Pemrograman Dasar Numerik

Nur Uddin, PhD.

Pemrograman Dasar Numerik

Nur Uddin, PhD.

Pemrograman Dasar Numerik

Penulis:

Nur Uddin, PhD

Desain Sampul:
Michel Sutedja, M.Ds, S.Sn

Universitas Pembangunan Jaya
Jl. Cendrawasih Raya B7/P, Sawah Baru,
Ciputat, Tangerang Selatan 15413
Pertama kali diterbitkan dalam bahasa Indonesia
oleh Penerbit UPJ Press pada bulan September 2019

ISBN 978-623-7455-10-3

Kata Pengantar

Puji syukur kepada Allah SWT atas segala nikmat dan karunia-Nya sehingga terbitlah buku ini. Buku ini mengenalkan dasar-dasar pemrograman yang diperlukan dalam penyelesaian persoalan secara numerik. Materi dalam buku ini mengajarkan bagaimana membangun sebuah program dari dasar secara bertahap. Materi didesain sedemikian rupa sehingga mudah dipahami oleh mahasiswa meskipun tidak memiliki latar belakang pemrograman sebelumnya. Pemrograman dalam buku ini menggunakan Matlab. Matlab merupakan suatu software cukup mudah dipelajari dan banyak digunakan dalam pemrograman numerik.

Masukan dan saran dari pembaca merupakan hal yang sangat bermanfaat untuk memperbaiki dan meningkatkan kualitas buku ini. Penulis sangat terbuka dan berterimakasih untuk saran dan masukannya.

Semoga buku ini bermanfaat bagi para pembaca, khususnya mahasiswa yang mengikuti perkuliahan numerik.

Tangerang Selatan, Agustus 2019

Nur Uddin, PhD

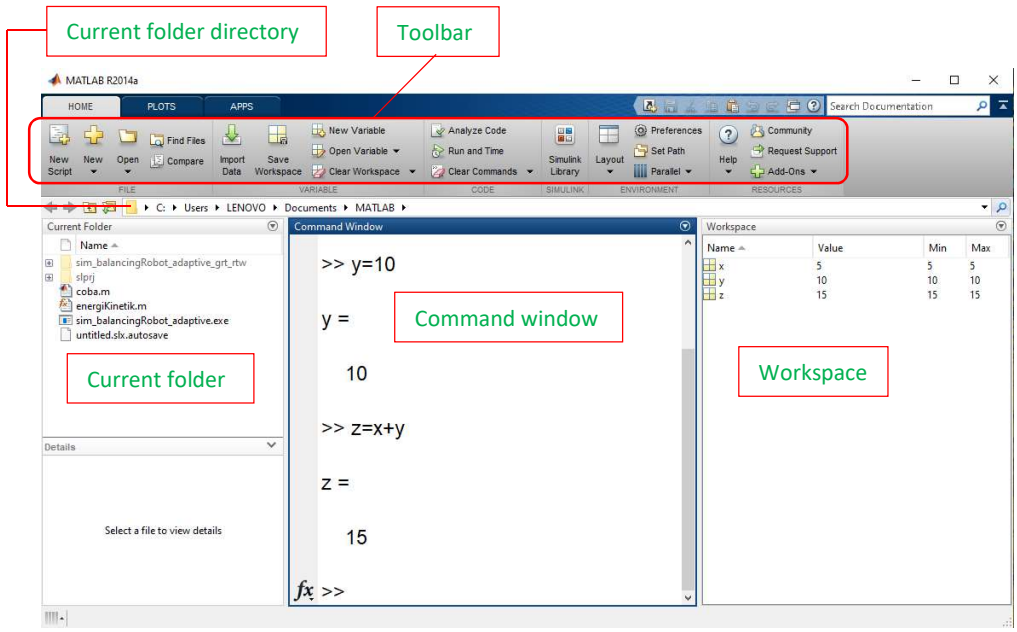
Daftar Isi

1. Pengenalan Matlab

1.1 Matlab *environment*

Tampilan Matlab ditunjukan pada Gambar 1.1. Dalam gambar tersebut ditunjukan beberapa bagian sebagai berikut:

- *Current folder directory*
- *Toolbar*
- *Current folder*
- *Command window*
- *Workspace*



Gambar 1.1

1.2 Pengoperasian Matlab seperti *calculator*

Matlab merupakan software yang cukup mudah dioperasikan bagi orang yang masih awam tentang pemrograman computer. Matlab dapat dioperasikan seperti menggunakan kalkulator sebagaimana ditunjukan dalam contoh 1.1 sebagai berikut.

Contoh 1.1:

a. Dalam *command window* ketikan perkalian 5x10, seperti berikut:

```
>> 5*10
```

b. Kemudian tekan tombol Enter, maka munculah hasil perkalian tersebut:

```
>> 5*10
ans=
     50
```

Contoh 1.2:

Menghitung $6 : 2$ sebagai berikut:

```
>> 6/2
ans=
     3
```

1.3 *Operator* dasar matematika dalam Matlab

Contoh 1.1 menunjukan operasi perkalian menggunakan Matlab. Operasi perkalian di dalam Matlab dilakukan oleh *operator* *. Tabel berikut menunjukan *operator* untuk operasi dasar matematika dan pernyataannya dalam Matlab.

Tabel 1.1: *Operator* Matematika

Operasi matematika	***Operator***		**Contoh**	
	Matematika	**Matlab**	**Matematika**	**Matlab**
Penjumlahan	+	+	2+3	2+3
Pengurangan	-	-	2-1	2-1
Perkalian	x	*	3x7	3*7
Pembagian	:	/	8:2	8/2
Pangkat (exponensial)		^	2^3	2^3

1.4 Operasi matematika: penjumlahan

Penjumlahan dua buah bilangan dalam Matlab diberikan dalam contoh berikut.

Contoh 1.3:

Operasi penjumlahan 5 + 6 = 11 dinyatakan dalam Matlab sebagai berikut:

```
>> 5+6
ans =
   11
```

1.5 Operasi matematika: pengurangan

Operasi pengurangan dalam Matlab diberikan dalam contoh berikut.

Contoh 1.4:

Operasi pengurangan 5 - 6 = -1 dinyatakan dalam Matlab sebagai barikut:

```
>> 5-6
ans =
   -1
```

1.6 Operasi matematika: perkalian

Operasi perkalian bilangan dalam Matlab diberikan dalam contoh berikut.

Contoh 1.5:

Operasi perkalian $5 \times 6 = 30$ dinyatakan dalam Matlab sebagai berikut:

```
>> 5*6
ans =
   30
```

1.7 Operasi matematika: pembagian

Operasi pembagian dalam Matlab diberikan dalam contoh berikut.

Contoh 1.6:

Operasi pembagian $12 \div 3 = 4$ dinyatakan dalam Matlab sebagai berikut:

```
>> 12/3
ans =
   4
```

Contoh 1.7:

Operasi pembagian $5 \div 2 = 2.5$ dinyatakan dalam Matlab sebagai berikut:

```
>> 5 / 2
ans =
   2.5000
```

Catatan:

Secara *default*, Matlab menuliskan angka desimal dalam empat digit.

Pembangian dengan nol akan menghasilkan nilai tak hingga atau *infinity* yang disimbolkan oleh ∞. Dalam Matlab nilai tak hingga tersebut dinyatakan oleh **`Inf`**, sebagai mana ditunjukan dalam Contoh 1.8 berikut.

Contoh 1.8:

Operasi pembagian $5 \div 0 = \infty$ dinyatakan dalam Matlab sebagai berikut:

```
>> 5/0
ans =
   Inf
```

1.8 Operasi matematika: pangkat

Operasi pemangkatan adalah suatu operasi perkalian suatu bilangan dengan bilangan itu sendiri dengan jumlah operasi perkalian sebanyak nilai pangkatnya. Dalam Matlab operasi pemangkatan dinyatakan oleh operator *hat* (^).

Contoh 1.9:

Lima pangkat empat

$$5^4 = 5 \times 5 \times 5 \times 5 = 625$$

Dengan menggunakan Matlab operasi lima pangkat empat tersebut dinyatakan sebagai berikut:

```
>> 5^4
ans =
   625
```

1.9 Operasi matematika: akar pangkat

Dari contoh diatas, diketahui $5^4 = 625$ maka 5 merupakah akar pangkat 4 dari 625. Operasi pencarian akar tersebut dalam matematika dinyatakan oleh:

$$\sqrt[4]{625} = (625)^{\frac{1}{4}} = 625^{0.25}$$

dalam Matlab operasi tersebut dinyatakan oleh:

```
>> 625^(1/4)
ans =
     5
```

atau dapat juga dinyatakan sebagai berikut:

```
>> 625^0.25
ans =
     5
```

Contoh 1.10:

Hitunglah $\sqrt[3]{27}$

Jawab:

```
>> 27^(1/3)
ans =
   3
```

1.10 Fungsi matematika

Fungsi matematika adalah suatu proses pengitungan dari suatu input untuk menghasilkan output. Setiap fungsi memiliki nama. Secara sederhana bentuk umum sebuah fungsi adalah sebagai berikut:

$$y = namaFungsi(x)$$

Dimana:

y : variabel output yang menyimpan nilai yang dihasilkan dari proses penghitungan di dalam fungsi.

$namaFungsi$: nama sebagai identitas fungsi tersebut.

x : input atau nilai yang dimasukan ke dalam fungsi.

Tanda kurung digunakan untuk memisahkan antara nama fungsi dan input.

Dalam pemrograman Matlab, fungsi dinyatakan dengan format yang sama seperti format diatas.

Contoh 1.11:

Dalam Matlab `sqrt` adalah nama fungsi untuk menghitung akar kuadrat, maka kita dapat menggunakan fungsi tersebut untuk mengitung $\sqrt{9}$ dengan *command* sebagai berikut:

```
>> y=sqrt(9)
y =
     3
```

Terkadang variabel output y tidak perlu digunakan jika hanya ingin melihat hasil dari fungsi tersebut, seperti ditunjukan oleh *command* berikut:

```
>> sqrt(9)
ans =
     3
```

1.11 Fungsi trigonometri

Fungsi trigonometri merupakan salah satu fungsi matematika yang sering dijumpai dalam studi engineering. Penghitungan fungsi trigonometri dari suatu sudut dalam matematika bisa menggunakan radian atau derajat. Namun dalam Matlab, penghitungan fungsi trigonometri berbasis pada radian bukan derajat, sehingga semua sudut harus dinyatakan dalam radian.

Contoh 1.12 :

Gunakan Matlab untuk mengitung $\sin 30^o$.

Jawab:

Penghitungan $\sin 30^o$ tersebut dilakukan dengan tahapan sebagai berikut:

- Sudut 30^o harus dikonversikan menjadi radian:

$$30\ derajat = 30 \times \frac{\pi}{180}\ radian$$

- Nilai π di dalam Matlab dinyatakan oleh "pi" sehingga *command* di Matlab untuk menghitung $\sin 30^o$ adalah:

```
>> sin(30*pi/180)
```

Penghitungan fungsi trigometri yang lain pada prinsipnya sama dengan Contoh 1.12 diatas. *Command* Matlab untuk fungsi trigonometri diberikan pada Tabel 1.2 sebagai berikut.

Tabel 1.2: Fungsi trigonometri

Fungsi trigonometri		**Contoh**	
Matematika	Matlab	Matematika	Matlab
sin	sin()	$\sin 30^o$	`sin(30*pi/180)`
cos	cos()	$\cos 60^o$	`cos(60*pi/180)`
tan	tan()	$\tan 45^o$	`tan(45*pi/180)`

1.12 Prioritas operasi matematika

Jika dalam suatu operasi matematika terdiri dari beberapa *operator* yang berbeda maka ada prioritas dalam operasi matematika. Prioritas operasi diberikan dalam Tabel 1.3. Prioritas operasi matematika ini berlaku dalam pemrogaman di Matlab.

Tabel 1.3: Prioritas operasi matematika

Prioritas	**Operasi**
1	Semua operasi di dalam tanda kurung, dari tanda kurung yang paling dalam.
2	Operasi pangkat (exponensial) dan akar pangkat, dari kiri ke kanan.
3	Operasi perkalian dan pembagian, dari kiri ke kanan.
4	Operasi penjumlahan dan pengurangan, dari kiri ke kanan.

Contoh 1.13:

1. Prioritas 1:

   Operasi matematika:

   $$3 \times \big(1 + 2 \times (1 + 2)\big) = 21$$

   Matlab *code*:

```
>> 3*(1+2*(1+2))
```

2. Prioritas 2:

   Operasi matematika:

   $$5 \times 4^{0.5} \times 3^2 = 90$$

   Matlab *code*:

```
>> 5*4^0.5*3^2
```

3. Prioritas 3:

   Operasi matematika:

   $$5 \times 4 \div 2 \times 4 = 40$$

   Matlab *code*:

```
>> 5*4/2*4
```

4. Prioritas 4:

   Operasi matematika:

   $$1 + 4 - 3 + 2 = 4$$

   Matlab *code*:

```
>> 1+4-3+2
```

1.14 Mendefinisikan variabel

Variabel adalah suatu elemen yang dideklarasikan untuk merepresentasikan suatu nilai.

Contoh 1.14:

Variabel x merepresentasikan nilai 5, y merepresentasikan nilai 4 dan z merepresentasikan nilai -0.5. Secara matematik kedua variabel tersebut didefinisikan sebagai berikut:

$$x = 5$$
$$y = 4$$
$$z = -0.5$$

Deklarasi kedua variabel tersebut dalam Matlab dinyatakan sebagai berikut:

```
>> x = 5
x =
     5

>> y = 4
y =
     4

>> z = -0.5
z =
     -0.05
```

1.15 Penggunaan tanda titik koma (;)

Dalam Matlab, tanda titik koma (;) diguanakan agar nilai yang dihasilkan dalam sebuah operasi tidak ditampilkan pada *command window*. Sebagai contoh, sebuah variabel dideklarasikan dengan *command* berikut:

```
>> a=10
```

Dengan menekan tombol ***Enter*** maka nilai hasil pendeklarasian variabel tersebut akan tampilkan pada *command window* seperti berikut:

```
>> a=10
a=
     10
```

Menampilkan nilai sebuah variabel yang didefinisiakan pada *command window* seperti contoh diatas merupakan suatu hal yang tidak perlu dilakukan, dimana variabel baru saja didefinisikan. Menampilkan nilai tersebut akan membuat *command window* menjadi penuh dan memakan memori komputer. Menampilkan nilai dalam proses iterasi akan memperlambat komputasi, sehingga dalam membuat program diusahan untuk sesedikit mungkin menampilkan nilai pada *command window*.

Agat nilai yang dihasilkan dalam proses komputasi tidak ditampilkan pada *command window*, Matlab menggunakan tanda titik koma (;) untuk mengakhiri setiap *command* atau baris program.

Sebagai contoh tanda titik koma (;) digunakan untuk mengakhir sebuah *command* yang mendefinisikan sebuah variabel, misalnya:

```
>> a=10;
```

maka jika tombol ***Enter*** ditekan akan menghasilkan tampilan seperti berikut:

```
>> a=10;
>>
```

Meskipun nilai variabel a tersebut tidak ditampilkan dalam *command window*, namun nilai variabel a tersebut sudah disimpan dalam memory komputer.

Contoh 1.15:

Penggunaan tanda ; untuk mengakhiri pendeklarasian variabel pada Contoh 1.14, yaitu

$$x = 5$$
$$y = 4$$
$$z = -0.5$$

ditunjukan sebagai berikut:

```
>> x = 5;
>> y = 4;
>> z = -0.5;
>>
```

1.16 Operasi variabel

Variabel dideklarasikan untuk merepresentasikan sebuah nilai. Operasi matematika dari nilai yang direpresentasikan tersebut dapat dilakukan dengan menggunakan variabel tersebut.

Contoh 1.16:

Dengan pendeklarasian dua variabel berikut

$x = 5$ dan $y = 4$

maka dapat dilakukan berbagai operasi matematika. Berikut diberikan empat contoh operasi matematika menggunakan variable.

- Penjumlahan: $x + y$

```
>> x=5;
>> y=4;
>> x + y
ans =
     9
```

- Pengurangan: $x - y$

```
>> x - y
ans =
     1
```

- Perkalian: $x \times y$

```
>> x*y
ans =
    20
```

Meskipun perkalian $x \times y$ sering ditulis dalam matematika sebagai xy, namun perkalian tersebut tidak dapat dideklarasikan di dalam Matlab seperti berikut:

```
>> xy
Undefined function or variabel 'xy'.
```

Deklarasi tersebut menghasilkan notifikasi *error* berupa '*Undefined function or variabel*'. Hal ini dikarenakan deklarasi diatas merupakan perintah untuk memanggil variabel atau fungsi `xy`, dimana `xy` tidak pernah didefinisikan sebelumnya.

- Akar dari penjumlahan dua variabel: $\sqrt{x+y}$

```
>> (x+y)^0.5
ans =
     3
```

1.17 Mendefinisikan variabel baru sebagai hasil operasi variabel lain

Suatu variabel baru dapat didefinisikan sebagai representasi dari hasil operasi matematik variabel yang lain.

Contoh 1.17:

Diberikan $x = 5$ dan $y = 4$.

a. Hitunglah nilai $x + y$ dan simpanlah hasil penjumlahan tersebut sebagai variabel z.
b. Hitunglah nilai $x - y$ dan simpanlah hasil pengurangan tersebut sebagai variabel w.

Jawab:

Secara matematik operasi penjumlahan dan pengurangan diatas dinyatakan oleh persamaan berikut:

$$z = x + y$$

$$w = x - y$$

dan deklarasi dalam Matlab adalah sebagai berikut:

```
>> x = 5;
>> y = 4;
>> z = x+y
z =
     9
>> w =  x-y
```

Contoh 1.18:

Diberikan $x = 5$ dan $y = 4$, dengan menggunakan Matlab hitunglah $q = \sqrt{x + y}$.

Jawab:

```
>> x = 5; y = 4;
>> q =  ( x + y )^0.5
q =
     3
```

Contoh 1.19:

Diberikan a= 3 dan b= 2, dengan menggunakan Matlab hitunglah $c = a^2 + b^2$.

Jawab:

```
>> a = 3; b = 2;
>> c =  a^2 + b^2
c =
     13
```

1.18 *Overwrite* variabel

Variabel ditugaskan untuk merepresentasikan suatu nilai. Penugasan tersebut bersifat sementara atau *temporary*. Meskipun suatu variabel telah ditugaskan untuk merepresentasikan suatu nilai, namun variabel tersebut bisa dipindah-tugaskan untuk merepresentasikan suatu nilai yang baru dan tidak lagi mereprentasikan nilai yang lama.

Penggantian tugas variabel untuk merepresentasikan nilai yang baru dan tidak lagi merepresentasikan nilai yang lama ini disebut sebagai *overwriting*.

Contoh 1.20:

Variabel u pada awalnya digunakan untuk merepresentasikan nilai 5, namun kemudian variabel u tersebut diganti untuk merepresentasikan nilai 6.

⇨ $u = 5$ diubah menjadi $u = 6$

```
>> u = 5
u =
      5
>> u=6
u =
      6
>> u
u =
      6
```

Contoh 1.21:

Variabel s digunakan untuk merepresentasikan nilai 25, namun kemudian variabel s ini digunakan untuk merepresentasikan $\sqrt{s}$.

⇨ $s = 25$ diubah menjadi $s = \sqrt{s}$

```
>> s = 25
s =
     25
>> s = s^0.5
s =
     5
>> s
s =
     5
```

Contoh 1.22:

Variabel t merepresentasikan nilai 2, namun kemudian variabel t ini digunakan untuk merepresentasikan $t + 1$.

⇨ $t = 2$ diubah menjadi $t = t + 1$

```
>> t = 2
t =
     2
>> t = t + 1
t =
     3
>> t
t =
     3
```

1.19 Menampilkan nama variabel yang telah didefinisikan (**>> who**)

Untuk mengetahui variabel yang telah didefinisikan dalam *command window*, digunakanlah *command* `who`.

Contoh 1.23:

Misal kita sudah mendefinisikan variabel a, b, c dan d dalam *command window*, maka nama-nama variabel yang telah didefinisikan tersebut bisa cek dengan *command who* dan jika tombol Enter ditekan akan muncul nama-nama variabel tersebut seperti berikut:

```
>> a = 2
a =
     2

>> b = 3
b =
     3

>> c = 5;

>> d = a+c
d =
    7
 >> who
Your variabels are:
a  b  c  d
```

1.20 Membersihkan *command window* (**>> clc**)

Diatas kita telah mendefinisikan beberapa variabel, melakukan operasi matematika, dan menjalankan fungsi di dalam *command window*, tentunya *command window* menjadi penuh. Untuk membersihkan *command window* digunakan *command* `clc`.

```
>> clc
```

Command `clc` hanyalah menghapus tulisan pada tampilan *command window* saja, namun variabel yang tersimpan di dalam *command window* masih tetap tersimpan.

1.21 Menghapus variabel (**>> clear nama_variabel**)

Untuk menghapus variabel yang tersimpan dalam *command window*, digunakan *command clear*.

Contoh 1.24:

Melanjutkan Contoh 1.23, variabel a, b, c dan d telah tersimpan dalam *command window*.

- Gunakan *command* `who` untuk melihat variabel yang tersimpan.

```
>> who
Your variabels are:
a  b  c  d
```

- Hapuslah variabel a dengan *command* `clear a` seperti berikut:

```
>> clear a
```

- Sekarang cek variabel yang tersimpan dengan *command* `who` sebagai berikut:

```
>> who
Your variabels are:
b  c d
```

1.22 Menghapus semua variabel yang tersimpan (>> **clear all**)

Untuk menghapus semua variabel yang tersimpan dalam *command window*, gunakan *command* `clear all`.

```
>> clear all
```

Contoh 1.25:

Mengadopsi Contoh 1.19 dimana didefinisikan empat buah variable a, b, c, dan d dalam *command window*. Keempat variable yang telah didefinisikan tersebut dapat dihapus semua dengan menggunakan perintah `clear all` sebagai berikut.

```
>> a = 2;
>> b = 3;
>> c = 5;
>> d = a+c;
>> who
Your variabels are:
a  b  c  d

>> clear all
```

Sekarang cek variabel yan tersimpan dalam *command window* dengan *command* `who` seperti berikut, maka tidak akan muncul nama variabel yang tersimpan karena sudah tidak ada lagi variabel yang tersimpan dalam *command window*.

```
>> who
>>
```

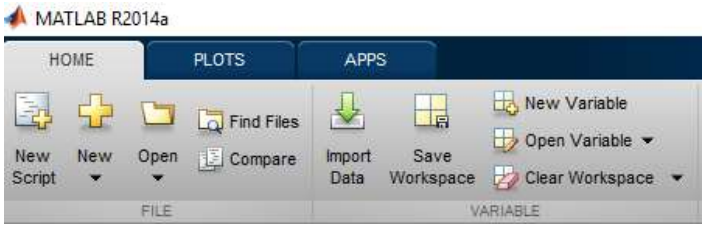
2. Pemrograman M-*file* dan *Function*

2.1 Pengenalan M-*file*

Pada pembahasan sebelumnya di Bab 1 telah didemonstrasikan beberapa contoh operasi Matlab, diantaranya: operasi aritmatika, trigonometri, pendefinisian variabel, dan penghapusan variabel. Sejauh ini operasi Matlab tersebut dideklarasikan secara langsung pada *command window*.

Command window merupakan tempat dimana semua operasi Matlab dijalankan. Namun apa yang dituliskan pada *command window* tersebut tidak tersimpan sehingga kita harus menuliskan lagi setiap kali melakukan operasi Matlab. Untuk sebuah program yang hanya terdiri dari satu atau dua baris tentunya hal tersebut tidak menjadi masalah. Namun dalam kenyataannya, sebuah program dapat terdiri dari puluhan, ratusan, bahkan ribuan baris. Hal tentunya akan menjadi masalah jika program yang telah dibuat dan terdiri dari banyak baris tidak dapat disimpan. Matlab memberikan solusi untuk masalah tersebut dengan menyediakan fasilitas berupa *text editor.*

Text editor ini digunakan untuk menuliskan suatu program Matlab. *Text editor* ini dapat dibuka dengan menekan ikon ***new script*** seperti ditunjukan pada Gambar 2.1. Progam yang ditulis dalam *text editor* ini akan disimpan dalam suatu *file* dengan ekstensi *m* (bentuk umum nama *file*-nya adalah ***nama_file.m***). *File* tersebut dikenal sebagai m-*file*. Program yang dibuat dan disimpan dalam bentuk m-*file* tersebut dapat dijalankan secara langsung di *command window* dengan mengetikan nama *file*nya (***nama_file***) seperti diilustrasikan dalam Gambar 2.2.



Gambar 2.1: Matlab *menu bar*

```
>> nama_file
```

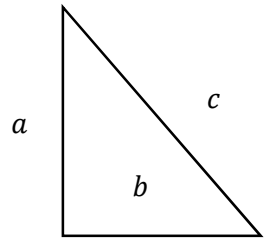
Gambar 2.2: Menjalankan program yang disimpan dalam bentuk m-file di command window Matlab.

2.2 Pembuatan program m-*file* sederhana

Berikut diberikan contoh pembuatan progam m-*file* yang sederhana.

Contoh 2.1:

Membuat program m-*file* untuk menghitung panjang sisi miring dari suatu segita siku-siku dengan rumus Pythagoras.



$$c^2 = a^2 + b^2$$

Misal diketahui panjang $a = 4$ dan $b = 3$, tentukan panjang c!

Jawab:

Penghitungan panjang *c* dapat diselesaikan dengan sebuah program m-*file* dengan langkah-langkah sebagai berikut:

i. Buka m-*file editor* dan tuliskan *code* berikut:

```
a = 4;
b = 3;
c = (a^2+b^2)^0.5
```

ii. Simpan *code* tersebut dengan nama file ***pytha1.m***

iii. Jalankan program tersebut dengan mengetikan nama *file* di *command window* dan tekan enter. Pastikan Matlab bekerja pada folder dimana m-*file* tersebut disimpan. M-*file* tersebut juga dapat dijalankan dengan menekan tombol Run di menu *Editor*.

```
>> pytha1
c =
      5
```

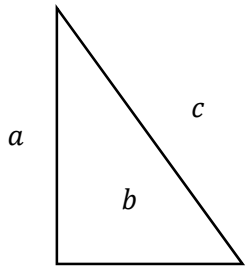
2.3 Pembuatan program m-*file* sederhana yang memiliki input

Dalam Contoh 2.1, program m-*file* hanya memberikan output penghitungan berdasarkan nilai variabel yang didefinisikan di dalam program m-*file*. Bila program ***pytha1.m*** tersebut digunakan untuk menghitung segitiga yang berbeda, maka pajang sisi segitiga tersebut

harus didefinisikan di dalam program m-*file*. Hal ini membuat program tersebut tidak fleksibel. Akan lebih praktis jika data segitiga dapat dimasukan tanpa harus mendefinisikannya di m-*file*. Matlab menyediakan fasilitas tersebut dengan *command* i**nput** sebagai suatu cara agar program berinteraksi dengan pengguna untuk memasukan data.

Contoh 2.2:

Membuat program m-*file* untuk menghitung panjang sisi miring dari suatu segita siku-siku dengan rumus Pythagoras, dimana nilai a dan b dinyatakan sebagai input.



$$c^2 = a^2 + b^2$$

i. Buka m-*file editor* dan tuliskan *code* berikut:

```
a=input('Masukan panjang a: ');
b=input('Masukan panjang b: ');
c=(a^2+b^2)^0.5
```

ii. Simpan *code* tersebut, misal dengan nama ***pytha2.m***

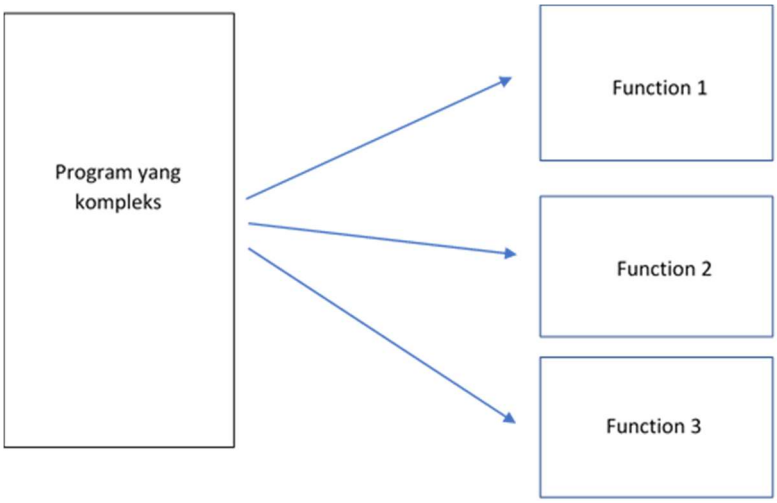
iii. Jalankan program tersebut dengan mengetikan nama *file* di *command window* dan tekan enter. Masukan panjang a dan panjang b. Tekan enter untuk mendapakan nilai c.

```
>> pytha2
Masukan panjang a: 6
Masukan panjang b: 8

c =
    10
```

2.4 *Function*

Pada pembahasan sebelumnya diberikan dua contoh program, yaitu: ***pytha1.m*** dan ***phyta2.m*.** Kedua program tersebut merupakan program sederhana dimana didalam program tersebut hanya terdapat satu proses perhitungan. Program sederhana hanya digunakan dalam latihan ketika memulai belajar membuat program. Dalam aplikasi nyata, jarang sekali ditemui program yang hanya terdiri dari satu proses perhitungan aja, sebuah program biasanya terdiri dari banyak proses perhitungan.

Sebuah program yang kompleks terdiri dari banyak proses perhitungan dan ditulis dalam banyak baris program. Banyaknya baris program akan menyulitkan dalam pembuatan program, tracing *error*, *maintenance*, dan modifikasi. Untuk mengatasi hal tersebut, sebuah program yang kompeks dipecah menjadi beberapa proses komputasi yg lebih sederhana yang disebut ***function***. Gambar 2.3 memberikan ilustrasi sebuah program yang kompleks dan dipartisi kedalam tiga buah *function* yang lebih sederhana.



Gambar 2.3: Ilustrasi sebuah program m-*file* yang kompleks dipartisi menjadi tiga *function*.

Function merupakan sebuah modul komputasi. Sebuah *function* terdiri dari tiga elemen dasar: input, proses, dan output. Input adalah data yang diberikan kepada *function* untuk diolah melalui sebuah proses komputasi. Proses komputasi tersebut menghasilkan suatu data yang disebut sebagai output dari *function* tersebut. *Function* dalam Matlab dibuat di dalam Matlab *Editor* dengan format sebagai berikut.

```
function variabel_output = namaFunction(variabel_input)
```

Catatan:

Function yang dibuat harus disimpan dengan nama yang sama dengan namaFunction.

2.4.1 Function dengan satu input satu output:

```
function y=namaFunction(x)
```

Contoh 2.3:

Membuat *function* untuk menghitung sinus suatu sudut dalam derajat.

i. Buka m-*file editor* baru (Control+N) dan tuliskan *code* berikut:

```
function y = sinDerajat(x)
a=x*pi/180; % konversi derajat ke radian
y=sin(a);
```

ii. Simpan m-*file* dengan nama **sinDerajat.m**

iii. Jalankan *function* tersebut di *command window*. Pemanggilan *function* harus sama persis dengan nama *function* termasuk huruf besar dan huruf kecil.

```
>> sinDerajat(30)
ans =
    0.5000
```

2.4.2 Function dengan banyak input satu output:

```
function y=namaFunction(x1,x2,…,xn)
```

Contoh 2.4:

Membuat *function* untuk menghitung energy kinetic sebuah benda yang bermasa m bergerak dengan kecepatan v.

$$E_k = \frac{1}{2}mv^2$$

i. Buka m-*file editor* baru dan ketikan *code* berikut:

```
function Ek = energiKinetik(m,v)
Ek = 0.5*m*v^2;
```

ii. Simpan dengan nama **energiKinetik.m**

iii. Jalankan *function* tersebut di *command window*, misal untuk menghitung energi kinetic dari benda bermasa 5 kg dan bergerak dengan kecepatan 4 m/s.

```
>> energiKinetik(5,4)
ans =
    40
```

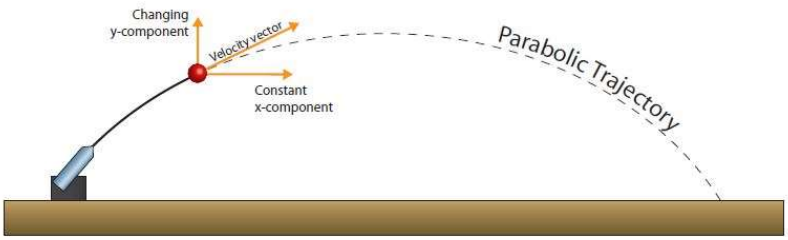
2.4.3 Function dengan satu input banyak output:

```
function [y1,y2,…,yn] =namaFunction(x)
```

Contoh 2.5:

Sebuah meriam terpasang dengan sudut tetap 30 derajat sebagaimana diilustrasikan dalam Gambar 2.4. Sebuah peluru ditembakan dari meriam tersebut dengan kecepatan awal v_0 m/s. Jika percepatan gravitasi 9.8 m/s^2, tentukan:

a. Jarak horizontal terjauh dari meriam tersebut.
b. Tentukan tinggi peluru maksimal
c. Lama peluru terbang



Gambar 2.4: Gerak parabolik sebuah peluru.

Jawab:

Rumus untuk menghidung jarak horizontal terjauh, tinggi maksimum, dan lama peluru terbang diberikan oleh rumus berikut:

- Jarah horizontal terjauh : $x_{max} = \frac{v_0^2 sin2\theta}{g}$

- Tinggi maksimum : $y_{max} = \frac{v_0^2 \sin^2 \theta}{2g}$
- Lama peluru terbang : $t_f = \frac{2v_0 \, sin\theta}{g}$

Sebuah Matlab *function* dapat dibangun untuk perhitungan tersebut berdasarkan rumus-rumus diatas dengan langkah-langkah sebagai berikut:

i. Buka m-*file editor* baru dan ketikan *code* berikut:

```
function [xm, ym, tf] = fixedParabolic(v0)
g=9.8;
theta=30*pi/180;
xm=v0^2*sin(2*theta);
ym=v0^2*(sin(theta))^2/(2*g);
tf=2*v0*sin(theta)/g;
```

ii. Simpan dengan nama **fixedParabolic.m**

iii. Jalankan *function* tersebut di *command window* dengan memasukan nilai kecepatan awal peluru 30 m/s sebagai berikut:

```
>>  [xm,ym,tf] = fixedParabolic(30)
xm =
  779.4229
ym =
   11.4796
tf =
    3.0612
```

2.4.4 *Function* dengan banyak input banyak output:

function [y1,y2,…,yn] =namaFunction(x1,x2,…,xn)

Contoh 2.6:

Sebuah meriam terpasang dengan sudut θ dalam derajat yang nilainya bervariasi (dapat ubah-ubah). Sebuah peluru ditembakan dari meriam tersebut dengan kecepatan awal v_0 m/s. Jika percepatan gravitasi 9.8 m/s^2, buatlah sebuah *function* untuk menghitung ketiga data berikut:

a. Jarak horizontal terjauh dari meriam tersebut.
b. Tentukan tinggi peluru maksimal
c. Lama peluru terbang

Jawab:

i. Rumus penghitungan ketiga data tersebut diberikan sebagai beriku:

Jarah horizontal terjauh : $x_{max} = \frac{v_0^2 sin2\theta}{g}$

Tinggi maksimum : $y_{max} = \frac{v_0^2 \sin^2 \theta}{2g}$

Lama peluru terbang : $t_f = \frac{2v_0 \, sin\theta}{g}$

ii. Membuat Matlab *function*:
Buka m-*file editor* baru dan ketikan *code program* berikut:

```
function [xm, ym, tf] = vPrbl (v0,theta)
g=9.8;
thetaR=theta*pi/180;
xm=v0^2*sin(2*thetaR);
ym=v0^2*(sin(thetaR))^2/(2*g);
tf=2*v0*sin(thetaR)/g;
```

Simpan *code program* tersebut dengan nama **vPrbl.m**

iii. Mencoba menggunanan *function* yang dibuat untuk penghitungan untuk dua kasus berikut:

- Sudut meriam $\theta = 45$ derajat, kecepatan awal peluru $v_0 = 50$ m/s

```
>> [xmax,ymax,tf]=vPrbl(50,45)
xmax =
        2500
ymax =
   63.7755
tf =
    7.2154
```

- Sudut meriam $\theta = 60$ derajat, kecepatan awal peluru $v_0 = 40$ m/s

```
>> [xm,ym,tf]=vPrbl (60,40)
```

Bab 3

Vektor dan Operasi Vektor

Salah satu objek yang sangat penting dalam analisa numerik adalah matriks. Matrik adalah susunan angka atau variabel dengan ukuran tertentu (m x n). Vektor merupakan bentuk special dari matrik, dimana vektor adalah matriks yang berukuran m x 1 atau 1 x n. Vektor yang berukuran m x 1 disebut sebagai *column vector* dan vektor yang berukuran 1 x n disebut *row vector*.

3.1 Vektor

Dalam fisika, vektor didefinisikan sebagai sesuatu yang mempunyai besar dan arah. Dalam matematika, vektor didefinisikan sebagai sesuatu yang dapat dijumlahkan atau dikalikan dengan suatu skalar. Vektor dapat memiliki satu, dua, hingga *n*. Vektor biasanya dinyatakan sebagai susunan angka atau variabel di dalam tanda kurung kotak. Terdapat dua jenis vektor: *column vector* dan *row vector*.

a. *Column vector*

Column vector adalah vektor yang hanya memiliki satu *column*. Berikut contoh *column vector* dan ekspresi dalam Matlab.

Contoh 3.1:

Berikut diberikan contoh beberapa *column vector* dan pendeklarasiannya dalam Matlab

Column vector	Contoh	Expresi di Matlab*
Dimensi satu	$[3]$	`>> [3]`
Dimensi dua	$\begin{bmatrix}4\\5\end{bmatrix}$	`>>  [4;5]`
Dimensi tiga	$a=\begin{bmatrix}2\\7\\8\end{bmatrix}$	`>> a= [2;7;8]`

** elemen dalam column vector dipisahkan dengan tanda titik koma (;) untuk ganti baris*

b. *Row vector*

Row vector merupakan vektor yang hanya terdiri dari satu baris. Berikut diberikan contoh *row vector* dan ekspresinya dalam Matlab.

Contoh 3.2:

Row vector dan ekspresi dalam Matlab

Row vector	Contoh	Expresi di Matlab*
Dimensi satu	$[3]$	`>> [3]`
Dimensi dua	$[3 \quad 4]$	`>> [3 4]`
Dimensi tiga	$b=[2 \quad 7 \quad 8]$	`>> b=[2,7,8]`

** elemen dalam row vector dipisahkan oleh spasi atau tanda koma (,).*

3.2 Ukuran vektor

Ukuran vektor sangat diperlukan dalam operasi vektor, sebagai contoh operasi penjumlahan dan pengurangan dua buah vektor hanya dapat dilakukan jika kedua vektor memiliki ukuran yang sama. Ukuran vektor dapat diperoleh dengan menggunakan Matlab melalui *command*: ***size()***

Contoh 3.3:

Tentukan ukuran vektor berikut:

a) $v = [3 \quad 1 \quad 0]$ b) $u = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$ c) $w = 0$

Jawab:

i. Ukuran vektor v dapat diperoleh dengan *code* berikut:

```
>> v=[3 1 0]
v =
     3     1     0
>> size(v)
ans =
     1     3
```

Hasil tersebut menunjukan bahwa vektor v memiliki ukuran 1×3 yang artinya vektor v terdiri dari 1 baris dan 3 kolom.

ii. Ukuran vektor u dapat diperoleh dengan *code* berikut:

```
>> u=[5;3;4]
u =
     5
     3
     4
>> size(u)
ans =
     3     1
```

Hasil tersebut menunjukan bahwa vektor u memiliki ukuran 3×1 yang artinya vektor u terdiri dari 3 baris dan 1 kolom.

iii. Ukuran vektor w dapat diperoleh dengan *code* berikut:

```
>> w=0
w =
   0
>> size(w)
ans =
     1     1
```

Variabel w merupakan suatu skalar. Hasil penghitungan menggunakan Matlab menunjukan bahwa vektor w memiliki ukuran 1×1 yang artinya vektor w terdiri dari 1 baris dan 1 kolom. Dengan demikian skalar merupakan suatu vektor yang memiliki ukuran 1×1.

3.3 Transpose

Transpose adalah sebuah operasi untuk merubah bentuk sebuah coloumn vektor menjadi *row vector* atau sebaliknya. *Transpose* dalam matematik dinyatakan dengan menuliskan superscript T dari suatu vektor.

Di dalam Matlab, *transpose* dinyatakan dengan tanda petik satu (‘) dari suatu vektor. Misal *transpose* dari vektor a dinyatakan sebagai a^T dan Matlab dinyatakan sebagai >> a’

Contoh 3.4:

Vektor dan *transpose*	**Expresi di Matlab**
$[3]^T = [3]$	`>> 3'`
$[3 \quad 4]^T = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$	`>> [3 4]'`

$b = [2 \quad 7 \quad 8]$
$b^T = \begin{bmatrix} 2 \\ 7 \\ 8 \end{bmatrix}$
	`>> b = [2 7 8]`
	`b =`
	`2 7 8`
	`>> b'`
	`ans =`
	`2`
	`7`
	`8`

3.4 Operasi Dasar Vektor

Operasi dasar vektor yang dibahas dalam section ini meliputi penjumlahan dan pengurangan vektor, perkalian vektor dengan skalar, dan perkalian vektor dengan vektor.

3.4.1 Penjumlahan dan pengurangan dua buah vektor

Penjumlahan atau pengurangan vektor hanya dapat dilakukan jika ukuran dari vektor tersebut sama.

Contoh 3.5:

a. Definisikan tiga vektor berikut:

```
>> q=[3 5 5];
>> r=[6 3 7];
>> s=[3; 5; 6];
```

b. Lakukan operasi penjumlahan berikut:

```
>> q+r
ans =
     9     8    12
>> q-r
ans =
    -3     2    -2
>> q+s
Error using  +
Matrix dimensions must agree.
```

Investigasi apa yang menyebabkan q+s menghasilkan *error* !

3.4.2 Perkalian vektor dengan skalar

Vektor dapat dikalikan dengan skalar.

Contoh 3.6:

Perkalian vektor dengan skalar	Expresi di Matlab
$5 \times [3] \quad = [15]$	`>>5* [3]`
$5 \times [3 \quad 4] \quad = [15 \quad 20]$	`>> 5* [3 4]`

$b = [2 \quad 7 \quad 8]$
$b^T = \begin{bmatrix} 2 \\ 7 \\ 8 \end{bmatrix}$
$3b^T = \begin{bmatrix} 6 \\ 21 \\ 24 \end{bmatrix}$
	`>> b=[2 7 8]`
	`>>3*b'`
	`ans =`
	`6`
	`21`
	`24`

3.4.3 Perkalian vektor

Perkalian vektor mengikuti aturan matriks dimana:

a. Perkalian dua buah: vektor a berukuran mx1 dan vektor b berukuran 1xn akan menghasilkan vektor c yang berukuran mxn.

$$[a]_{m\times 1}[b]_{1\times n} = [c]_{m\times n}$$

Contoh 3.7:

Perkalian vektor dengan vektor	Expresi di Matlab

Diberikan dua buah vektor:
$a = \begin{bmatrix}1\\2\\3\end{bmatrix},\ b = [4 \quad 5]$
 	
Hitunglah:
$c = ab$
	`>> a=[1;2;3]`
	`a =`
	`1`
	`2`
	`3`
	
	`>> b=[4,5]`
	`b =`
	`4 5`
	
	`>> c=a*b`
	`c =`
	`4 5`
	`8 10`
	`12 15`
	
	`>> size(c)`
	`ans =`
	`3 2`

Vektor a berukuran 3×1 dan vektor b berukuran 1×2, perkalian vektor a dan b menghasilkan vektor dengan ukuran 3×2.

b. Perkalian vektor a berukuran mx1 dengan vektor a *transpose* menghasilkan matrik bujursangkar d yang berukuran mxm.

$$[a]_{m\times 1}[a^T]_{1\times m} = [c]_{m\times m}$$

Contoh 3.8:

Perkalian vektor dengan vektor	**Expresi di Matlab**

Diberikan vektor:
$a = \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$
Hitunglah:
$c = aa^T$
	`>> a=[3;5;1]`
	`a =`
	`3`
	`5`
	`1`
	`>> size(a)`
	`ans =`
	`2 1`
	`>> c=a*a'`
	`c =`
	`9 15 3`
	`15 25 5`
	`3 5 1`
	`>> size(c)`
	`ans =`
	`3 3`

c. Perkalian vektor b yang berukuran 1xn dengan vektor b *transpose* menghasilkan skalar e (vektor berukuran 1x1).

$$[b]_{1\times n}[b^T]_{n\times 1} = [d]_{1\times 1}$$

Contoh 3.9:

Perkalian vektor dengan vektor	Expresi di Matlab

Diberikan vektor:
$b = [4 \quad 5 \quad 1]$
 	
Hitunglah:
$d = bb^T$
	`>> b=[4 5 1]`
	`b =`
	`4 5 1`
	
	`>> size(b)`
	`ans =`
	`1 3`
	
	`>> d=b*b'`
	`d =`
	`42`
	
	`>> size(d)`
	`ans =`
	`1 1`

Contoh 3.10:

Perkalian vektor dengan vektor	Expresi di Matlab

Diberikan vektor:
$a = \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$
	`>> a=[3;5;1]`
	`a =`
	`3`
	`5`
	`1`
	
	`>> size(a)`
	`ans =`
	`3 1`

Hitunglah:
$c = a^T a$
	`>> c=a'*a`
	`c =`
	`35`
	
	`>> size(c)`
	`ans =`
	`1 1`

Hitunglah: $d = aa^T$	`>> d=a*a'`

Hasil perhitungan pada Contoh 3.10 tersebut menunjukan bahwa dalam operasi vector:

$$a^T a \neq aa^T.$$

3.5 Operasi vektor secara *element by element*

Operasi vektor secara *element by element* adalah melakukan operasi pada *element* vektor. Operasi vektor secara *element by element* ditandai dengan adanya tanda titik (.) sebelum *operator*.

3.5.1 Operasi ekponensial vektor *element by element*

Operasi vektor *element by element* juga dapat dilakukan untuk operasi ekponensial dengan symbol (.^)

Contoh 3.11:

Operasi pembagian vektor *element by element*	**Expresi di Matlab**

Diberikan vektor:
$b = [4 \quad 9 \quad 10]$
Hitunglah nilai kuadrat setiap *element* vektor b tersebut.
	`>> b = [4  9 10];`
	`>> d = b.^2`
	`d =`
	`16  81  100`

Diberikan vektor:
$c = [25 \quad 36 \quad 64]$
Hitunglah nilai akar kuadrat setiap *element* vektor c tersebut.
	`>> c = [25  36  64];`
	`>> c.^0.5`
	`ans =`
	`5  6  8`

3.5.2 Operasi perkalian vektor *element by element*

Perkalian vektor *element by element* adalah mengkalikan *element* yang sama dari dua buah vektor. Misal c adalah hasil perkalian *element by element* dari vektor a dan b, perkalian ini dinyatakan sebagai berikut:

$$[c_{i1}] = [a_{i1} \times b_{i1}] \text{ ,}$$

dimana a dan b merupakan *column vector.*

Perkalian *element by element* tersebut juga dapat dinyatakan oleh:

$$[c_{1i}] = [a_{1i} \times b_{1i}]$$

dimana a dan b merupakan *row vector*.

Perkalian vektor *element by element* dalam Matlab dinyatakan dengan symbol (.*).

Contoh 3.12:

Operasi perkalian vektor *element by element*	**Expresi di Matlab**

Diberikan vektor:
$b = [4 \quad 5 \quad 1]$
$c = [2 \quad 3 \quad 2]$
 	
Hitunglah hasil perkalian *element by element* vektor b dan c.
$d = b .\times c$
	`>> b = [4 5 1];`
	`>> c = [2 3 2];`
	`>> d = b.*c`
	`d =`
	`8 15 2`

3.5.3 Operasi pembagian vektor *element by element*

Pembagian vektor *element by element* dinyatakan dengan symbol (./) dan prinsip operasinya sama dengan perkalian vektor *element by element*.

Contoh 3.13:

Operasi pembagian vektor *element by element*	**Expresi di Matlab**

Diberikan vektor:
$b = [4 \quad 9 \quad 10]$
$c = [2 \quad 3 \quad 5]$
 	
Hitunglah hasil pembagian *element by element* vektor b oleh c.
$d = b./c$
	`>> b = [4 9 10];`
	`>> c = [2 3 5];`
	`>> d = b./ c`
	`d =`
	`1 3 2`

Diberikan vektor:
$e = [10 \quad 30 \quad 45]$
$f = [0.2 \quad -6 \quad 9]$
 	
Hitunglah hasil pembagian *element by element* vektor e oleh f.
$g = e./f$
	`>> e = [10 30 45];`
	`>> f = [0.2 -6 9];`
	`>> g = e./ f`
	`g =`
	`50 -5 5`

3.5.4 Operasi vektor secara *element by element* pada suatu fungsi

Vektor dapat dijadikan input bagi suatu fungsi. *Function* dalam Matlab melakukan operasi komputasi secara *element by element*.

Contoh 3.14:

```
>> x = [100 ; 4 ; 81]
x =
   100
     4
    81

>> y = sqrt(x)
y =
    10
     2
     9
```

Contoh 3.15:

```
>> theta=[0 30 60 90];
theta =
     0    30    60    90

>> sin(theta*pi/180)
ans =
       0    0.5000    0.8660    1.0000
```

4. Matriks dan Operasi Matriks

Matrik merupakan suatu variabel yang memiliki elemen satu atau lebih. Matrik memiliki struktur elemen dengan ukuran tertentu (m x n), dimana m menyatakan banyaknya baris dan n menyatakan banyaknya kolom. Vektor merupakan bentuk special dari matrik dimana baris atau kolomnya berjumlah satu.

Salah satu keunggalan Matlab dari bahasa pemrograman yang lain adalah Matlab memberikan kemudahan dalam melakuka operasi matriks. Matriks dideklarasikan dengan sangat sederhana didalam pemrograman menggunakan Matlab, begitu juga dengan operasi matriks.

4.1 Deklarasi matriks

Cara mendeklarasikan matriks dilakukan secara mudah baik dalam *command window* atau m-*file*. Setiap elemen matriks dalam satu baris dipisahkan oleh spasi atau tanda koma. Tanda titik koma (;) digunakan untuk memisahkan baris dalam matriks.

Contoh 4.1:

Mendeklarasikan matriks $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \end{bmatrix}$ dalam Matlab.

```
>> A=[1 2 3;3 4 7]
A =
     1     2     3
     3     4     7
```

4.2 Ukuran matriks

Ukuran matriks dapat diketahui dengan perintah "*size*".

Contoh 4.2:

Tentukan ukuran matriks $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \end{bmatrix}$ dan $B = \begin{bmatrix} 2 & 3 \\ 1 & 3 \\ 2 & 5 \end{bmatrix}$.

Jawab:

```
>> A=[1 2 3;3 4 7]
A =
     1     2     3
     3     4     7

>> size(A)
ans =
     2     3

>> B = [2 3; 1 3; 2 5]
B =
   2    3
   1    3
   2    5

>> size(B)
ans =
     3     2
```

Matriks A berukuran 2×3 yang berarti matriks tersebut terdiri dari dua baris dan tiga kolom. Matriks B berukuran 3×2 karena memiliki tiga baris dan dua kolom.

4.3 Penjumlahan dan pengurangan matriks

Operasi penjumlahan dan pengurangan hanya dapat dilakukan jika dan hanya jika kedua matriks tersebut memiliki ukuran yang sama.

Contoh 4.3:

Diberikan tiga matriks sebagai berikut

$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \end{bmatrix}$, $B = \begin{bmatrix} 4 & 6 & 1 \\ 5 & 2 & 3 \end{bmatrix}$, dan $C = \begin{bmatrix} 3 & 2 \\ 5 & 6 \end{bmatrix}$.

Deklarasikan ketiga matriks tersebut dalam Matlab dan lakukan operasi penjumlahan berikut:

Operasi matriks	*Code program*

$A + B$	`>> A = [1 2 3;3 4 7];`
	`>> B = [4 6 1; 5 2 3];`
	`>> C = [3 2; 5 6];`
	`>> A+B`
	`ans =`
	`5 8 4`
	`8 6 10`

$A - B$	`>> A-B`
	`ans =`
	`-3 -4 2`
	`-2 2 4`

$A + C$	`>> A+C`
	`Error using +`
	`Matrix dimensions must agree.`

Investigasi apa yang menyebabkan penjumlahan $A + C$ menghasilkan *error*.

4.4 Perkalian matriks dengan skalar

Perkalian matrik dengan skalar akan menghasilkan matrik dengan ukuran yang sama dengan matriks tersebut namun elemen matrik akan bernilai sebesar hasil perkalian *element* matrik dengan skalar tersebut.

Contoh 4.4:

Diberikan matriks $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \end{bmatrix}$, hitunglah nilai $B = 5A$.

Jawab:

$B = 5A = \begin{bmatrix} 5 & 10 & 15 \\ 15 & 20 & 35 \end{bmatrix}$,

Deklarasi program Matlab untuk perkalian tersebut adalah:

```
>> A = [1 2 3;3 4 7]
A =
     1     2     3
     3     4     7

>> size(A)
ans =
     2     3

>> B=5*A
B =
     5    10    15
    15    20    35

>> size(B)
ans =
    2     3
```

4.5 Perkalian matriks dengan matriks

Operasi perkalian matrik A dan B dapat dilakukan jika dan hanya jika matriks A memiliki ukuran $m \times n$ dan matriks B memiliki ukuran $n \times p$. Hasil perkalian kedua matriks tersebut akan menghasilkan matriks dengan ukuran $m \times p$.

$$[A]_{m\times n}[B]_{n\times p} = [C]_{m\times p}$$

Contoh 4.5:

Diberikan matriks $A = \begin{bmatrix} 2 & -1 \\ 3 & 4 \end{bmatrix}$ dan $B = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \end{bmatrix}$, hitunglah nilai:

a. $C = AB$
b. $D = BA$

Jawab:

a. Penjumlahan matriks $C = AB$

Operasi matriks	***Code program***

$C = AB$	`>> A=[2 -1; 3 4];`
	`>> B = [1 2 3; 3 4 7];`
	`>> C=A*B`
	`C =`
	`-1 0 -1`
	`15 22 37`
	`>> size(C)`
	`ans =`
	`2 3`

b. Perkalian matriks $D = BA$

Operasi matriks	***Code program***
$D = BA$	>> A=[2 -1;3 4] A = 2 -1 3 4 >> size(A) ans = 2 2 >> B = [1 2 3;3 4 7] B = 1 2 3 3 4 7 >> size(B) ans = 2 3 >> D=B*A *Error using* *

Investigasi apa yang menyebabkan operasi perkalia D=B*A terjadi *error*.

4.6 *Transpose* Matriks

Transpose adalah sebuah operasi untuk merubah posisi suatu elemen matriks yang terletak pada baris m dan kolom n menjadi baris n dan kolom m. Operasi *transpose* sutu matriks secara matematik dinotasikan dengan tanda superscript T suatu matriks. Di dalam Matlab, *transpose* dinyatakan dengan memberikan tanda petik satu (‘) pada suatu matriks. Sebagai contoh *transpose* suatu matriks A secara matematik dinyatakan oleh A^T dan dalam Matlab dinyatakan oleh A’.

Contoh 4.6:

Matriks & *transpose* matriks	**Expresi di Matlab**

$[3]^T = [3]$	`>> [3]'`
	`ans`
	`3`

$[3 \quad 4]^T = \begin{bmatrix}3\\4\end{bmatrix}$	`>> [3 4]'`
	`ans`
	`3`
	`4`

$b = [2 \quad 7 \quad 8]$
$b^T = \begin{bmatrix}2\\7\\8\end{bmatrix}$
	`>> b=[2 7 8]`
	`ans`
	`2 7 8`
	`>>b'`

$A = \begin{bmatrix}2 & -1\\3 & 4\end{bmatrix}$
$A^T = \begin{bmatrix}2 & 3\\-1 & 4\end{bmatrix}$
	`>> A=[2 -1; 3 4]`
	`A =`
	`2 -1`
	`3 4`
	
	`>> A'`
	`ans =`
	`2 3`
	`-1 4`

4.7 Determinan Matriks

Determinan suatu matriks A dapat dihitung menggunakan Matlab dengan perintah:

```
det(A)
```

Contoh 4.7:

Persoalan matematika	**Solusi menggunakan Matlab**

1. Hitunglah determinan dari matriks A berikut:
$A = \begin{bmatrix} 2 & -1 \\ 3 & 4 \end{bmatrix}$
	`>> A=[2 -1; 3 4]`
	`A =`
	`2 -1`
	`3 4`
	`>> det(A)`

2. Hitunglah determinan dari matriks B berikut:
$B = A^T = \begin{bmatrix} 2 & 3 \\ -1 & 4 \end{bmatrix}$
	`>> B=A'`
	`B =`
	`2 3`
	`-1 4`
	`>> det(B)`

3. Hitunglah determinan dari matriks C berikut:
$C = \begin{bmatrix} 1 & 3 & 4 \\ 5 & 6 & 2 \\ 4 & 5 & 3 \end{bmatrix}$
	`>> C=[1 3 4;5 6 2;4 5 3]`
	`C =`
	`1 3 4`
	`5 6 2`
	`4 5 3`
	`>> det(C)`

4.8 Inverse Matriks

Inverse suatu matriks A dinyatakan secara matematik oleh A^{-1} dan dihitung menggunakan Matlab dengan perintah:

```
inv(A)
```

Contoh 4.8:

Persoalan matematika	Solusi menggunakan Matlab

1. Hitunglah inverse dari matriks A berikut:
$A = \begin{bmatrix} 2 & -1 \\ 3 & 4 \end{bmatrix}$
	`>> A=[2 -1; 3 4]`
	`A =`
	`2 -1`
	`3 4`
	`>> inv(A)`

2. Hitunglah inverse dari matriks B berikut:
$B = A^T = \begin{bmatrix} 2 & 3 \\ -1 & 4 \end{bmatrix}$
	`>> B=A'`
	`B =`
	`2 3`
	`-1 4`
	`>> inv(B)`

3. Hitunglah inverse dari matriks C berikut:
$C = \begin{bmatrix} 1 & 3 & 4 \\ 5 & 6 & 2 \\ 4 & 5 & 3 \end{bmatrix}$
	`>> C=[1 3 4;5 6 2;4 5 3]`
	`C =`
	`1 3 4`
	`5 6 2`
	`4 5 3`
	`>> inv(C)`

5. *Loop Computation*

Dalam bab ini akan dibahas *loop computation* menggunakan *for* dimana suatu proses komputasi dilakukan berulang hingga jumlah perulangan tertentu yang telah didefinisikan. *Loop computation* menggunakan for sering digunakan dalam penghitungan matematika, misalnya dalam penghitungan integral. Dalam Matlab *loop computation* biasanya ditulis dalam m-*file*.

5.1 *Loop computation* menggunakan *for*

Loop computation menggunakan *for* pada Matlab biasanya ditulis dalam m-*file* dengan format sebagai berikut:

```
for k=1:N

   proses komputasi

end
```

dimana `k` menyatakan indeks pengulangan dan N menyatakan banyaknya pengulangan.

Dalam pemrograman menggunakan Matlab, indeks pengulangan k tersebut merupakan bilangan bulat positif (k > 0). Dalam *loop* tersebut, proses dikerjakan secara berulang sebanyak N kali pengulangan.

5.2 Aplikasi *loop computation*

Proses komputasi dalam loop banyak sekali digunakan untuk penyelesaian permasalahan dalam kehidupan sehari-hari. Berikut diberikan beberapa contoh aplikasi loop for untuk penyelesaian beberapa kasus.

Contoh 5.1:

Hitung jumlah bilangan bulat dari 0 sampai dengan 20

Jawab:

a. Secara matematik penjumlahan tersebut dapat dinyatakan sebagai berikut:

$$y = 1 + 2 + \cdots + 20 = \sum_{k=1}^{20} k$$

b. Penjumlahan tersebut dapat dinyatakan dalam program Matlab dalam m-*file* sebagai berikut:

```
y=0;      % initial value
for k=1:20
    y=y+k;
end
```

Dalam program diatas, diperlukan inisialisasi y untuk mendefinisikan nilai awal y. Program bekerja dengan alur sebagai berikut:

i. Nilai y awal : y=0
ii. Pada saat k= 1, nilai y=0 (dari nilai awal) dan y=y+k mengimplikasikan y=0+1 sehingga menghasilkan nilai y=1.

iii. Pada saat k= 2, nilai y=1 (dari perhitungan di k=1) dan y=y+k mengimplikasikan y=1+1 dan menghasilkan nilai y=2.

iv. Proses perhitungan tersebut berulang hingga k=20.

Contoh 5.2:

Hitung jumlah bilangan bulat dari 0 sampai dengan 10

Jawab:

a. Nyatakan penjumlahan tersebut dalam persamaan matematika sebagai berikut:

$$y = 0+1+2+\cdots+10 = \sum_{n=0}^{10} n$$

b. Tuliskan Matlab programnya dalam m-*file*.

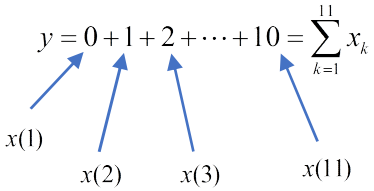
Penjumlahan $\sum_{n=0}^{10} n$ memiliki indeks dari 0 hingga 10. Dikarenakan indeks dalam Matlab paling kecil adalah 1 maka penjumlahan tersebut harus diubah sehingga n dalam penjumlahan tersebut dimulai dari 1, sebagaimana ditunjukan dalam persamaan berikut:

$$y = \sum_{n=0}^{10} n = \sum_{n=0+1}^{10+1} (n-1) = \sum_{n=1}^{11} (n-1)\,.$$

Sehinga dapat dibuat program Matlab dalam m-*file* sebagai berikut:

```
y=0;        % initial value
for n=1:11
    y=y+(n-1);
end
```

c. Program untuk penjumlahan diatas dapat juga dibuat dengan mendefinisikan suatu variabel baru, misalkan x sebagai berikut:



dimana $x(k) = k - 1$. Sehingga dapat dibuat program sebagai berikut:

```
y=0;        % initial value
for k=1:11
    x(k)=k-1;
    y=y+x(k);
end
```

Contoh 5.3:

Berikut diberikan data *temperature* di suatu kota untuk rentang waktu 15 hari. Hitunglah rata-rata *temperature* dalam selang waktu tersebut.

Date	11	12	13	14	15	16	17	18	19	20
Temperature	32	35	37	35	31	35	29	30	26	26

Jawab:

Penghitungan nilai *temperature* rata-rata dilakukan dengan langkah-berikut:

a. Nyatakan *temperature* dalam *row vector*:
b. Hitung jumlah data
c. Hitung total *temperature* dengan *loop* for
d. Hitung nilai rata-rata = nilai total / jumlah data

dan dituliskan dalam m-*file* sebagai berikut:

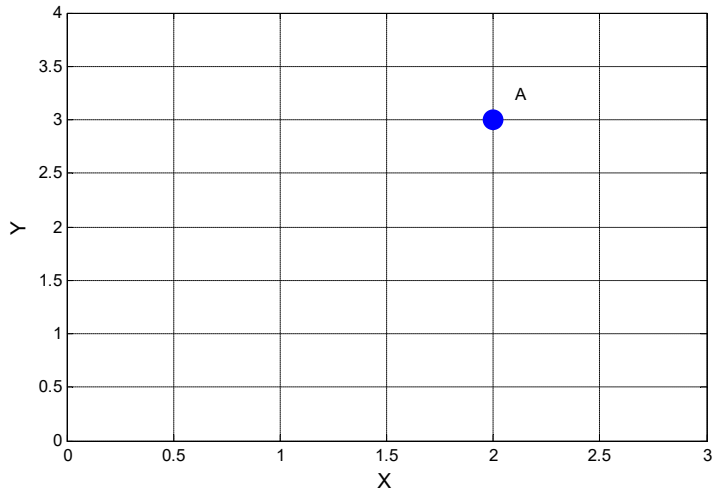
```
% a. Nyatakan temperature dalam row vector
T = [32,35,37,35,31,35,29,30,26,26];

% b. Hitung jumlah data
[Nr,Nc] = size(T);
N=Nc;

% c. Hitung total temperature (Tt)
Tt = 0;
for k=1:N
    Tt = Tt + T(k);
end

% d. Hitung temperature rata-rata (Tr)
Tr = Tt/N
```

6. Grafik

6.1 Pembuatan Grafik dari Satu Pasang Data

Grafik pada dasarnya merupakan visualisasi lokasi satu atau banyak titik dalam sebuah sistem koordinat. Grafik dua dimensi adalah visualisasi suatu titik pada sistem koordinat dua dimensi. Koordinat dua dimensi biasa dinyatakan oleh sistem koordinat XY. Sistem koordinat XY memiliki dua sumbu yang saling tegak lurus, yaitu sumbu X dan sumbu Y. Lokasi suatu titik pada sistem koordinat XY dinyatakan oleh koordinat (x,y), dimana x merupakan posisi pada sumbu X dan y merupakan posisi pada sumbu Y.

Gambar 6.1 memberikan contoh lokasi titik A pada sistem koordinat XY. Titik A terletak pada posisi 2 di sumbu X dan 3 di sumbu Y, sehingga koordinat titik A adalah (2,3).



Gambar 6.1: Grafik yang menampilkan lokasi titik A pada koordinat XY.

Data koordinat suatu titik diperlukan untuk menggambarkan posisi titik tersebut dalam sebuah sistem koordinat. Pembuatan grafik menggunakan Matlab dapat dilakukan dengan perintah ***plot(x,y)***, dimana (x,y) merupakan koordinat suatu titik.

Sebagai contoh untuk menggambarkan grafik yang berisikan sebuah titik seperti ditunjukan dalam Gambar 6.1 digunakan *code* sebagai berikut:

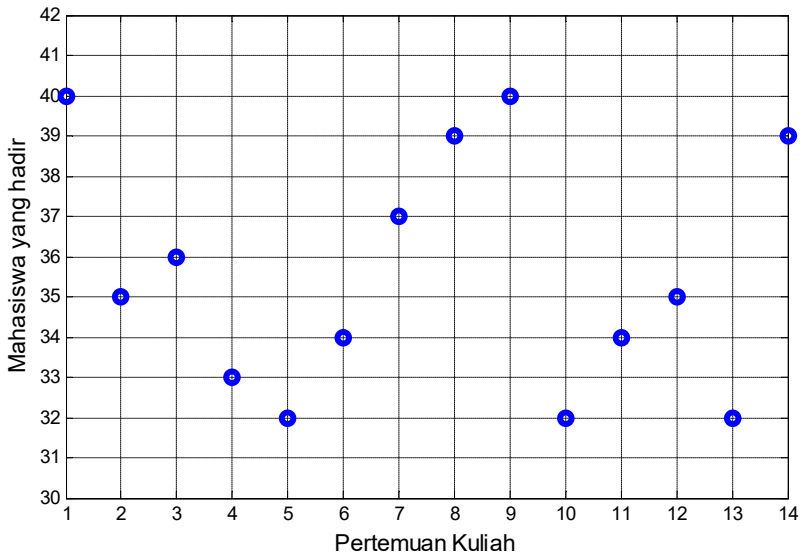
```
>> x=2;

>> y=3;

>> plot(x,y)
```

Code diatas merupakan perintah untuk menggambarkan grafik sebuah titik yang memiliki koordinat (2,3). Dengan menjalankan *code* tersebut akan muncul suatu grafik beriisikan satu titik. Namun demikian titik yang digambarkan begitu kecil sehingga tidak jelas nampak. Untuk memperjelas tampilan grafik yang dihasilkan, dapat digunakan marker dengan memodifikasi *code program* tersebut menjadi seperti berikut:

```
>> x=2;

>> y=3;

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

6.2 Pembuatan Grafik dari Banyak Data

Sering kali kita menjumpai grafik yang terdiri dari beberapa titik. Sebagai contoh grafik dari data mahasiswa yang mengikuti kuliah untuk 14 kali pertemuan seperti yang ditampilkan dalam Gambar 6.2.



Gambar 6.2: Kehadiran mahasiswa dalam perkuliahan.

Grafik tersebut menunjukan data bahwa pada pertemuan pertama mahasiswa yang hadir sebanyak 40 orang, pada pertemuan kedua sebanyak 35 orang, pada pertemuan ketiga sebanyak 36 orang, dan seterusnya hingga pertemuan keempat-belas sebanyak 39 orang. Data jumlah mahasiswa yang hadir dalam perkuliahan tersebebut secara lengkap diberikan dalam Tabel 6.1.

Tabel 6.1: Data kehadiran mahasiswa dalam sebuah perkuliahan.

Pertemuan kuliah	Jumlah mahasiswa yang hadir
1	40
2	35
3	36
4	33
5	32
6	34
7	37
8	39
9	40
10	32
11	34
12	35
13	32
14	39

Pada bagian sebelumnya, telah dibahas perintah dalam Matlab untuk menggambarkan grafik suatu titik adalah ***plot(x,y)***. Perintah yang sama juga digunakan untuk menggambarkan grafik dari beberapa titik, namun demikian x dan y tidak lagi berisikan koordiat satu titik namun beberapa titik, sehingga x dan y merupakan suatu vektor. Berdasarkan data dalam Tabel 6.1, vektor x dan y dapat dibangun sebagai berikut:

- x : vektor yang berisikan data pertemuan kuliah

  x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
- y: vektor yang berisikan data jumlah mahasiswa yang hadir

  y=[40, 35, 36, 33, 32, 34, 37, 39, 40, 32, 34,35, 32, 39]

Dengan menggunakan data kedua vektor tersebut, grafik seperti pada Gambar 6.2 dapat dibuat menggunakan Matlab dengan *code program* sebagai berikut:

```
>> x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14];
>>
y=[40,35,36,33,32,34,37,39,40,32,34,35,32,39
];
>> plot(x,y,'o')
```

Catatan:

Pastikan kedua vektor memiliki jumlah data yang sama, bisa dicek dengan menggunakan perintah ***size(x)*** dan ***size(y).***

6.3 Pembuatan Grafik Sebuah Fungsi

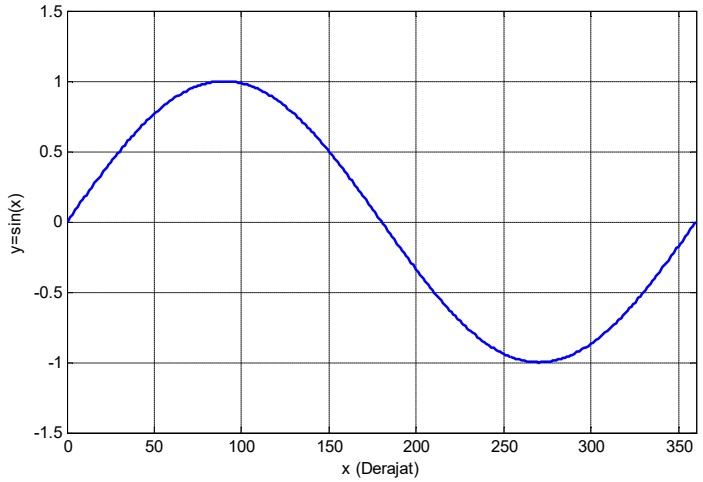
Grafik dapat digunakan untuk visualisasi suatu fungsi.

Contoh 6.1:

Gambarkan grafik dari fungsi $y = \sin(x)$ untuk $0 \leq x \leq 360^o$.

Jawab:

```
>> x=0:1:360;
>> xd=x*pi/180;
>> y=sin(xd);
>> plot(x,y)
>> xlabel(' x (Derajat) ')
>> ylabel(' y = sin(x) ')
```



Gambar 6.3: Grafik y=sin(x)

Perintah berikut digunakan untuk memberikan nama (label) pada sumbu x dan sumbu y dalam grafik tersebut.

```
>> xlabel('x (Derajat)')
>> ylabel('y = sin(x)')
```

7. *Conditional Statement*

Conditional statement merupakan sebuah proses komputasi dimana output yang dihasilkan bergantung pada suatu kondisi. Jika suatu kondisi terpenuhi, proses komputasi tersebut akan memberikan suatu hasil, namun jika kondisi tersebut tidak terpenuh maka proses komputasi tersebut akan memberikan hasil yang lain.

7.1 *Conditional Statement* dalam Kehidupan Sehari-hari

Conditional statement sering dijumpai dalam kehidupan sehari-hari, seperti contoh berikut:

Contoh 7.1:

Seorang pedagang menjual baju dengan harga Rp. 100.000 untuk satu buah baju dan Rp. 80.000 per baju untuk pembelian lebih dari sepuluh baju.

Contoh 7.2:

Seorang anak sekolah dasar ingin mengetahui sebuah bilangan merupakan bilangan genap atau ganjil. Proses yang dilakukan anak tersebut adalah membagi bilangan tersebut dengan dua. Apabila bilangan tersebut dapat dibagi dua maka bilangan tersebut merupakan bilangan genap, namun bila tidak dapat dibagi dua maka bilangan tersebut merupakan bilangan ganjil.

7.2 Pemrograman dengan *Conditional Statement*

Permrograman dengan *conditional statement* bekerja dengan cara membandingkan suatu kondisi yang akan dieksekusi dengan kondisi yang menjadi persyaratan. Perbandingan ini akan memberikan nilai benar atau salah. Nilai benar akan dihasilkan jika kondisi yang akan dieksekusi memenuhi kondisi yang menjadi persyaratan dan bernilai salah bila tidak memenuhi kondisi persyaratan.

Perbandingan yang dilakukan dalam *conditional statement* ini merupakan suatu perbandingan terhadap suatu kondisi seperti yang ditunjukan dalam Tabel 7.1.

Tabel 7.1: Simbol Perbandingan

Perbandingan	**Ekpresi dalam** MATLAB
Sama dengan	==
Lebih dari	>
Lebih dari atau sama dengan	>=
Kurang dari	<
Kurang dari atau sama dengan	<=
Tidak sama dengan	~=

Conditional statement dalam pemrograman menggunakan MATLAB memiliki format sebagi berikut:

```
if <kondisi terpenuhi>
   <jalankan proses>
else
   <jalankan proses yang lain>
end
```

Berikut diberikan dua contoh program dengan mengunakan *conditional statement*.

Contoh 7.3:

Dari Contoh 7.1 diketahui seorang pedagang yang menjual baju dengan harga Rp. 100.000 untuk satu buah baju dan Rp. 80.000 per baju untuk pembelian lebih dari sepuluh baju. Buatlah sebuah program untuk menghitung harga total dalam transaksi pembelian.

Jawab:

a. Tuliskan *code program* berikut dalam m-*file* dan simpan dengan nama prog_501.m

```
n = input ('Masukan jumlah baju yang dibeli: ');
if  n > 10
  h = n*80000;
else
  h = n*100000;
end

disp('Harga baju keseluruhan: ')
disp(h)
```

b. Jalan program tersebut, misal untuk dua transaksi dimana pembelian baju sebanyak 8 baju pada transaksi pertama dan 15 baju pada transaksi kedua, maka akan diperoleh hasil berikut:

```
>> prog_501
Masukan jumlah baju yang dibeli: 8
Harga baju keseluruhan:
      800000
>> prog_501
Masukan jumlah baju yang dibeli: 15
Harga baju keseluruhan:
     1200000
```

Contoh 7.4:

Dari Contoh 7.2 diketahui proses untuk mengetahui sebuah bilangan merupakan bilangan genap atau ganjil. Proses tersebut adalah bilangan tersebut dengan dua. Apabila bilangan tersebut dapat dibagi dua (tidak ada sisa) maka bilangan tersebut merupakan bilangan genap, namun bila tidak maka bilangan tersebut merupakan bilangan ganjil. Buatlah program untuk menentukan sebuah bilangan merupakan bilangan ganjil atau genap.

Jawab:

a. Dalam MATLAB tersedia fungsi ***rem(x,y)*** yang berguna untuk menghitung sisa setelah pembagian (*remainder*). Sebuah bilang x habis dibagi oleh y jika sisa setelah

pembagian adalah nol. Sehingga suatu bilangan a merupakan bilangan genap jika ***rem(a,2)*** menghasilkan nilai nol.

b. Tuliskan *code program* berikut dalam m-*file* dan simpan dengan nama prog_502.m

```
a = input ('Masukan suatu bilangan: ');
n = rem(a,2);
if  n == 0
  disp('Bilangan tersebut bilang genap.');
else
  disp('Bilangan tersebut bilang ganjil.');
end
```

c. Jalankan program tersebut dengan memasukan suatu bilangan, sebagai contoh 7 dan 6.

```
>> prog_502
Masukan suatu bilangan: 7
Bilangan tersebut merupakan bilang ganjil.


>> prog_502
Masukan suatu bilangan: 6
Bilangan tersebut merupakan bilang genap.
```

Daftar Pustaka

Chapra, S. C. and R. P. Canale, *Numerical Methods forEngineers,*6th ed., McGraw-Hill, New York, 2010.

Chapra, S.C., *Applied Numerica Methods with* MATLAB, 4th ed., Mc Graw Hill Education, New York, 2018

Moore, H., MATLAB *for Engineers,* 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2008.

UPJ Press
Universitas Pembangunan Jaya
Jl. Cendrawasih Raya B7/P, Sawah Baru,
Ciputat, Tangerang Selatan 15413

ISBN 978-623-7455-10-3