada no.1 tersebut pada nilai-nilai x = -1,5 , -1,2 , -0,9 , ... , 1,2 , 1,5

3. Buatlah plot dari ketiga polinomial pada no.1 tersebut pada rentang: $-1{,}5 \le x \le 1{,}5$. Buatlah inkremen x cukup kecil agar kurva terlihat mulus.

4. Hitunglah nol, minimum, dan maksimum dari fungsi rasional berikut ini pada rentang $-10 \le x \le 10$

$$F(x) = \frac{x-1}{x^2+1}$$

5. Hitunglah minimum dan maksimum dari fungsi dua variabel berikut ini pada rentang $-2 \le x \le 2$, $-2 \le y \le 2$

$$G(x, y) = \sin x \sin y + \sin xy$$

6. Berikut ini data distribusi pemakaian suatu telepon selama sebulan terakhir.

Distribusi pemakaian telepon			
Waktu pemakaian telepon (menit)		**Frekuensi**	
0	11	185	8
1	12	130	6
2	13	101	4
3	14	72	3
4	15	54	2
5	16	40	2
6	17	29	1
7	18	22	1
8	19	17	1
9	20	11	1
10		10	

Plot data distribusi ini dan dekatilah dengan dua metode interpolasi!

7. Misalkan terdapat tiga polinomial sebagai berikut:

$$m(x) = Ax + B \qquad n(x) = Cx^2 + Dx + E$$
$$k(x) = Fx^3 + Gx^2 + Hx + I$$

Cocokkanlah titik-titik data pada no.6 dengan kurva-kurva persamaan eksponensial berikut ini:

$$M(x) = e^{m(x)} = \exp(Ax)\exp B$$
$$N(x) = e^{n(x)} = \exp(Cx^2)\exp(Dx)\exp E$$
$$K(x) = e^{k(x)} = \exp(Fx^3)\exp(Gx^2)\exp(Hx)\exp(I)$$

- Hitunglah nilai A, B, C, .. , I dengan *command* **polyfit**.
- Plot titik-titik data beserta ketiga kurva tersebut di dalam satu gambar.

BAB 9

PERHITUNGAN INTEGRAL

Solusi numerik dari integral terbatas bisa dihitung secara efisien di MATLAB. Pertama, kita akan pelajari perhitungan integral dengan berbagai metode numerik. Berikutnya, kita kembangkan ke perhitungan integral lipat-2 dan lipat-3.

9.1 Menghitung Integral dengan Metode Numerik

Integral terbatas bisa diselesaikan secara numerik dengan MATLAB, yaitu:

$$q = \int_a^b f(x)dx$$

Terdapat sejumlah metode perhitungan integral secara numerik, misalkan: trapezoid, kuadratur, dll.

Tabel 9. 1

trapz(x,y)
menghitung integral dari **y** sebagai fungsi dari **x**. Vektor **x** dan **y** panjangnya harus sama. Nilai elemen dalam **x** sebaiknya disortir

trapz(x,A)
menghitung integral dari setiap kolom di **A** sebagai fungsi dari **x**; hasilnya berupa vektor baris berisi hasil integrasi. Jumlah kolom **A** harus sama dengan panjang **x**.

quad('fcn',a,b)
menghitung aproksimasi dari integral fungsi **fcn** pada interval $a \le x \le b$. Fungsi **fcn** harus didefinisikan terlebih dahulu dalam M-file.

quad('fcn',a,b,tol)
menghitung aproksimasi integral dari **fcn** dengan toleransi kesalahan sebesar **tol**.

quad('fcn',a,b,tol,trace,pic)
menghitung aproksimasi integral dari **fcn** dengan toleransi **tol**. Jika **trace** tidak nol, maka grafik yang mengilustrasikan integral akan diplot. Hasil integrasi dievaluasi pada **pic**. Kita bisa memberi nilai nol pada **tol** dan **trace** dengan matriks kosong **[ ]**.

quad8(...)
sama dengan *command* **quad**, tetapi menghitung dengan akurasi yang lebih tinggi.

quadl(...)
sama dengan *command* **quad8(...)**, namun untuk MATLAB versi terbaru.

Sebagai contoh, kita hitung integral berikut ini dengan metode numerik:

$$\int_0^2 e^{-x^3}\,dx$$

```
>> x = linspace(0,2,50);  % definisikan vektor x
>> y = exp(-x.^3);  % hitung nilai y
>> integral = trapz(x,y)s  % integralkan !
integral =
    0.8821
```

Dengan *command* **quad**, kita terlebih dahulu harus mendefinisikan fungsi dalam M-file:

```
function y = myfun(x)

y = exp(-x.^3);
```

Kita hitung integral tersebut dengan toleransi yang berbeda:

```
>> format long;  % format bilangan "long"
>> int_1 = quad('myfun',0,2,0.001), ...
int_2 = quad('myfun',0,2,0.00001)

int_1 =
   0.89309707589214
int_2 =
   0.89295225387894
```

Kita bandingkan akurasinya dengan **quad8**:

```
>> int_3 = quadl('myfun',0,2)
int_3 =
   0.89295351461757

>> format short;  % mengembalikan format ke "short"
```

Ini adalah hasil paling akurat yang bisa diperoleh MATLAB.

9.2 Integral Lipat-2

Kita bisa menghitung integral terbatas lipat-2 dengan menyelesaikan integralnya satu per satu menggunakan *command* **quad**.

Misalkan kita ingin menghitung integral berikut ini:

$$\int_0^1 \int_0^1 e^{-x^3-y}\, dydx$$

Pertama, kita buat M-file untuk fungsi ini:

```
function z = fungsiku(x,y)

z = exp(-x.^3-y);
```

Kedua, kita hitung integral-integral pada arah *y* untuk *x* yang tetap:

```
>> x = linspace(0,1,50);  % definisikan nilai x
```

```
>> for i = 1:50  % hitung integral unt setiap x(i)
       integral(i) = quad('fungsiku',0,1,[],[],x(i));
   end
```

Sekarang, kita memiliki 50 integral pada arah *y*. Ketiga, kita hitung integral arah *x*, misalkan dengan **trapz**.

```
>> Integral2 = trapz(x,integral)
Integral2 =
    0.5105
```

Cara lain yang lebih praktis untuk menghitung integral lipat-2 ialah menggunakan *command* berikut ini:

Tabel 9. 2

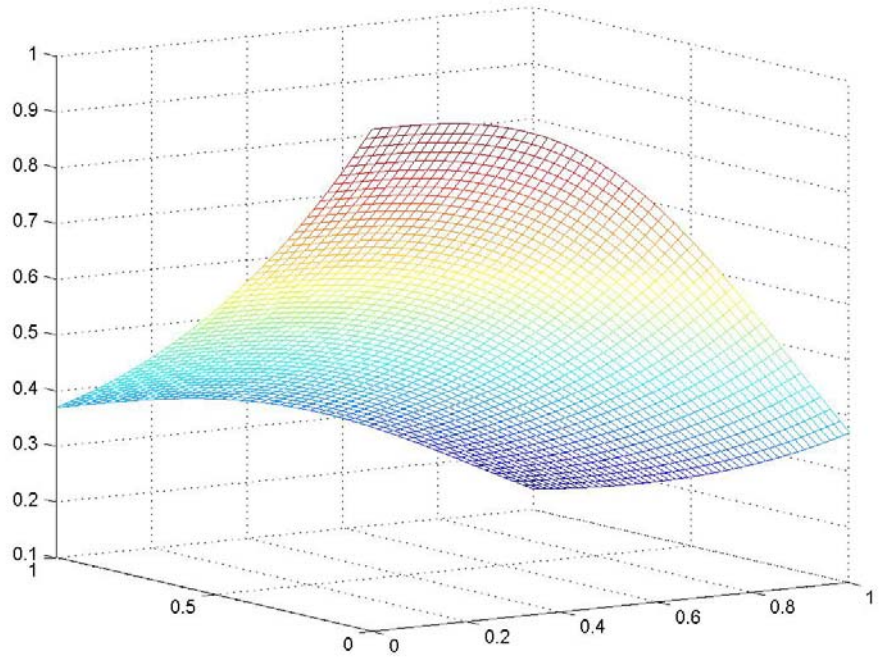
dblquad('fcn',xmin,xmax,ymin,ymax,tol)
menghitung integral lipat-2 untuk fungsi dua variabel **fcn** pada area segiempat **xmin** $\leq x \leq$ **xmax**, **ymin** $\leq y \leq$ **ymax**.

Untuk contoh integral di atas:

```
>> Integral_dobel = dblquad('fungsiku',0,1,0,1)
Integral_dobel =
    0.5104
```

Untuk mendapatkan gambaran dari fungsi tersebut, kita ketikkan:

```
>> [X,Y] = meshgrid(x,x);
>> Z = fungsiku(X,Y);
>> mesh(X,Y,Z)
```



Gambar 8. 1 Plot fungsi exp(-x^3-y) dengan domain [0,1]×[0,1]

9.3 Integral Lipat-3

Serupa dengan integral lipat-2, integral lipat-3 bisa kita selesaikan setahap demi setahap. Misalkan untuk integral berikut ini kita simpan dalam M-file:

$$\int_{-2}^{2}\int_{-2}^{2}\int_{-2}^{2}\sqrt{x^2+y^2+z^2}\,dzdydx$$

```
function w = funxyz(x,y,z)

w = sqrt(x.^2 + y.^2 + z.^2);
```

Kita akan selesaikan integral tersebut dengan metode yang berbeda dengan sebelumnya, yaitu menggunakan *nested-for*:

Pertama, kita definisikan batas-batas nilai *x*, *y*, dan *z*:

```
>> x = linspace(-2,2,50);  % definisikan nilai x
>> y = x;  % definisikan nilai y
>> z = x;  % definisikan nilai z
```

```
>> int_w = 0;
>> for i = 1:length(x)-1
     X = (x(i)+x(i+1))/2;
     dX = x(i+1)-x(i);

     for j = 1:length(y)-1
          Y = (y(j)+y(j+1))/2;
          dY = y(j+1)-y(j);

          for k = 1:length(z)-1
               Z = (z(k)+z(k+1))/2;
               dZ = z(k+1)-z(k);
               int_w = int_w + funxyz(X,Y,Z)*dX*dY*dZ;
          end
     end
end

>> int_w
int_w =
  122.9346
```

Cara lain yang lebih praktis ialah menggunakan *command* berikut ini:

Tabel 9. 3

triplequad('fcn',xmin,xmax,ymin,ymax,zmin,zmax,tol)
menghitung integral lipat-3 untuk fungsi tiga variabel **fcn** pada area balok **xmin** $\leq x \leq$ **xmax**, **ymin** $\leq y \leq$ **ymax**, **zmin** $\leq z \leq$ **zmax**, dengan toleransi kesalahan sebesar **tol**.

Untuk contoh integral di atas, kita hitung dengan toleransi 0,001:

```
>> Integral_tripel = triplequad('funxyz', ...
-2,2,-2,2,-2,2,0.001)

Integral_tripel =
  122.9577
```

Soal Latihan

1. Hitunglah integral terbatas berikut ini dengan metode trapezoid dan kuadratur:

   $$y = \int_{-10}^{10} \sqrt{100 - x^2}\, dx$$

   Bandingkan hasilnya dengan luas setengah lingkaran, yang merupakan bentuk area yang dibatasi persamaan tersebut:

   $$y = 50\pi$$

2. Hitunglah integral lipat-2 berikut ini:

   $$\int_{-4}^{4}\int_{-5}^{5} 10 - 2x^2 - y^2\, dy dx$$

3. Hitunglah integral lipat-3 dari fungsi tiga variabel berikut ini:

   $$w(x, y, z) = x^2 + xy + yz + z^2$$

   pada batas-batas $-1 \le x \le 1$, $-1 \le y \le 1$, $-1 \le z \le 1$.

DAFTAR PUSTAKA

Pärt-Enander, E. dan Sjöberg A., *The Matlab 5 Handbook*, Addison-Wesley, 1999.

Proakis, J.G. dan Manolakis, D.G., *Digital Signal Processing: Principles, Algorithms and Applications*, Macmillan, 1996.

www.mathworks.com/support

www.math.hmc.edu/calculus/tutorials/complex

LAMPIRAN 1

REFERENSI CEPAT

Berikut ini ringkasan *command*, sebagian telah dijelaskan pada isi buku, dan selebihnya bisa dieksplorasi sendiri dengan bantuan **help**.

Editing dan Kunci-kunci Khusus

↑ atau **Ctrl-P** dan ↓ atau **Ctrl-N**	melihat kembali *command* yang pernah diketik
← atau **Ctrl-B**	bergeser satu karakter ke kiri
→ atau **Ctrl-F**	bergeser satu karakter ke kanan
Ctrl-L atau **Ctrl-**←	bergeser satu kata ke kiri
Ctrl-R atau **Ctrl-**→	bergeser satu kata ke kanan
Ctrl-A atau **Home**	bergeser ke awal baris
Delete atau **BackSpace**	menghapus karakter
Ctrl-K	menghapus hingga akhir baris
Ctrl-C	menghentikan proses/kalkulasi yang sedang berjalan

Command Sistem Dasar

exit, quit	keluar dari MATLAB
diary	mencatat sesi dalam diary
save	menyimpan pekerjaan di ke dalam file
load	mengeluarkan kembali pekerjaan dari dalam file
type, dbtype	daftar file
what, dir, ls	daftar isi dari direktori
cd	mengubah direktori
pwd	melihat direktori saat ini

Help dan Demonstrasi

help	memunculkan help dari topik tertentu
lookfor	mencari teks
expo, demo	program demonstrasi
whatsnew	daftar fitur-fitur baru
info	informasi umum

Variabel

who, whos	daftar variabel yang aktif
clear	membersihkan variabel
size, length	ukuran matriks dan vektor
exist	eksistensi
format	mengatur format keluaran

Konstanta dan Variabel Standar

ans	jawaban terakhir yang tak tersimpan ke variabel
pi	π = 3.141 592 653 589 79
eps	akurasi relatif
realmax, realmin	bilangan terbesar dan terkecil
inf	tak hingga, didefinisikan sebagai 1/0
NaN	not-a-number, misalkan 0/0
i, j	unit imajiner, $\sqrt{-1}$
nargin, nargout	jumlah argumen masukan dan keluaran

Pencatat waktu

flops	jumlah flop (operasi floating point)
tic, toc, etime	menghidupkan dan mematikan pencatat waktu
clock, date	waktu dan tanggal saat ini
cputime	waktu sejak MATLAB dimulai

Fungsi Matematik

Fungsi elementer:

abs	nilai absolut
sign	fungsi signum
sqrt	akar kuadrat
pow2	kuadrat
exp	fungsi eksponensial
log, log2, log10	fungsi logaritmik
sin, cos, tan, cot, sec, csc	fungsi trigonometrik
asin, acos, atan2, atan, acot, asec, acsc	inversi fungsi trigonometrik
sinh, cosh, tanh, coth, asinh, acosh, atanh, acoth, sech, csch, asech, acsch	fungsi hiperbolik dan invers-hiperbolik

Fungsi matematika lanjut:

legendre	fungsi Legendre
bessel, bessely	fungsi Bessel
gamma, gammaln, gammainc	fungsi Gamma
beta, betaln, betainc	fungsi Beta
expint	integral eksponensial
erf, erfinv, erfc, erfcx	error-function
ellipke, ellipj	integral eliptik

Transformasi koordinat:

cart2pol, pol2cart	kartesian dan polar
cart2sph, sph2cart	kartesian dan bola

Operasi Bilangan Bulat dan Floating Point

round, fix, floor, ceil	pembulatan
rat	pendekatan ke bilangan rasional
rats	mengubah bilangan rasional ke string
rem	sisa pembagian
gcd	faktor pembagi terbesar
lcm	kelipatan pengali terkecil

Bilangan Kompleks

real, imag	komponen riil dan imajiner
conj	konjugasi
angle	sudut fase
unwrap	mengatur kembali argumen sudut
cplxpair	pasangan kompleks

Vektor dan Matriks

:	operator indeks
linspace, logspace	pembangkit vektor
eye	matriks identitas
ones, zeros	matriks satuan dan matriks nol
[ ]	matriks kosong
rand, randn	matriks random
diag	matriks diagonal
triu, tril	matriks segitiga
fliplr, flipud, rot90, reshape	membentuk-ulang matriks
hilb, invhilb, toeplitz, compan, gallery, hadamard, hankel, magic, pascal, rosser, vander, wilkinson	matriks-matriks khusus

Operasi Matriks

+ –	penjumlahan dan pengurangan
* .* **cross, dot, kron**	perkalian
/ \ ./ .\	pembagian dan pembagian terbalik
‘ .’	transposisi, konjugasi
^ .^	pangkat
>, <, >=, <=, ==, ~=	operator pembandingan
and, or, not, **&, |, ~, xor**	operator logika

Fungsi Matriks

det, trace, rank	determinan, trace, dan rank
inv, pinv	invers dan pseudo-invers
orth, null	basic subspaces
subspace	sudut antara subspaces
expm, logm, sqrtm, funm, polyvalm	fungsi-fungsi matriks
size, length	ukuran dan panjang matriks / vektor
any, all, isnan, isinf, isieee, issparse, isstr, isempty, finite	fungsi logika

Antarmuka dengan Pengguna

disp	memunculkan nilai atau teks di *command window*
input	input dari keyboard
ginput	membaca koordinat
pause	eksekusi berhenti sementara
waitforbuttonpress	menunggu aksi dari pengguna
format	format keluaran di *command window*
menu	mengeluarkan menu dengan pilihan
lasterr	memunculkan pesan error terakhir

Grafik

Grafik 2-D dan 3-D:

plot	plot grafik 2-dimensi
plot3	plot garis dalam 3-dimensi
fplot	plot fungsi
subplot	membagi *figure* yang ada menjadi subplot
errorbar	plot grafik dengan error-bar
comet, comet3	plot beranimasi, 2-D, 3-D
polar	plot dalam koordinat polar
semilogx, semilogy, loglog	plot logaritmik
quiver, feather, compass, rose	grafik bilangan kompleks
stem	plot data diskrit
hist, bar, stairs	plot histogram, diagram batang dan tangga

Mengatur grafik:

figure	menciptakan atau memunculkan suatu *figure*
clf	membersihkan *figure*
hold	menahan plot yang ada agar tidak hilang tertimpa plot baru
subplot	membagi *figure* yang ada menjadi subplot
clc	membersihkan tampilan *command window*

home	mengembalikan kursor ke pojok kiri-atas
axis	mengatur sumbu plot
zoom	memperbesar / memperkecil (untuk grafik 2-D)
grid	memunculkan / menghilangkan grid
title, xlabel, ylabel, zlabel	menuliskan berbagai teks di dalam plot
text	menuliskan teks di manapun di dalam plot
gtext	menempatkan teks dengan mouse
ginput	membaca koordinat di dalam plot
rbbox	memindahkan suatu area segi empat
hidden	memperlihatkan / menyembunyikan permukaan
view	mengatur posisi dan sudut penglihatan

Plot permukaan dan kontur:

contour	plot kontur
contour3	plot kontur dalam ruang 3-D
clabel	memberi tanda pada garis kontur
meshgrid	membuat jalinan titik untuk plot 3-D
cylinder, sphere	grid untuk geometri silinder dan bola
surf	plot permukaan (*surface*)
mesh	plot mesh
meshc, meshz, waterfall	plot mesh dengan garis referensi
surfl, surfc, surfnorm	plot permukaan dengan pencahayaan khusus, kontur, dan garis normal
pcolor	plot permukaan dilihat dari atas
fill, fill3	mengisi poligon
slice	plot fungsi tiga variabel

Suara

sound	membunyikan suara
wavwrite, wavread	menulis dan membaca file .WAV

Pemrograman

Conditional statements:

```
if kondisi
   command
end
```

```
if kondisi
   command_A
else
   command_B
end
```

```
if kondisi_1
   command_1
elseif kondisi_2
   command_2
elseif kondisi_3
   command_3
   ...
end
```

```
switch  nama_variabel
case { kondisi_1, kondisi_2, ... }
   command_1
case { kondisi_A, kondisi_B, ... }
   command_2
case { kondisi_X, kondisi_Y, ... }
   command_3
...
default
   command
end
```

Loop:

```
for  variabel = awal : inkremen : akhir
   command
end

while  kondisi
   command
end
```

Lain-lain:

%	penanda komentar
break	keluar dari suatu loop
return	keluar dari program
continue	melanjutkan loop tanpa menjalankan *command* di bawahnya
global	mendeklarasikan variabel global
nargin	jumlah argumen input
nargout	jumlah argumen output

Analisis Data dan Statistik

max, min	maksimum dan minimum
sum, cumsum	jumlah dan jumlah kumulatif
prod, cumprod	produk dan produk kumulatif
diff, gradient, del2	difference
mean, median	rata-rata dan median
std	deviasi standar
var, cov	variansi dan kovariansi
skew, kurt	*skewness* dan *kurtosis*
corrcoef	matriks korelasi
sort	sortir
hist, bar, stairs	histogram, diagram batang dan tangga
stem	plot data diskrit
fft, ifft	transformasi Fourier

Analisis Fungsi dan Interpolasi

fzero	nol dari fungsi
fmin, fminbnd	minimum dari fungsi satu variabel
fmins, fminsearch	minimum dari fungsi multi variabel
interp1, interp1q	interpolasi
interpft	interpolasi Fourier
spline	interpolasi spline

Polinomial dan Curve-Fitting

polyval, polyvalm	mengevaluasi polinomial
conv, deconv	konvolusi, perkalian polinomial
residue	menghitung residu
polyder	turunan pertama dari fungsi polinomial
poly	polinomial karakteristik
compan	companion matrix
polyfit	aproksimasi polinomial

Integral

trapz, quad, quad8, quadl	menghitung integral terbatas
dblquad	menghitung integral terbatas lipat-2
triplequad	menghitung integral terbatas lipat-3

LAMPIRAN 2

PENGENALAN BILANGAN KOMPLEKS

Bilangan kompleks yang merupakan perluasan dari bilangan riil, mengandung semua akar-akar dari persamaan kuadrat. Jika kita definisikan i sebagai solusi dari persamaan $x^2 = -1$, atau dengan kata lain:

$$i := \sqrt{-1}$$

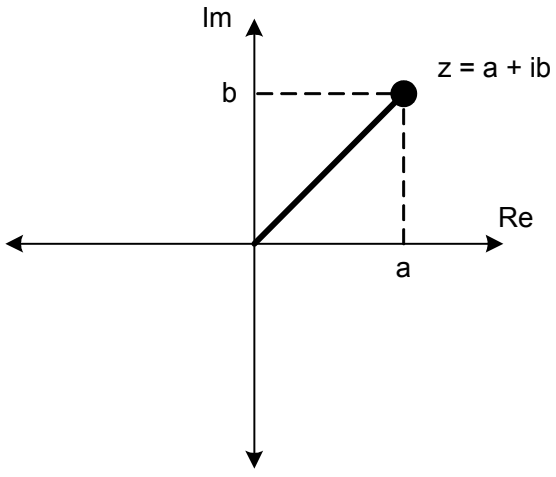
maka himpunan semesta bilangan kompleks C dinyatakan dalam **bentuk standar** sebagai:

$$\{a + ib \mid a, b \in R\}$$

Kita biasa menggunakan notasi $z = a + ib$ untuk menyatakan bilangan kompleks. Bilangan a disebut **komponen riil** dari z (Re z), sementara b disebut **komponen imajiner** dari z (Im z).

Dua bilangan kompleks dikatakan sama jika dan hanya jika komponen riilnya sama dan komponen imajinernya juga sama.

Kita menggambarkan bilangan kompleks dengan mengasosiasikan $z = a + ib$ dengan titik (a,b) pada **bidang kompleks**.



Operasi Dasar

Operasi dasar dari bilangan kompleks didefinisikan berikut ini:

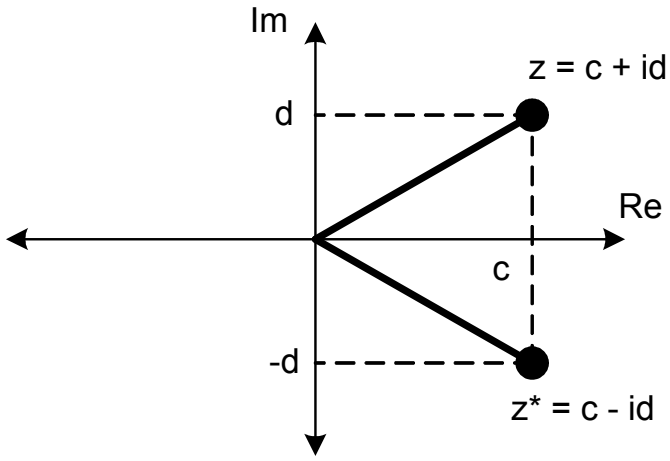
$$(a + ib) + (c + id) = (a + c) + i(b + d)$$

$$(a+ib)-(c+id)=(a-c)+i(b-d)$$
$$(a+ib)(c+id)=(ac-bd)+i(bc+ad)$$

$$\frac{a+ib}{c+id}=\frac{a+ib}{c+id}\frac{c-id}{c-id}=\frac{ac+bd}{c^2+d^2}+i\frac{bc-ad}{c^2+d^2}$$

Ketika membagi $a + ib$ dengan $c + id$, kita merasionalkan penyebut mengingat:
$(c+id)(c-id)=c^2-icd+icd-i^2d^2=c^2+d^2$

Bilangan $c + id$ dan $c - id$ disebut **konjugat kompleks**.



Jika $z = c + id$, kita gunakan notasi z^* untuk $c - id$.

Ditinjau sebagai vektor dalam bidang kompleks, $z = a + ib$ memiliki **magnitude**:

$$|z|=\sqrt{a^2+b^2}$$

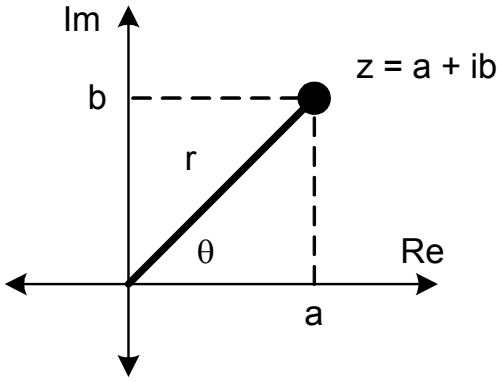
Magnitude disebut juga **modulus** atau **nilai absolut**. Perlu diingat bahwa: $zz^*=|z|^2$

Contoh:
$(2+3i)(2-3i)=4-6i+6i-9i^2=4+9=13$
$|2+3i|=|2-3i|=\sqrt{4+9}=\sqrt{13}$

Bentuk Polar

Untuk $z = a + ib$, kita dapatkan:

$$a = r\cos\theta \qquad b = r\sin\theta$$



dan kita dapatkan pula:

$$r = |z| = \sqrt{a^2 + b^2}$$

$$\theta = \arctan\frac{b}{a}$$

Kemudian, $z = r\cos\theta + i \cdot r\sin\theta$

Dengan persamaan Euler: $e^{i\theta} = \cos\theta + i\sin\theta$
kita dapatkan **bentuk polar**:

$$z = re^{i\theta}$$

Di sini, r disebut **magnitude** dari z dan θ disebut **argumen** dari z (arg z). Nilai argumen ini tidak unik; kita bisa tambahkan kelipatan bulat dari 2π ke dalam θ tanpa mengubah z.

Kita definisikan Arg z, yaitu **nilai prinsipil** dari argumen yang berada dalam selang $-\pi < \text{Arg } z \le \pi$. Nilai prinsipil ini unik untuk setiap z tetapi menciptakan diskontinuitas pada sumbu riil negatif ketika meloncat dari π ke $-\pi$. Loncatan ini disebut **branch cut**.

Contoh:

$$e^{i\pi} = \cos\pi + i\sin\pi = -1$$

$$3e^{i\pi/2} = 3\left(\cos\frac{\pi}{2} + i\sin\frac{\pi}{2}\right) = 3i$$

$$-2e^{i\pi/6} = -2\left(\cos\frac{\pi}{6} + i\sin\frac{\pi}{6}\right) = -\sqrt{3} - i$$

Perkalian dan pembagian bilangan kompleks dalam bentuk polar menjadi lebih mudah. Jika $z_1 = r_1 e^{i\theta_1}$ dan $z_2 = r_2 e^{i\theta_2}$ maka:

$$z_1 z_2 = r_1 r_2 e^{i(\theta_1+\theta_2)}$$

$$\frac{z_1}{z_2} = \frac{r_1}{r_2} e^{i(\theta_1-\theta_2)}$$

Jika $z = re^{i\theta}$ maka $z^* = re^{-i\theta}$, dan juga $zz^* = \left(re^{i\theta}\right)\left(re^{-i\theta}\right) = r^2$

Contoh:
Untuk menghitung $(1+i)^8$, kita terlebih dahulu bisa menuliskan $(1+i)$ dalam bentuk polar sebagai $\sqrt{2}e^{i\pi/4}$, kemudian:

$$\left(\sqrt{2}e^{i\pi/4}\right)^8 = \left(\sqrt{2}\right)^8 e^{i8\pi/4} = 16e^{i2\pi} = 16$$

Akar-akar dari Satu

Persamaan

$$z^n = 1$$

memiliki n-buah solusi berbentuk kompleks, disebut akar pangkat-n dari satu. Kita ketahui bahwa akar dari persamaan tersebut memiliki magnitude 1.

Sekarang misalkan $z = e^{i\theta}$, maka

$$\left(e^{i\theta}\right)^n = 1 \Leftrightarrow e^{in\theta} = e^{i(2\pi k)}$$

$$\Leftrightarrow n\theta = 2\pi k \Leftrightarrow \theta = \frac{2\pi k}{n}$$

$\left(e^{i\theta}\right)^n = e^{in\theta}$, dengan persamaan Euler, menghasilkan **formula de Moivre**:

$$(\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta$$

Sehingga akar pangkat-n dari satu memiliki bentuk:

$$z = e^{i\frac{2\pi k}{n}}$$

Terdapat n buah akar yang berbeda, di mana akar-akar tersebut terdistribusi merata pada lingkaran satuan dalam bidang kompleks.

Contoh:

Kita akan menghitung $\sqrt[3]{1}$.

$$\sqrt[3]{1} = \left(e^{i2\pi k}\right)^{1/3} = e^{i2\pi k/3} \qquad k = -1, 0, 1$$

untuk k = -1 diperoleh:

$$e^{-i2\pi/3} = \cos\frac{2\pi}{3} - i\sin\frac{2\pi}{3} = -\frac{1}{2} - i\frac{1}{2}\sqrt{3}$$

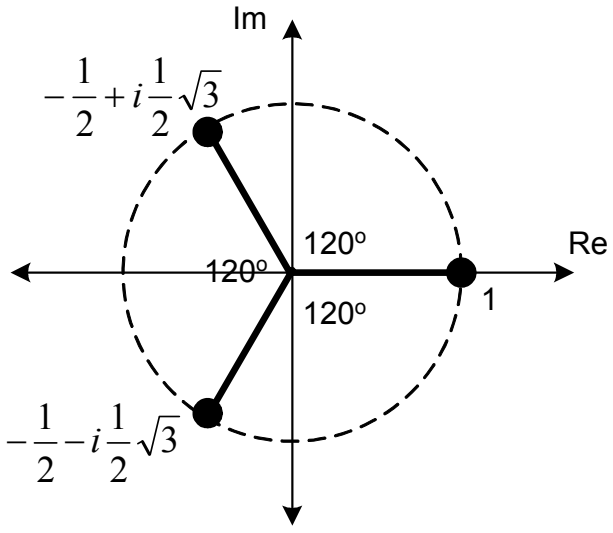
untuk k = 0 diperoleh:

$$e^{0} = 1$$

untuk k = 1 diperoleh:

$$e^{i2\pi/3} = \cos\frac{2\pi}{3} + i\sin\frac{2\pi}{3} = -\frac{1}{2} + i\frac{1}{2}\sqrt{3}$$

Bila kita gambarkan pada lingkaran satuan di bidang kompleks:



Sekarang kita akan menghitung solusi dari $z^5 = 32$.

$$z = \sqrt[5]{32} = \left(32e^{i2\pi k}\right)^{1/5} = 32^{1/5}e^{i2\pi k/5}$$

untuk k = -2, -1, 0, 1, 2

untuk k = -2, diperoleh:

$$z_1 = 2e^{-i4\pi/5} = 2\left(\cos\frac{4\pi}{5} - i\sin\frac{4\pi}{5}\right) = -1{,}618 - 1{,}176i$$

untuk k = -1, diperoleh:

$$z_2 = 2e^{-i2\pi/5} = 2\left(\cos\frac{2\pi}{5} - i\sin\frac{2\pi}{5}\right) = 0{,}618 - 1{,}902i$$

untuk k = 0, diperoleh:

$$z_3 = 2e^{0} = 2$$

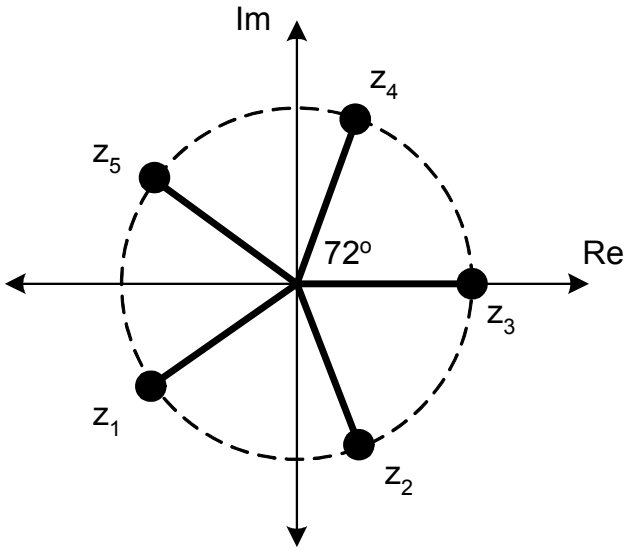
untuk k = 1, diperoleh:

$$z_4 = 2e^{i2\pi/5} = 2\left(\cos\frac{2\pi}{5} + i\sin\frac{2\pi}{5}\right) = 0{,}618 + 1{,}902i$$

untuk k = 2, diperoleh:

$$z_5 = 2e^{i4\pi/5} = 2\left(\cos\frac{4\pi}{5} + i\sin\frac{4\pi}{5}\right) = -1{,}618 + 1{,}176i$$

Digambarkan dalam bidang kompleks:



LAMPIRAN 3

JAWABAN SOAL LATIHAN

Berikut ini salah satu alternatif jawaban untuk memecahkan berbagai masalah dalam latihan.

Bab 2:

1.
```
>> 12/3.5, (3+5/4)^2
ans =
    3.4286
ans =
   18.0625

>> (.25^2 + .75^2)^(1/2), 2/(6/.3)
ans =
    0.7906
ans =
    0.1000
```

2.
```
>> A=25, B=50, C=125, D=89
A =
    25
B =
    50
C =
   125
D =
    89

>> X = A+B+C, Y = A/(D+B)
X =
   200
Y =
    0.1799

>> Z = D^(A/B) + C
Z =
  134.4340
```

3. **luas** : valid, **kel_1** : valid,
   2_data : tidak valid, karena diawali dengan angka,
   diff:3 : tidak valid, karena mengandung titik-dua,
   Time : valid, **time** : valid, **time_from_start** : valid,
   10_hasil_terakhir : tidak valid, karena diawali dengan angka**,**
   nilai-awal : tidak valid, karena mengandung tanda minus

4.
```
>> x=pi/6; y=.001;
>> sqrt(y), exp(-x), sin(x)
ans =
    0.0316
ans =
    0.5924
ans =
    0.5000

>> cos(2*x), tan(3*x)
ans =
    0.5000
ans =
  1.6331e+016

>> log10(y), log2(y), log(y)
ans =
    -3
ans =
   -9.9658
ans =
   -6.9078
```

5.
```
>> clear
>> p = 9 + 16*i; q = -9 + 16*i;
>> r=p*q, s=p/q, p-r
r =
  -337
s =
   0.5193 - 0.8546i
ans =
  3.4600e+002 +1.6000e+001i

>> r+s, p^2, sqrt(q)
ans =
 -3.3648e+002 -8.5460e-001i
ans =
 -1.7500e+002 +2.8800e+002i
ans =
   2.1630 + 3.6985i

>> abs(p), angle(p)
ans =
   18.3576
ans =
    1.0584

>> abs(q), angle(q)
ans =
   18.3576
ans =
    2.0832
```

```
>> abs(r), angle(r)
ans =
   337
ans =
    3.1416

>> abs(s), angle(s)
ans =
     1
ans =
   -1.0248
```

Bab 3:

1.
```
>> vektor_1=[10 20 30 40]
vektor_1 =
    10    20    30    40

>> vektor_2=[-5;
-15;
-40]
vektor_2 =
    -5
   -15
   -40

>> matriks=[1 3 5 0;
3 1 3 5;
5 3 1 3;
0 5 3 1]
matriks =
     1     3     5     0
     3     1     3     5
     5     3     1     3
     0     5     3     1
```

2.
```
>> A=[4 8;2 4], B=[1 1;1 -1]
A =
     4     8
     2     4
B =
     1     1
     1    -1

>> C=[A B]
C =
     4     8     1     1
     2     4     1    -1
```

```
>> W=[B B;B -B]
W =
     1     1     1     1
     1    -1     1    -1
     1     1    -1    -1
     1    -1    -1     1
```

3. a) `>> size(vektor_1), size(vektor_2), size(matriks)`

```
ans =
     1     4
ans =
     3     1
ans =
     4     4
```

Sehingga ukuran vektor/matriks ialah masing-masing: 1×4, 3×1, dan 4×4.

b) `>> prod(size(vektor_1)), ...`
`prod(size(vektor_2)), prod(size(matriks))`

```
ans =
     4
ans =
     3
ans =
    16
```

4. `>> 5.*eye(4)`

```
ans =
     5     0     0     0
     0     5     0     0
     0     0     5     0
     0     0     0     5
```

```
>> [5.*ones(2), zeros(2);
-5.*eye(2), 5.*(ones(2)-eye(2))]
ans =
     5     5     0     0
     5     5     0     0
    -5     0     0     5
     0    -5     5     0
```

5. `>> bil_acak = sqrt(0.2).*randn(1,100) + 1`

6. `>> M = [1 5:5:20; 2.^[0:4]; -3:3:9; 2.^[5:-1:1];`
`5 -5 5 -5 5]`

```
M =
     1     5    10    15    20
     1     2     4     8    16
    -3     0     3     6     9
    32    16     8     4     2
```

```
     5    -5     5    -5     5

>> M(1,:)
ans =
     1     5    10    15    20

>> M(:,3)
ans =
    10
     4
     3
     8
     5

>> M(3:5,2:4)
ans =
     0     3     6
    16     8     4
    -5     5    -5

>> [M(1,1) M(2,2) M(3,3) M(4,4) M(5,5)]
ans =
     1     2     3     4     5
```

7.
```
>> x = -10:10
>> y = 7.5:-0.5:0
>> z = 1:3:100

>> format long
>> w = logspace(-3,6,10)
>> format short
```

8.
```
>> N = M(:,1:4)
N =
     1     5    10    15
     1     2     4     8
    -3     0     3     6
    32    16     8     4
     5    -5     5    -5

>> fliplr(N)
>> flipud(N)
>> reshape(N,10,2)
>> reshape(N,4,5)
```

Bab 4:

1.
```
>> M=[10 20; 5 8]; N=[-1 1;1 -1];
>> M+N, M-N, N+9
ans =
     9    21
     6     7
ans =
    11    19
     4     9
ans =
     8    10
    10     8

>> M*N, N*M
ans =
    10   -10
     3    -3
ans =
    -5   -12
     5    12
```

2.
```
>> a=[0 5 5]; b=[1 1 1];
>> dot(a,b)
ans =
    10

>> cross(a,b), cross(b,a)
ans =
     0     5    -5
ans =
     0    -5     5
```

3. Pertama, definisikan matriks **A** yang berisi koefisien yang melekat pada variabel x, y, z.

```
>> A=[1 2 -3; 4 5 6; 7 8 9];
```

Kedua, definisikan vektor **b** yang merupakan ruas kanan dari persamaan.

```
>> b=[-7; 11; 17]
```

Ketiga, hitung solusi dengan invers matriks.

```
>> x=inv(A)*b
x =
    1.0000
   -1.0000
    2.0000
```

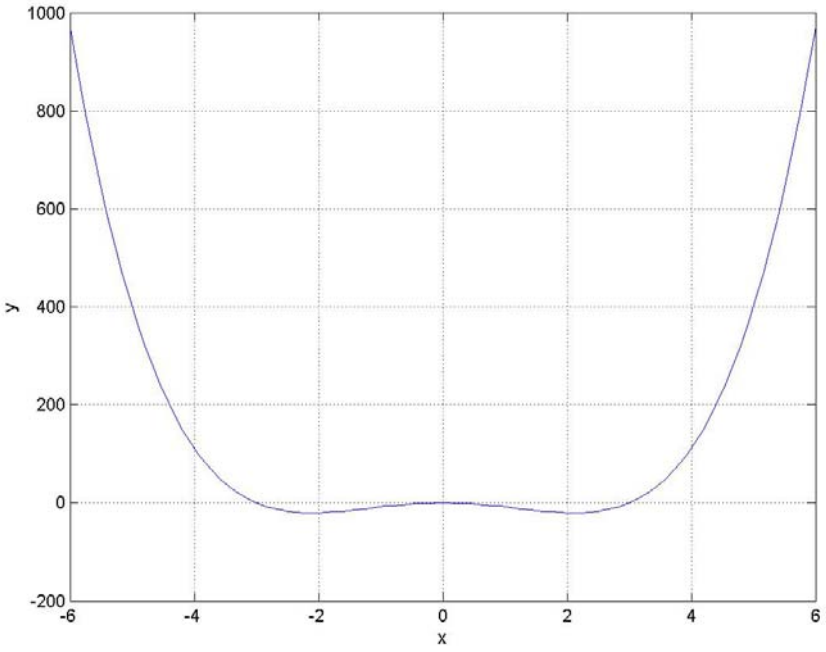
Sehingga diperoleh: $x = 1,\ y = -1,\ z = 2$.

4.
```
>> x = [ -5:0.05:5 ]';  % membuat vektor x
>> y = sqrt(25-x.^2);   % menghitung y

>> pj = length(x);  % menghitung panjang vektor x
>> awal = round(pj/2); akhir = awal + 1/0.05;
>> % menentukan indeks untuk x=0 hingga x=1
>> [ x(awal:akhir), y(awal:akhir) ]
ans =
         0    5.0000
    0.0500    4.9997
    0.1000    4.9990
    0.1500    4.9977
    0.2000    4.9960
    0.2500    4.9937
    0.3000    4.9910
    0.3500    4.9877
    0.4000    4.9840
    0.4500    4.9797
    0.5000    4.9749
    0.5500    4.9697
    0.6000    4.9639
    0.6500    4.9576
    0.7000    4.9508
    0.7500    4.9434
    0.8000    4.9356
    0.8500    4.9272
    0.9000    4.9183
    0.9500    4.9089
    1.0       4.8990
```

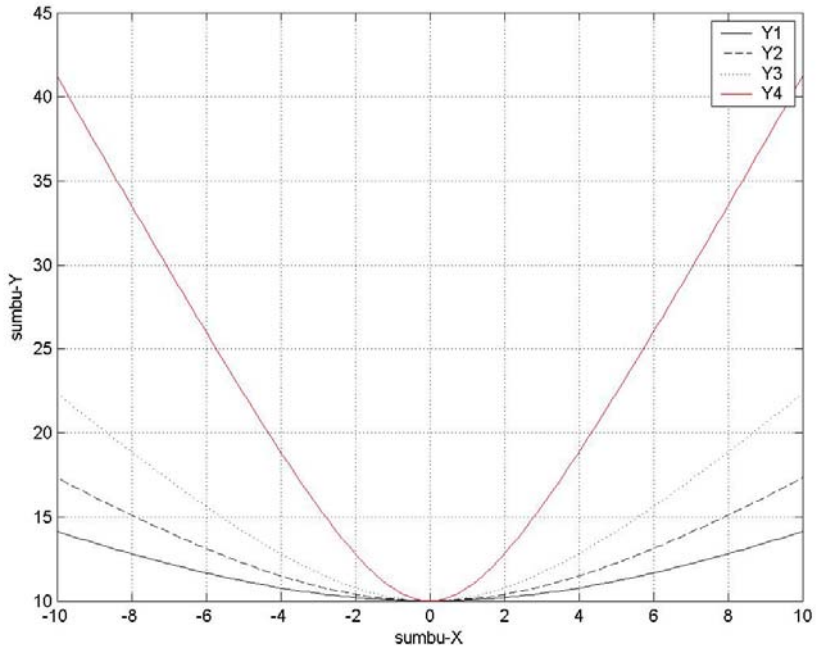
5.
```
>> x = -5:0.1:5; % membuat vektor x
>> sinus=sinh(x); cosinus=cosh(x); tangent=tanh(x);
>> clc  % membersihkan layar
>> disp('Tabel hiperbolik-trigonometri:'), ...
disp('x      sinh      cosh      tanh'), ...
disp('--------------------------------')
>> [x'  sinus' cosinus' tangent']
ans =
  -5.000   -74.2032   74.2099   -0.9999
  -4.900   -67.1412   67.1486   -0.9999
  -4.800   -60.7511   60.7593   -0.9999
  ....
  -0.100   -0.1002    1.0050    -0.0997
   0        0         1.0000     0
   0.100    0.1002    1.0050     0.0997
   0.200    0.2013    1.0201     0.1974
  ....
   4.900   67.1412   67.1486     0.9999
   5.000   74.2032   74.2099     0.9999
```

Bab 5:

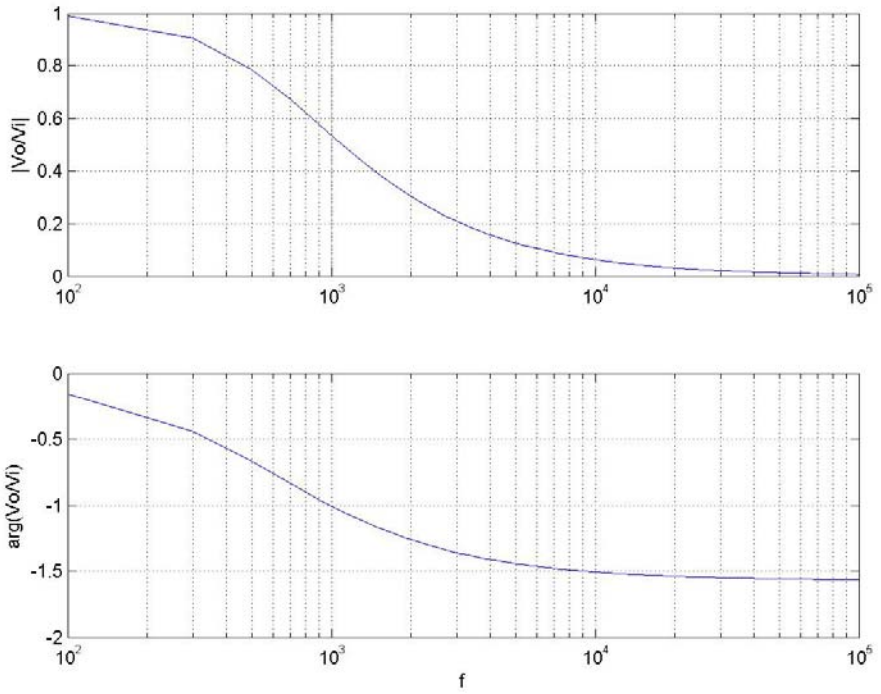
1.
```
>> x = linspace(-6,6,100); % mendefinisikan x
>> y = x.^4 - 9.*x.^2;  % menghitung y
>> figure; plot(x,y); grid on; % membuat plot x-y
>> xlabel('x'), ylabel('y');
```



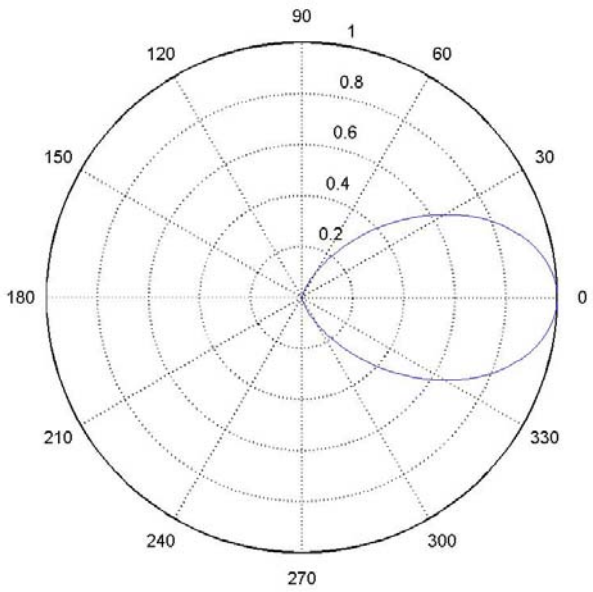
2.
```
>> x = linspace(-10,10,150);   % definisikan x
>> y1= sqrt(100 + x.^2);       % hitung y1 s.d. y4
>> y2= sqrt(100 + 2.*x.^2);
>> y3= sqrt(100 + 4.*x.^2);
>> y4= sqrt(100 + 16.*x.^2);

>> figure;
>> plot(x,y1,'k-',x,y2,'k--',x,y3,'k:',x,y4,'r-');
>> grid on; % membuat plot
>> xlabel('sumbu-X'), ylabel('sumbu-Y')
>> legend('Y1','Y2','Y3','Y4')
```



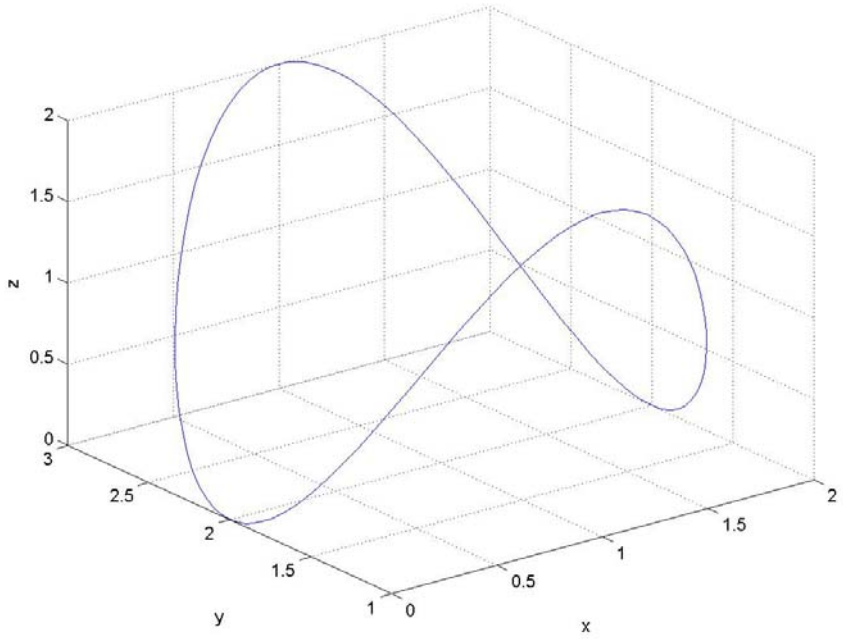
3.
```
>> clear
>> f=linspace(100,1e5,500); % finisikan frekuensi
>> F=4e3; % frekuensi cut-off
>> Vo_Vi = 1./(1+j*2*pi.*f./F); % menghitung Vo/Vi

>> figure;
>> subplot(2,1,1); semilogx(f,abs(Vo_Vi));
>> % plot respon amplituda
>> grid on; ylabel('|Vo/Vi|');

>> subplot(2,1,2); semilogx(f,angle(Vo_Vi));
>> % plot respon fasa
>> grid on; ylabel('arg(Vo/Vi)'); xlabel('f');
```



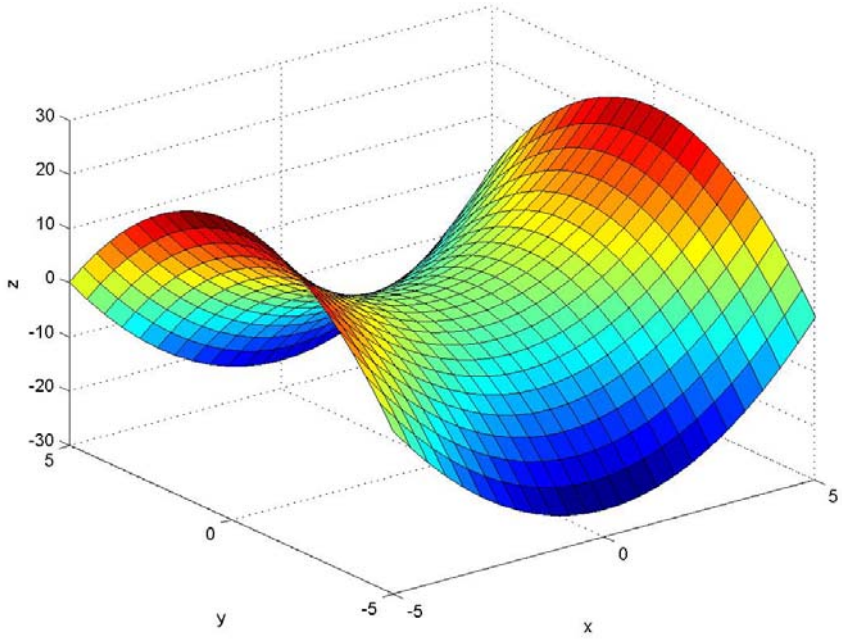
4.
```
>> phi=linspace(-pi/2,pi/2,100);
>> % definisikan rentang sudut phi
>> U = cos(phi).^3; % menghitung U
>> figure; polar(phi,U); grid on;
```



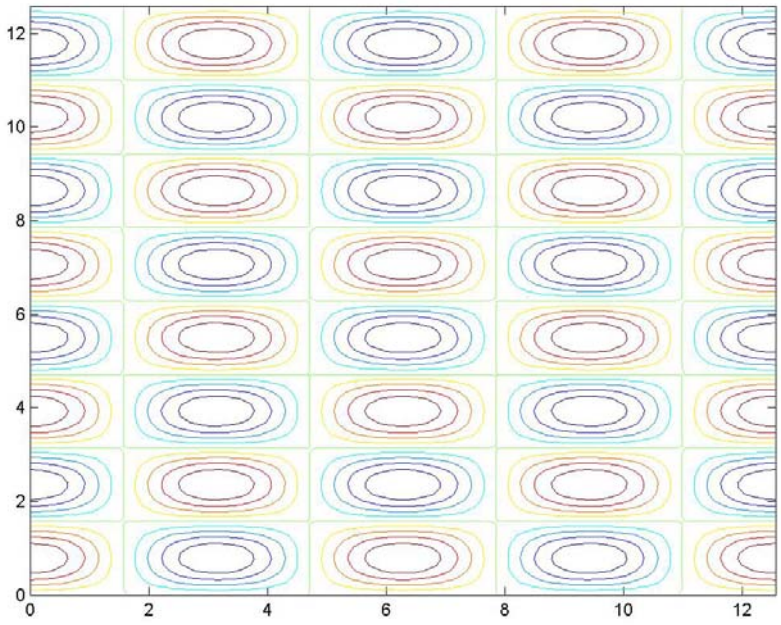
5.
```
>> t = linspace(0,2*pi,100);
>> % definisikan parameter t
>> x = 1 + cos(t); y = 2 + sin(t); z = 1 -
cos(2.*t);
>> % hitung x,y,z

>> figure; plot3(x,y,z);
>> grid on; xlabel('x'), ylabel('y'), zlabel('z')
```



6.
```
>> clear;
>> x = linspace(-5,5,25); y=x;
>> % definisikan batas x dan y
>> [X,Y]=meshgrid(x,y);
>> % buat jalinan titik pada bidang xy
>> Z = X.^2 - Y.^2; % hitung z

>> figure; surf(X,Y,Z);
>> grid on; xlabel('x'), ylabel('y'), zlabel('z')
```



7.
```
>> clear;
>> x = linspace(0,4*pi,100); y=x;
>> [X,Y]=meshgrid(x,y);
>> Z = cos(X).*sin(2.*Y);
>> figure; contour(x,y,Z);
```



8.
```
>> Fs = 2e3; % frekuensi sampling 2 kHz
>> t = 0:1/Fs:1; % durasi tone = 1 detik
>> f = [262 294 330 349 392 440 495 524];
>> suara=[];
```

```
>> for i=1:8
    suara = [suara cos(2*pi*f(i).*t)];
end

>> sound(suara,Fs);
>> wavwrite(suara,Fs,'file_suara.wav')
```

Bab 6:

1.
```
% Program untuk menghitung volume &
% luas permukaan balok: prog_balok.m

panjang=5; lebar=3; tinggi=6.5;

volume = panjang * lebar * tinggi
luas = 2* (panjang*lebar + panjang*tinggi + ...
lebar*tinggi)
```

Kita jalankan program tersebut:

```
>> prog_balok
volume =
   97.5000
luas =
   134
```

2.
```
% Fungsi untuk menghitung volume &
% luas permukaan balok: hitung_balok.m

function [vol,area] = hitung_balok(p,l,t)

vol = p*l*t;     % hitung volume
area = 2*(p*l + p*t + l*t);   % luas permukaan
```

Kita jalankan fungsi tersebut:

```
>> [V,L] = hitung_balok(10,5,3)
V =
   150
L =
   190
```

3.
```
% Fungsi untuk menghitung volume &
% luas permukaan prisma segi-4: hitung_prisma.m

function [vol,area] = hitung_prisma(p,l,t)

vol = 1/3*p*l*t; % hitung volume

% hitung tinggi segitiga pada sisi lebar alas
t_l = sqrt((p/2)^2 + t^2);
```

```
% hitung tinggi segitiga pada sisi panjang alas
t_p = sqrt((l/2)^2 + t^2);

% hitung luas permukaan prisma
area = p*l + p*t_p + l*t_l;
```

Kita jalankan fungsi tersebut:

```
>> [V,L] = hitung_prisma(6,4,5)
V =
    40
L =
   79.6348
```

4.
```
% Program segitiga Pascal: prog_pascal.m

clear;

x = input('Masukkan jumlah level: ');
if x < 1  % jika level negatif atau nol
   return
end

x = ceil(x); % pembulatan kalau-kalau x bukan
             % bilangan bulat
disp('     1') % tampilkan level-1
if x==1
   return
end

disp('     1     1') % tampilkan level-2
if x==2
   return
end

P=[1 1];
for i=3:x
   for j=1:i-2
          q(j) = P(j) + P(j+1);
   end
   P = [1 q 1];
   disp(P)  % tampilkan level-3 dst..
end
```

Kita coba jalankan program tersebut:

```
>> prog_pascal
Masukkan jumlah: 6
     1
     1     1
     1     2     1
     1     3     3     1
     1     4     6     4     1
     1     5    10    10     5     1
```

5.
```
% Fungsi untuk menghitung jumlah hari
% di antara dua tanggal: hitung_hari.m

function jml_hari = hitung_hari(tgli,blni,thni, ...
                    tglf,blnf,thnf)

% tgli, blni, thni : tanggal, bulan, dan tahun
%                awal, dalam angka
% tglf, blnf, thnf : tanggal, bulan, dan tahun
%                akhir, dalam angka

% jml hari dlm setiap bulan: Januari s.d. Desember
tabel_bulan = [0 31 28 31 30 31 30 31 31 30 31 ...
30 31];

if (thni<1900) | (thnf<1900)
% keluar dari program jika tahun < 1900
   disp('Tahun harus >= 1900')
   return
elseif (blni<1) | (blnf<1) | (tgli<1) | (tglf<1)
% keluar jika bulan/tanggal < 1
   disp('Bulan dan tanggal harus positif')
   return
end

if (thni>thnf) | (thni==thnf & blni>blnf) | ...
   (thni==thnf & blni==blnf & tgli>tglf)
% keluar jika awal lebih dulu daripada akhir
   disp('Masukkan: tgl,bln,thn_awal, ...
   tgl,bln,thn_akhir')
   return
end

if (tgli>tabel_bulan(blni+1)+iskabisat(thni)) | ...
   (tglf > tabel_bulan(blnf+1) + iskabisat(thnf))
   disp('Tanggal terlalu besar')
   return
end

jml_hari=0;
if thni~=thnf
   for i=thni:thnf-1
        jml_hari = jml_hari + 365 + iskabisat(i);
    end
end

for i=1:blni
   jml_hari = jml_hari - tabel_bulan(i);
   if i==3
          jml_hari = jml_hari - iskabisat(thni);
   end
end
```

```
for i=1:blnf
   jml_hari = jml_hari + tabel_bulan(i);
if i==3
          jml_hari = jml_hari + iskabisat(thnf);
   end
end

jml_hari = jml_hari + tglf - tgli;
```

Kita jalankan fungsi tersebut untuk menghitung jumlah hari antara tanggal 1 Januari 2000 – 1 Januari 2001, dan antara 2 Januari 2004 – 5 November 2006.

```
>> jml_hari = hitung_hari(1,1,2000,1,1,2001)
jml_hari =
   366
>>  jml_hari = hitung_hari(2,1,2004,5,11,2006)
jml_hari =
        1038
```

Bab 7:

1.
```
>> waktu = 6:17;
>> pend = [100 350 824 1056 1525 1247 958 1008 ...
897 921 958 215];
```

   a)
```
>> max(pend)
ans =
         1525
```

   b)
```
>> sum(pend)
ans =
        10059
```

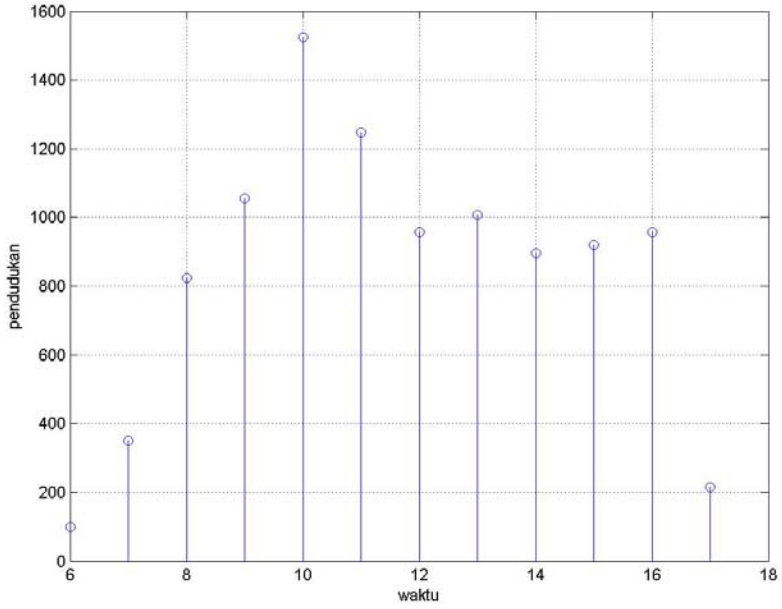
   c)
```
>> mean(pend), median(pend)
ans =
  838.2500
ans =
  939.5000
```

   d)
```
>> std(pend), var(pend)
ans =
  418.4339
ans =
  1.7509e+005
```

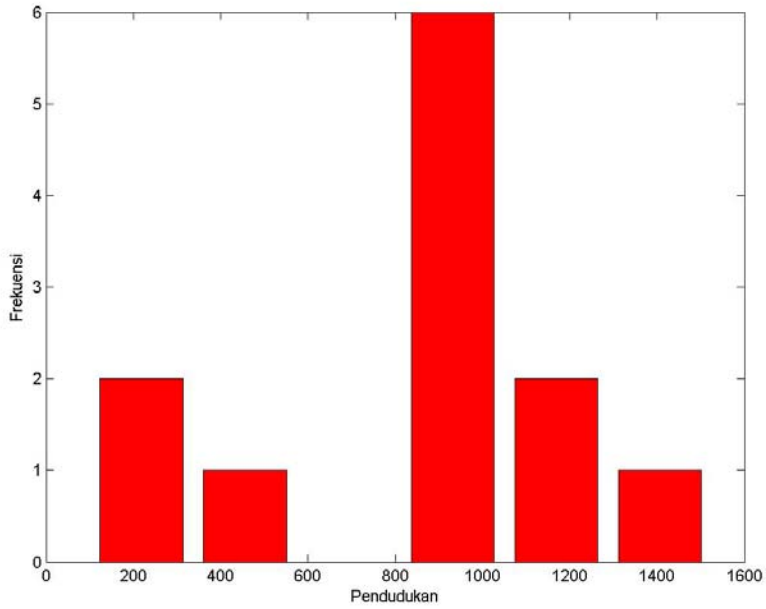
   e)
```
>> urut_asc = sort(pend)
urut_asc =
  Columns 1 through 6
  100   215   350   824   897   921
  Columns 7 through 12
  958   958  1008  1056  1247  1525
```

```
>> urut_dsc = fliplr(urut_asc)
urut_dsc =
  Columns 1 through 6
  1525  1247  1056  1008   958   958
  Columns 7 through 12
  921    897   824   350   215   100
```

f)
```
>> figure; stem(waktu,pend); grid on;
>> xlabel('waktu'), ylabel('pendudukan')
```



g)
```
>> [m,y]=hist(pend,6);
>> figure; bar(y,m,'r')
>> xlabel('Pendudukan'), ylabel('Frekuensi')
```



2.
```
>> waktu = 9:16;
>> T = [27.0 28.2 29.5 29.6 30.0 30.5 29.8 28.9;
26.8 28.1 30.3 30.6 30.0 31.0 29.6 27.5;
27.1 28.8 29.0 29.1 31.2 31.3 30.2 26.8];
```

a)
```
>> avg_harian = mean(T')
avg_harian =
   29.1875   29.2375   29.1875
```

b)
```
>> avg_11_13 = mean(T(:,3:5))
avg_11_13 =
   29.6000   29.7667   30.4000
```

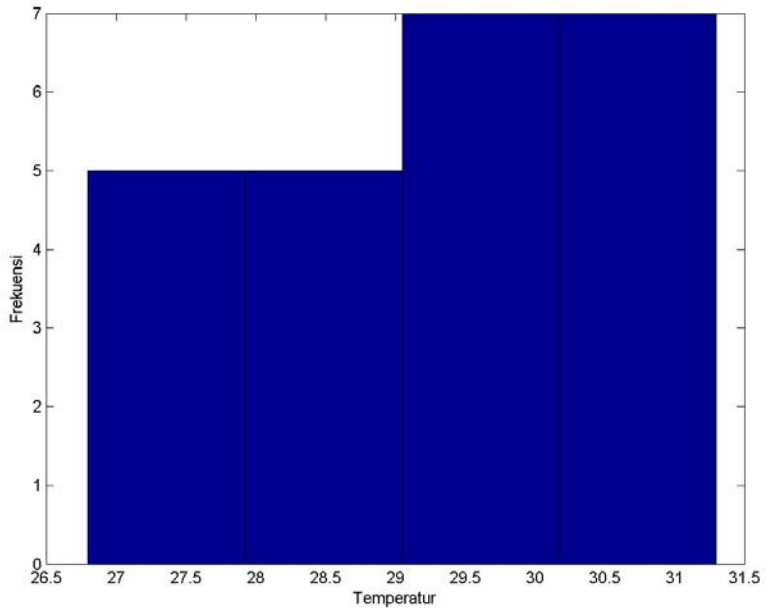
c)
```
>> avg_hari_1_2 = mean( [ T(1:2,:) ]' )
avg_hari_1_2 =
   29.1875   29.2375
```

d)
```
>> avg_3hari_9_12 = mean(mean(T(:,1:4)))
avg_3hari_9_12 =
   28.6750
```

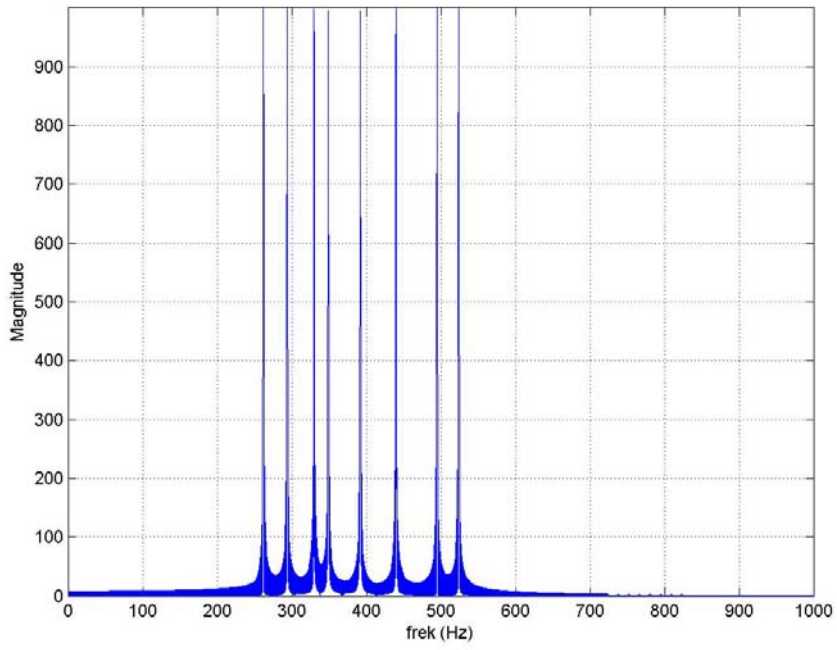
e)
```
>> max_T = max(max(T)), min_T = min(min(T))
max_T =
   31.3000
min_T =
   26.8000
```

f)
```
>> TT = reshape(T,1,24);
% bentuk ulang matriks T menjadi vektor TT
>> avg_T = mean(TT), med_T = median(TT)
avg_T =
   29.2042
med_T =
   29.5500

>> sd_T = std(TT), var_T = var(TT)
sd_T =
    1.4101
var_T =
    1.9882
```

g)
```
>> figure; hist(TT,4);
>> xlabel('Temperatur'), ylabel('Frekuensi')
```



3.
```
>> [suara,Fs] = wavread('file_suara.wav');
>> pj = length(suara);
>> S = fft(suara,pj);

>> frek = [ 0:(pj-1) ].* Fs/pj;
>> figure;
>> plot(frek,abs(S)); axis([0 1000 0 max(abs(S))])
>> grid on; xlabel('frek (Hz)'),ylabel('Magnitude')
```



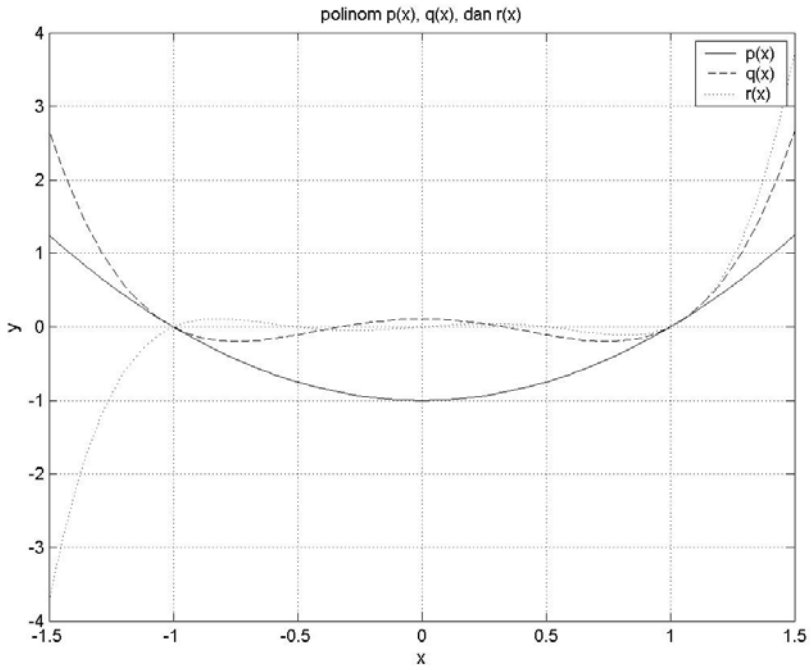
Terlihat delapan komponen frekuensi yang mewakili nada-nada satu oktaf.

Bab 8:

1.
```
>> p=[1 0 -1], q=[1 0 -10/9 0 1/9]
p =
    1    0   -1
q =
    1.0000   0    -1.1111   0    0.1111
>> r=conv([1 3/2 1/2],[1 -3/2 1/2 0])
r =
    1.0000  0   -1.2500     0    0.2500    0
```

2.
```
>> x = -1.5:0.3:1.5;
>> eval_p = polyval(p,x)
eval_p =
Columns 1 through 6
 1.2500  0.4400 -0.1900 -0.6400 -0.9100 -1.0000
Columns 7 through 11
-0.9100 -0.6400 -0.1900  0.4400  1.2500

>> eval_q = polyval(q,x)
eval_q =
Columns 1 through 6
 2.6736  0.5847 -0.1328 -0.1593  0.0192  0.1111
Columns 7 through 11
 0.0192 -0.1593 -0.1328  0.5847  2.6736

>> eval_r = polyval(r,x)
eval_r =
Columns 1 through 6
-3.7500 -0.6283  0.0958  0.0422 -0.0437  0
Columns 7 through 11
 0.0437 -0.0422 -0.0958  0.6283  3.7500
```

3.
```
>> x = linspace(-1.5,1.5,100);
>> yp=polyval(p,x); yq=polyval(q,x);
>> yr=polyval(r,x);

>> figure; plot(x,yp,'k-',x,yq,'k--',x,yr,'k:');
>> grid on, xlabel('x'), ylabel('y')
>> legend('p(x)','q(x)','r(x)')
>> title('polinom p(x), q(x), dan r(x)')
```

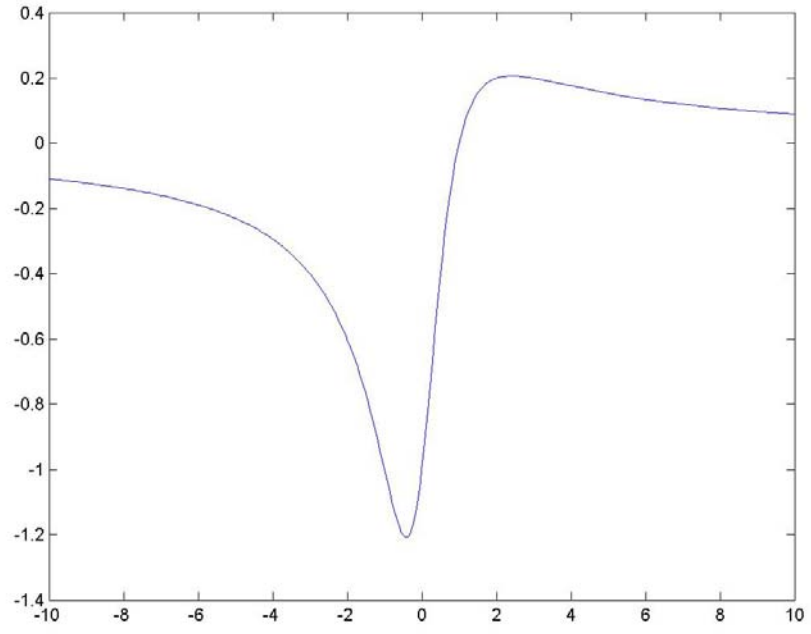


4. Kita definisikan fungsi tersebut dalam M-file:

```
function y = fungsi_Fx(x)

y = (x-1)./(x.^2+1);
```

Kita simpan dengan nama **fungsi_Fx.m**, kemudian kita plot:

```
>> figure; fplot('fungsi_Fx',[-10 10]);
```



Kita cari nol dari fungsi dengan *initial guess* $x = 0$.

```
>> nol_1 = fzero('fungsi_Fx',0)
nol_1 =
     1
```

Kita cari minimum dari fungsi pada rentang $-2 \le x \le 2$.

```
>> min_1 = fminbnd('fungsi_Fx',-2,2)
min_1 =
   -0.4142
```

Untuk mencari maksimum kita tuliskan terlebih dahulu M-file baru bernama **minus_Fx.m**

```
function y = minus_Fx(x)

y = -1.* fungsi_Fx(x);
```

Kita cari maksimum dari fungsi pada rentang $0 \le x \le 6$.

```
>> max_1 = fminbnd('minus_Fx',0,6)
max_1 =
    2.4142
```

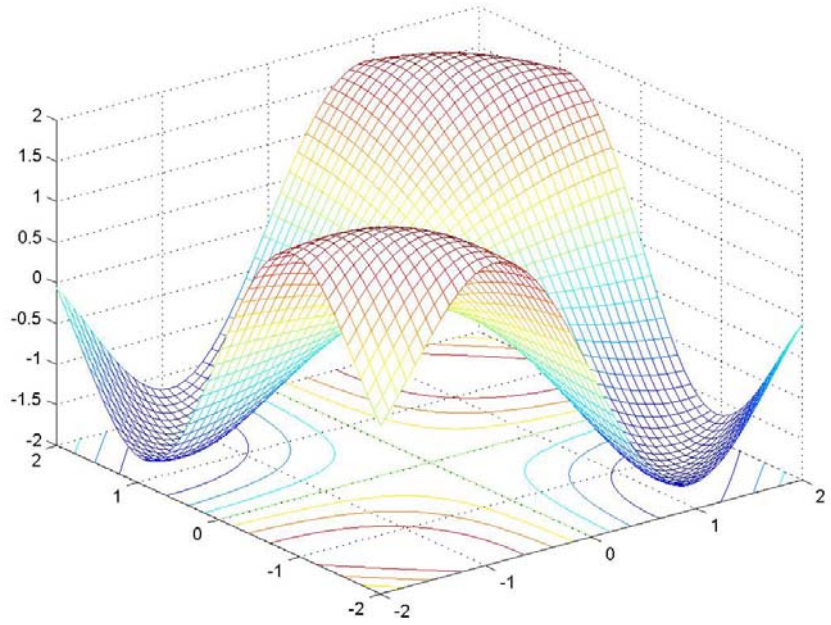
Nol fungsi ada di $x = 1$, minimum di $x = -0{,}4142$, dan maksimum di $x = 2{,}4142$.

5. Kita definisikan fungsi tersebut dalam M-file:

```
function z = fungsi_Gxy(x)

z = sin(x(1)).*sin(x(2)) + sin(x(1).*x(2));
```

Kita simpan dengan nama **fungsi_Gxy.m** dan kita plot:

```
>> x=linspace(-2,2,50); % menciptakan vektor x
>> % asumsikan y = x
>> for i = 1:50  % menghitung Gxy pada setiap titik
      for j = 1:50
         z(i,j) = fungsi_Gxy([x(i) x(j)]);
      end
   end

>> meshc(x,x,z); % plot grafik 3-D plus kontur
```



Terlihat bahwa terdapat dua minimum. Kita cari minimum dengan *initial guess* di $(x,y) = (1,-1)$ dan $(-1,1)$.

```
>> min_1 = fminsearch('fungsi_Gxy',[1,-1])
min_1 =
    1.3233   -1.3233
>> min_2 = fminsearch('fungsi_Gxy',[-1,1])
min_2 =
   -1.3233    1.3233
```

Untuk mencari maksimum terlebih dahulu kita tulis M-file baru bernama **minus_Gxy.m**

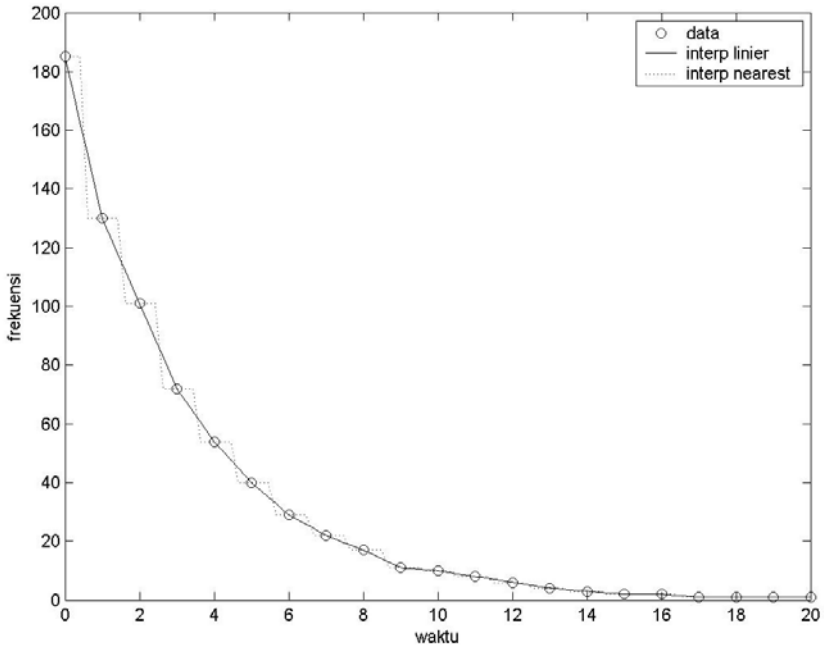
```
function z = minus_Gxy(x)

z = -fungsi_Gxy(x);
```

Terlihat bahwa terdapat pula dua maksimum. Kita akan cari dengan *initial guess* di $(x,y) = (1,1)$ dan $(-1,-1)$.

```
>> max_1 = fminsearch('minus_Gxy',[1,1])
max_1 =
    1.3233    1.3233
>> max_2 = fminsearch('minus_Gxy',[-1,-1])
max_2 =
   -1.3233   -1.3233
```

6.
```
>> waktu = 0:20;
>> frek = [185 130 101 72 54 40 29 22 17 11 10 ...
8 6 4 3 2 2 1 1 1 1];

>> t = linspace(0,20,100);
>> F1 = interp1(waktu,frek,t,'*linear');
>> F2 = interp1(waktu,frek,t,'*nearest');
```

```
>> figure;
>> plot(waktu,frek,'ko',t,F1,'k-',t,F2,'k:');
>> xlabel('waktu'), ylabel('frekuensi')
>> legend('data','interp linier','interp nearest')
```



7. Data pendudukan akan dicocokkan dengan fungsi eksponensial, sementara **polyfit** bekerja untuk fungsi polinomial; sehingga data tersebut harus dilogaritma natural terlebih dahulu:

```
>> log_frek = log(frek);
>> p_m = polyfit(waktu,log_frek,1)
p_m =
   -0.2782    5.0864
>> p_n = polyfit(waktu,log_frek,2)
p_n =
    0.0027   -0.3314    5.2549
>> p_k = polyfit(waktu,log_frek,3)
p_k =
    0.0003   -0.0075   -0.2523    5.1394
```

Ketiga polinomial di atas masing-masing mewakili:

$$m(x) = -0{,}2782x + 5{,}0864$$

$$n(x) = 0{,}0027x^2 - 0{,}3314x + 5{,}2549$$

$$k(x) = 0{,}0003x^3 - 0{,}0075x^2 - 0{,}2523x + 5{,}1394$$

Kemudian ketiga polinomial dievaluasi dan ditransformasikan kembali ke bentuk eksponensial:

```
>> kurva_m=polyval(p_m,t); kurva_M=exp(kurva_m);
>> kurva_n=polyval(p_n,t); kurva_N=exp(kurva_n);
```

```
>> kurva_k=polyval(p_k,t); kurva_K=exp(kurva_k);
```

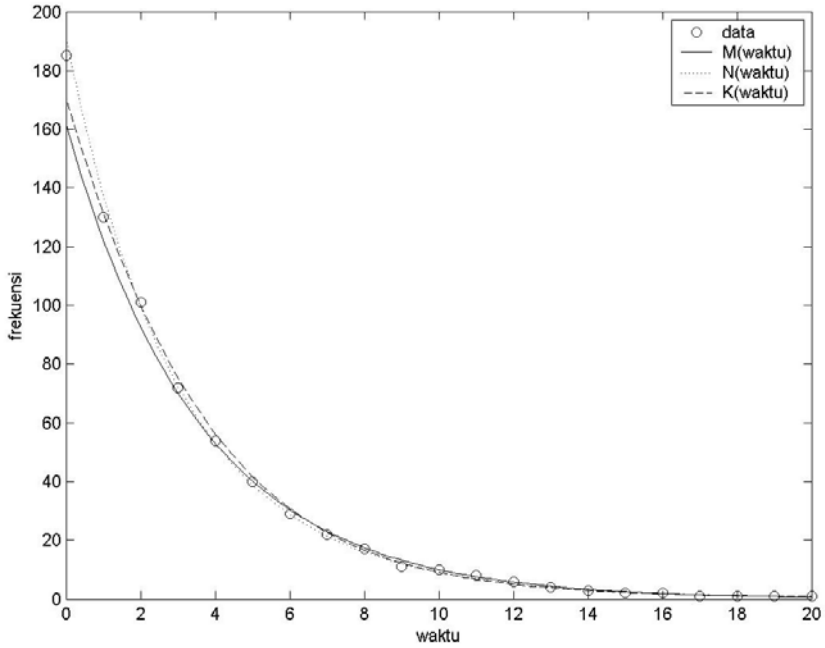
Sehingga diperoleh persamaan eksponensial:

$$M(x) = e^{m(x)} = 161{,}8\exp(-0{,}2782x)$$

$$N(x) = e^{n(x)} = 191{,}5\exp(0{,}0027x^2)\exp(-0{,}3314x)$$

$$K(x) = e^{k(x)} = 170{,}6\exp(0{,}0003x^3)\exp(-0{,}0075x^2)\exp(-0{,}2523x)$$

```
>> figure;
>> plot(waktu,frek,'ko',t,kurva_M,'k-', ...
t,kurva_N,'k:',t,kurva_K,'k--')
>> xlabel('waktu'), ylabel('frekuensi')
>> legend('data','M(waktu)','N(waktu)','K(waktu)')
```



Bab 9:

1.
```
>> x = linspace(-10,10,200);   % metode trapezoid
>> y = sqrt(100 - x.^2);
>> int_trap = trapz(x,y)
int_trap =
  157.0204
```

   Untuk metode kuadratur, kita tuliskan M-file terlebih dahulu bernama **fungsi_01.m**

```
function y = fungsi_01(x)

y = sqrt(100 - x.^2);

>> int_quad = quad('fungsi_01',-10,10)
int_quad =
  157.0796
```

2. Kita tuliskan M-file bernama **fungsi_02.m**

```
function z = fungsi_02(x,y)

z = 10 - 2.*x.^2 - y.^2;

>> int_2 = dblquad('fungsi_02',-4,4,-5,5)
int_2 =
  -720
```

3. Kita tuliskan M-file bernama **fungsi_03.m**

```
function w = fungsi_03(x,y,z)

w = x.^2 + x.*y + y.*z + z.^2;

>> int_3 = triplequad('fungsi_03',-1,1,-1,1,-1,1)
int_3 =
    5.3333
```

PROFIL PENULIS

Nama : Teguh Widiarsono
Alamat : Johar Baru IVA Gg.L no.1A, Johar Baru, Jakarta Pusat, 10560
Email : teguh98047@yahoo.com
Telepon / HP : 021 – 421 3852 / 0815 619 2813
Tempat / Tanggal Lahir : Purwokerto / 2 Desember 1980

Profil Singkat :

Penulis bernama lengkap Teguh Widiarsono, lahir di Purwokerto, 1980. Penulis menamatkan S1 (Sarjana) di ITB pada 2002, bidang keahlian Teknik Telekomunikasi, Departemen Teknik Elektro, dengan predikat Cum-Laude. Tahun 2004, penulis menyelesaikan S2 (Magister Teknik) di ITB, dengan bidang keahlian Sistem Informasi Telekomunikasi, Departemen Teknik Elektro. Selama berkuliah, penulis kerap menulis dan memberikan tutorial di bidang teknik elektro, terutama telekomunikasi. Selama berkuliah S2, penulis bekerja sebagai asisten riset di Pusat Antar-Universitas ITB. Sejak 2005, penulis bekerja sebagai *network engineer* dan tinggal di Jakarta.