

Bab I

Pendahuluan



Logika dan pemrograman computer merupakan salah satu mata kuliah dasar di Program Studi Teknik Mesin dengan bahasa pemrograman Matlab. Mata kuliah ini disajikan pada semester III yang meliputi pembahasan mengenai algoritma, logika dan bahasa pemrograman, matematika sederhana, operasi array, fungsi M-file, system GUI, Graphic, Simulink dan control sistem. Logika dan pemrograman komputer sangat ditunjang oleh mata kuliah tertentu khususnya kalkulus karena dalam pemrograman komputer sangat diperlukan pemahaman di bidang matematika sederhana dan matematika model. Logika dan pemrograman komputer merupakan penunjang yang sangat mendasar bagi mata kuliah teknik control.

Bahasa pemrograman sebagai media untuk berinteraksi antara manusia dengan computer dewasa ini dibuat agar semakin mudah dan cepat. Banyak bahasa pemrograman yang bisa digunakan dalam pemecahan masalah keteknikan, seperti C++, Pascal, Delphi, Visual basic, Java dan yang lainnya. Semua itu mampu membantu kita dalam berinteraksi dengan computer dan masalah keteknikan. Dalam perkuliahan Logika dan Pemrograman Komputer khususnya di Teknik Mesin Universitas Udayana bahasa pemrograman yang dipakai adalah bahasa Matlab.

Matlab dikembangkan sebagai bahasa pemrograman sekaligus alat visualisasi yang menawarkan banyak kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan matematika, rekayasa teknik, fisika, statistika, komputasi dan modeling. Matlab dibangun dari bahasa induknya yaitu bahasa C, namun tidak dapat dikatakan sebagai varian dari C, karena dalam sintak maupun cara kerjanya sama sekali berbeda dengan C. Namun dengan hubungan langsungnya terhadap C, Matlab memiliki kelebihan-kelebihan bahasa C bahkan mampu berjalan pada semua platform Sistem Operasi tanpa mengalami perubahan sintak sama sekali.

Matlab merupakan singkatan dari Matric Laboratory, yakni merupakan bahasa pemrograman haig perpomace, bahasa pemrograman level tinggi yang khususnya untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan.

Bab II

Algoritma



2.1 Konsep Algoritma

Algoritma merupakan pondasi yang harus dikuasai oleh setiap mahasiswa yang ingin menyelesaikan suatu masalah secara berstruktur, efektif, dan efisien, teristimewa lagi bagi mahasiswa yang ingin menyusun program komputer untuk menyelesaikan suatu persoalan. Konsep dan dasar-dasar penyusunan algoritma akan dibahas dalam bab ini.

Definisi

Algoritma:

1. Teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas, tetapi tersusun secara logis dan sistematis.
2. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya.
3. Algoritma adalah sekelompok aturan untuk menyelesaikan perhitungan yang dilakukan oleh tangan atau mesin.
4. Algoritma adalah langkah demi langkah sebuah prosedur berhingga yang dibutuhkan untuk menghasilkan sebuah penyelesaian
5. Algoritma adalah langkah –langkah perhitungan yang mentransformasikan dari nilai masukan menjadi keluaran
6. Algoritma adalah urutan operasi yang dilakukan terhadap data yang terorganisir dalam struktur data
7. Algoritma adalah sebuah program abstrak yang dapat dieksekusi secara fisik oleh mesin
8. Algoritma adalah sebuah model perhitungan yang dilakukan oleh computer

Catatan Sejarah

Abu Ja'far Muhammad Ibnu Musa Al-Kwarizmi, penulis buku "Aljabar wal muclabala" beberapa abad yang lalu (pada abad IX), dianggap sebagai pencetus pertama Algoritma karena di dalam buku tersebut Abu Ja'far menjelaskan langkah-langkah

dalam menyelesaikan berbagai persoalan aritmetika (aljabar), Kemungkinan besar kata "Algoritma" diambil dari kata "Al-Kwarizmi" yang kemudian berubah menjadi "Algorism", selanjutnya menjadi "Algorithm".

Ciri Algoritma

Donald E. Knuth, seorang penulis beberapa buku algoritma abad XX, menyatakan bahwa ada beberapa ciri algoritma, yaitu:

- Algoritma mempunyai awal dan akhir. Suatu algoritma harus berhenti setelah mengerjakan serangkaian tugas atau dengan kata lain suatu algoritma memiliki langkah yang terbatas.
- Setiap langkah harus didefinisikan dengan tepat sehingga tidak memiliki arti ganda (*not ambiguous*).
- Memiliki masukan (input) atau kondisi awal.
- Memiliki keluaran (output) atau kondisi akhir.
- Algoritma harus efektif; bila diikuti benar-benar akan menyelesaikan persoalan.

Algoritma bisa ditemukan dalam kehidupan sehari-hari, misalnya sebagai berikut:

Tabel 2.1 Contoh Algoritma dalam kehidupan sehari-hari

Proses	Algoritma	Contoh langkah
1. Membuat Kue	Resep Kue	Campurkan 2 butir telur ke dalam adonan, kemudian kocok hingga mengembang.
2. Membuat Pakaian	Pola pakaian	Gunting kain dad pinggir kid bawah ke arah kanan atas sepanjang 15 cm.
3. Praktikum Kimia	Petunjuk Praktikum	Campurkan 10 ml Asam Sulfat ke dalam 15 ml Natrium hidroksida.

Sifat Algoritma

Berdasarkan ciri algoritma yang dipaparkan oleh Donald Knuth dan definisi Algoritma maka dapat disimpulkan sifat utama suatu algoritma, yaitu sebagai berikut:

- **input:** Suatu algoritma memiliki input atau kondisi awal sebelum algoritma dilaksanakan dan bisa berupa nilai-nilai pengubah yang diambil dari himpunan khusus.
- **output:** Suatu algoritma akan menghasilkan output setelah dilaksanakan, atau algoritma akan mengubah kondisi awal menjadi kondisi akhir, di mana nilai output diperoleh dari nilai input yang telah diproses melalui algoritma.

- **definiteness:** Langkah-langkah yang dituliskan dalam algoritma terdefinisi dengan jelas sehingga mudah dilaksanakan oleh pengguna algoritma.
- **finiteness:** Suatu algoritma harus memberi kondisi akhir atau output setelah melakukan sejumlah langkah yang terbatas jumlahnya untuk setiap kondisi awal atau input yang diberikan.
- **effectiveness:** Setiap langkah dalam algoritma bisa dilaksanakan dalam suatu selang waktu tertentu sehingga pada akhirnya memberi solusi sesuai yang diharapkan.
- **generality:** Langkah-langkah algoritma berlaku untuk setiap himpunan input yang sesuai dengan persoalan yang akan diberikan, tidak hanya untuk himpunan tertentu.

Agar bentuk algoritma dan proses penyusunannya dapat mulai dipahami maka berikut ini akan diuraikan proses pembuatan algoritma mulai bentuk yang menggunakan bahasa sehari-hari, disusul penjelasan format algoritma yang dapat dijadikan acuan, dan beberapa contoh pembuatan algoritma yang mengikuti format tersebut.

Contoh: Susun algoritma untuk mencari angka terbesar (maksimum) dari suatu kumpulan

bilangan bulat yang terbatas jumlahnya.

Solusi:

1. Anggaplah angka pertama dalam kumpulan tersebut adalah yang terbesar (maksimum).
2. Bandingkan angka maksimum ini dengan angka berikutnya dalam kumpulan. Bila angka berikut tersebut lebih besar maka jadikanlah maksimum.
3. Ulangi langkah 2 ini sehingga tidak ada lagi angka yang tersisa dalam himpunan.
4. Hentikan perbandingan setelah semua angka selesai dibandingkan sehingga angka terbesar dalam himpunan tersebut adalah angka maksimum terakhir.

Solusi tersebut dalam bentuk simbol instruksi adalah sebagai berikut:

- andaikan N = banyaknya angka dalam himpunan
- $index = 1$
- $maksimum = angka(index)$
- selama $index < N$, lakukan:
 - $index = index + 1$

- bila $\text{angka}(\text{index}) > \text{maksimum}$ maka $\text{maksimum} = \text{angka}(\text{index})$
- ulangi lagi untuk index berikutnya
- angka terbesar dalam himpunan adalah maksimum terakhir.

Suatu algoritma tentu dapat ditulis dengan menggunakan bahasa sehari-hari seperti contoh di atas. Namun, algoritma seperti ini masih sulit untuk langsung diterjemahkan apabila akan diimplementasi ke dalam suatu bahasa pemrograman komputer. Setiap algoritma tentu saja memerlukan suatu langkah "peralihan" ke suatu bahasa program ketika akan dilaksanakan dengan menggunakan komputer sehingga semakin dekat bentuk algoritma ini ke bentuk program komputer maka semakin mudah diterjemahkan. Suatu algoritma juga dapat ditulis langsung dalam bentuk "bahasa program", tetapi algoritma seperti ini hanya bisa digunakan untuk bahasa program tertentu dan terpaksa diubah kembali untuk disesuaikan apabila akan diimplementasi dengan bahasa program yang lain. Suatu struktur tertentu akan diperkenalkan dalam buku ini agar algoritma dapat diterima secara umum dan cukup efisien apabila akan diterjemahkan ke dalam bahasa program.

Struktur Algoritma

Agar algoritma dapat ditulis lebih teratur maka sebaiknya dibagi ke dalam beberapa bagian. Salah satu struktur yang sering dijadikan patokan adalah berikut:

- **Bagian Kepala (Header):** memuat nama algoritma serta informasi atau keterangan tentang algoritma yang ditulis.
- **Bagian Deklarasi/Definisi Variabel:** memuat definisi tentang nama variabel, nama tetapan, nama prosedur, nama fungsi, tipe data yang akan digunakan dalam algoritma.
 - **Bagian Deskripsi/Rincian Langkah:** memuat langkah-langkah penyelesaian masalah, termasuk beberapa perintah seperti baca data, tampilkan, ulangi, yang mengubah data input menjadi output, dan sebagainya.

Contoh 1: Berikut ini adalah contoh struktur sebuah algoritma:

Algoritma Luas - lingkaran

{ menghitung luas sebuah lingkaran apabila jari-jari lingkaran tersebut diberikan }

Deklarasi

| Definisi nama tetapan }

```

const N = 10;
const phi = 3.14;
definisi nama,peubah/variabel }
real jari_jari, luas;

```

Deskripsi

```

read(jari-jari);
luas = phi * jari_jari * jari_jari;
write(luas);

```

Contoh 2: Contoh berikut ini adalah algoritma untuk menghitung nilai rata sejumlah angka yang dimasukkan lewat keyboard.

Algoritma Nilai_Rata

(menghitung nilai rata sejumlah bilangan yang dimasukkan lewat keyboard)

Definisi Variabel

```

integer x, N, k, jumlah;
real nilai_rata;

```

Rincian Langkah

```

{ masukkan jumlah data }
read(N);
k  1;
jumlah  0;
while (k <= N) do
    ( baca data )
    read(x);
    jumlah  jumlah + x;
    k  k + 1;
endwhile

( hitung nilai rata )
nilai_rata  jumlah / N;
write(nilai_rata);

```

Contoh 3: Contoh algoritma yang menerima dua buah angka bulat kemudian menampilkan angka yang lebih besar.

Algoritma Lebih_besar

(menerima dua angka kemudian menampilkan angka yang lebih besar)

Definisi Variabel

```

integer angka 1, angka 2;

```

Rincian Langkah

```

( memasukkan angka }
write ("Masukkan angka 1:"); read (angka1);
write ("Masukkan angka 2:"); read (angka2);
( periksa yang lebih besar }
if (angka1 > angka2)
    then write("yang lebih besar=",angka1);
    else write("yang lebih besar=", angka2);
endif.

```

Contoh 4: Algoritma berikut ini mencari angka terbesar dari suatu himpunan angka.

Algoritma Terbesar

(membentuk himpunan angka, kemudian mencari angka terbesar }

Definisi Variabel

```

integer N=25, max, indeks;
integer Angka[N];

```

Rincian Langkah

```

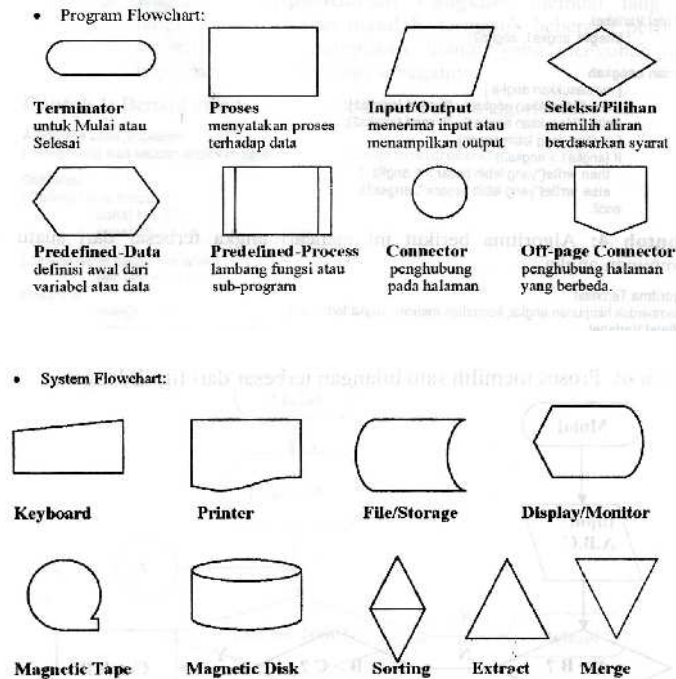
( memasukkan anggota himpunan angka }
for( i = 1 to N step 1)
    write("Masukkan angka ke:",i);
    read( Angka[ i ] );
endfor.
{ max adalah angka pertama }
max   Angka[ 1 ];
indeks  1;
( bandingkan max dengan setiap angka dalam himpunan }
while ( indeks < N) do
    indeks   indeks + 1;
    if ( Angka[indeks] > max )
        then max   Angka[indeks];
enddo.
write ( "Angka terbesar=",max );

```

2.2 Flowcharting

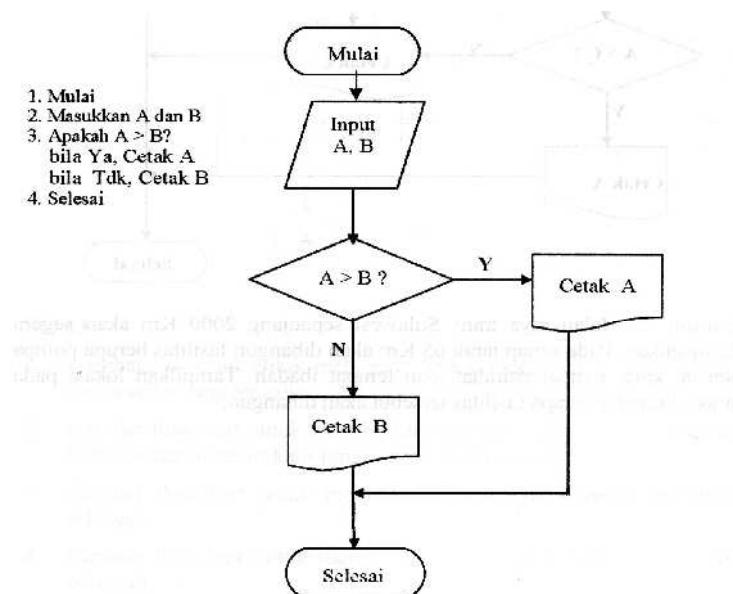
Flowchart adalah suatu teknik untuk menyusun rencana program telah diperkenalkan dan telah dipergunakan oleh kalangan programmer komputer sebelum algoritma menjadi populer, yaitu *flowcharting*. Flowchart adalah untaian simbol gambar (chart) yang

menunjukkan aliran (flow) dari proses terhadap data. Simbol-simbol flowchart dapat diklasifikasikan menjadi simbol untuk program dan simbol untuk sistem (peralatan hardware).

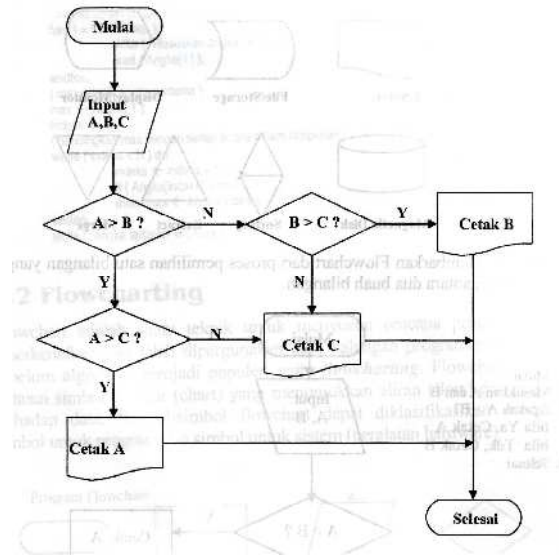


Gambar 2.1 Simbol program flowchart dan system flowchart

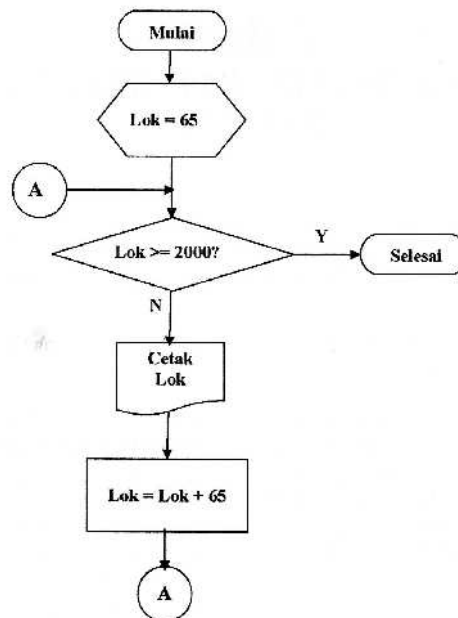
Contoh : Gambarkan Flowchart dari proses pemilihan satu bilangan yang lebih besar di antara dua buah bilangan.



Contoh : Proses memilih satu bilangan terbesar dari tiga bilangan.



Contoh : Jalan raya trans Sulawesi sepanjang 2000 Km akan segera ditingkatkan. Pada setiap jarak 65 Km akan dibangun fasilitas berupa pompa bensin, kafe, tempat istirahat, dan tempat ibadah. Tampilkan lokasi pada jarak kilometer berapa fasilitas tersebut akan dibangun.



2.3 Analisa Algoritma

Untuk suatu fungsi polynomial $f(n)$ dengan n data masukan, terdapat tiga keadaan yang bisa muncul selama waktu tempuh algoritma dan disebut kompleksitas waktu yaitu Worst Case, Average Case dan Best Case. Penjelasan masing-masing kompleksitas sebagai berikut:

1. **Worst Case** merupakan waktu tempuh yang bernilai maksimum dari suatu fungsi $f(n)$ untuk setiap input yang mungkin. Keadaan ini disebut sebagai keadaan terburuk (keadaan terjelek) dari sebuah algoritma.
2. **Average Case** merupakan suatu keadaan dari waktu tempuh yang ekuivalen dengan nilai ekspektasi dari fungsi $f(n)$ untuk setiap input data yang mungkin. Fungsi $f(n)=e$ dan didefinisikan $e = n_1p_1 + n_2p_2 + \dots + n_kp_k$ dengan n_1, n_2, \dots, n_k merupakan nilai-nilai yang muncul, sedangkan p_1, p_2, \dots, p_k merupakan probabilitas dari setiap nilai (n_1) yang muncul.
3. **Best Case** merupakan waktu tempuh yang bernilai minimum dari suatu fungsi $f(n)$ untuk setiap input yang mungkin. Keadaan ini disebut sebagai keadaan terbaik dari suatu proses sebuah algoritma menyelesaikan permasalahan.

Kompleksitas sebuah algoritma adalah fungsi $g(n)$ yang berada diatas batas bilangan operasi (*running time*) yang dibentuk oleh algoritma ketika diberikan input berukuran n . Relatif sulit membuat statistic perilaku masukan, sehingga paling sering diberikan perilaku keadaan jelek (*worst case*). Waktu maksimum dari kompleksitas $g(n)$ didekati dengan $O(f(n))$ dimana $f(n)$ adalah salah satu dari fungsi berikut ini:

$f(n)=n$ (kompleksitas linier), $f(n)=\log n$ (kompleksitas logaritmatik), $f(n)=n^a$ dimana $a \geq 2$ (kompleksitas polynomial), $f(n)=a^n$ (kompleksitas eksponensial).

2.3 Himpunan

Dalam matematika himpunan adalah segala koleksi benda-benda tertentu yang dianggap sebagai satu kesatuan. Walaupun ini merupakan ide sederhana, himpunan merupakan salah satu konsep penting dan mendasar dalam matematika modern. Definisi lain dari himpunan adalah sekelompok obyek yang direpresentasikan dalam satu satuan (unit). Setiap objek dari himpunan disebut elemen atau anggota dan umumnya setiap elemen dalam himpunan mempunyai beberapa kesamaan sifat atau karakteristik. Unit

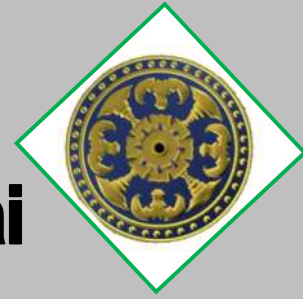
yang melingkupi beberapa anggota disebut semesta pembicaraan dan mempunyai paling sedikit satu elemen.

SOAL

1. Gambar flowchart untuk mengganti ban kempes sebuah mobil dengan ban reserve yang tersedia.
2. Gambar flowchart untuk menyiapkan secangkir kopi manis di pagi hari (dimulai dari memasak air hingga menghadirkan kopi).
3. Gambar flowchart untuk memilih satu bilangan terbesar dari empat bilangan.
4. Gambar flowchart untuk memilih satu bilangan terbesar dari N buah bilangan.

Bab III

Tipe Data, Variabel, Nilai dan Ekspresi



Pada prinsipnya, suatu program komputer memanipulasi data untuk menjadi informasi yang berguna. Dengan demikian, perlu dipahami beberapa hal yang berkaitan dengan data, yaitu *tipe data*, *variabel*, dan *nilai data* sebagai berikut:

- **Tipe data:** setiap data memiliki tipe data, apakah merupakan angka bulat (integer), angka biasa (real), atau berupa karakter (char), dan sebagainya.
- **Variabel:** setiap data diwakili oleh suatu variabel, dan variabel ini diberi nama agar bisa dibedakan terhadap variabel lainnya.
- **Mai:** setiap data memiliki harga atau nilai, misalnya umur seseorang diwakili oleh variabel UMUR yang bertipe bilangan, dan memiliki nilai 20 tahun. Perlu diketahui bahwa dalam representasi nilai data pada komputer, setiap tipe data memiliki batasan nilai masing-masing.

3.1 Tipe Data

Ada dua kategori dari tipe data, yaitu *tipe dasar* dan *tipe bentukan*. *Tipe dasar* adalah tipe data yang selalu tersedia pada setiap bahasa pemrograman, antara lain *bilangan bulat (integer)*, *bilangan biasa (real)*, *bilangan tetap (const)*, *karakter (character atau char)*, *logik (logic atau boolean)*. Tipe bentukan adalah tipe data yang dibentuk dari kombinasi tipe dasar, antara lain *larik (array)*, *rekaman (record)*, *string (string)*.

Tipe Dasar

1. Bilangan bulat (integer)

- Bilangan atau angka yang tidak memiliki titik desimal atau pecahan, seperti 10, +255, -1024, +32767.
- Tipe dituliskan sebagai **integer** atau **int**

- Jangkauan nilai bergantung pada implementasi perangkat keras komputer, misalnya dari -32768 s/d +32767; untuk algoritma tidak kita batasi.
- Operasi aritmetik: tambah +, kurang -, kali *, bagi /, sisa basil bagi
- Operasi perbandingan : lebih kecil <, lebih kecil atau sama <= lebih besar >, lebih besar atau sama >= sama=, tidak sama ><

2. Bilangan biasa (real)

- Bilangan atau angka yang bisa memiliki titik desimal atau pecahan, dan ditulis sebagai: 235.45, +1023.55, -987.3456 atau dalam notasi ilmiah seperti: 1.245E+03, 7.45E-02, +2.34E-04, -5.43E+04, dsb.
- Tipe dituliskan sebagai : **real**
- Jangkauan nilai: bergantung pada implementasi perangkat keras komputer, misalnya dari -2.9E-39 s/d +1.7E+38, untuk algoritma tidak kita batasi.
- Operasi aritmatik dan perbandingan juga berlaku bagi bilangan biasa.

3. Bilangan tetap (const)

- Bilangan tetap (**const**) adalah tipe bilangan, baik bernilai bulat maupun tidak, yang nilainya tidak berubah selama algoritma dilaksanakan.
- Tipe dituliskan sebagai **const**.
- Jangkauan nilai meliputi semua bilangan yang mungkin.

4. Karakter (character)

- Karakter adalah data tunggal yang mewakili semua huruf, simbol baca, dan juga simbol angka yang tidak dapat dioperasikan secara matematis, misalnya: 'A', 'B', ..., 'Z', 'a', 'b', ..., 'z', '?', '!', ':', ';', dst.
- Tipe dituliskan sebagai **char**.
- Jangkauan nilai meliputi semua karakter dalam kode ASCII, atau yang tertera pada setiap tombol keyboard.
- Operasi perbandingan dapat dilakukan dan dievaluasi menurut urutan kode ASCII, sehingga huruf 'A' (Hex 41) sebenarnya lebih kecil dari huruf 'a' (Hex 61).

5. Logik (logical)

- Tipe data logik adalah tipe data yang digunakan untuk memberi nilai pada hasil perbandingan, atau kombinasi perbandingan.
- Tipe dituliskan sebagai **boolean**
- Jangkauan nilai ada dua: **true** dan **false**

- Contoh: $45 > 56$ hasilnya **false**, Amir < Husni hasilnya **true**
- Ada beberapa operasi untuk data jenis logik, antara lain **and**, **or**, dan **not**

Tabel 3.1 Operasi untuk data logik

A	B	A and B	A or B	Not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Tipe Bentuk

1. Array (larik)

- Array adalah tipe data bentukan, yang merupakan wadah untuk menampung beberapa nilai data yang sejenis. Kumpulan bilangan bulat adalah array integer, kumpulan bilangan tidak bulat adalah array real.
- Cara mendefinisikan ada dua macam, yaitu:
 - *Nilai ujian : array [1 .. 10] of integer;* atau
 - *int nilai_ujian [0];*
- Kedua definisi di atas menunjukkan bahwa nilai_ujian adalah kumpulan dari 10 nilai bertipe bilangan bulat.

2. String

- String adalah tipe data bentukan yang merupakan deretan karakter yang membentuk satu kata atau satu kalimat, yang biasanya diapit oleh dua tanda kutip.
- Sebagai contoh: nama, alamat, dan judul adalah tipe string.
- Cara mendefinisikannya adalah:
 - *String Nama, Alamat;* atau
 - *Nama, Alamat : String;*

3. Record (rekaman)

- Record adalah tipe data bentukan yang merupakan wadah untuk menampung elemen data yang tipenya tidak perlu sama dengan tujuan mewakili satu jenis objek.
- Sebagai contoh, mahasiswa sebagai satu jenis objek memiliki beberapa elemen data seperti: nomer_stb, nama, umur, t4lahir, jenkel.

- Cara mendefinisikan record mahasiswa tersebut adalah sebagai berikut:

Type DataMhs : record

```
< nomer — stb : integer,
Nama_mhs : string,
umur : integer,
t4lahir : string,
jenkel : char;
```

3.2 Variabel

Variabel adalah nama yang mewakili suatu elemen data seperti: jenkel untuk jenis enis kelamin, t4lahir untuk tempat lahir, alamat untuk alamat, dan sebagainya. Ada aturan tertentu yang wajib diikuti dalam pemberian nama variabel, antara lain:

- Harus dimulai dengan abjad, tidak boleh dengan angka atau simbol.
- Tidak boleh ada spasi di antaranya
- Jangan menggunakan simbol-simbol yang bisa membingungkan seperti titik dua, titik koma, koma, dan sebagainya.
- Sebaiknya memiliki arti yang sesuai dengan elemen data.
- Sebaiknya tidak terlalu panjang.

Contoh variabel yang benar: Nama, Alamat, Nilai_Ujian

Contoh variabel yang salah: 4XYZ, IP rata, Var:+xy,458;

3.3 Pemberian Nilai

Ada dua cara yang dapat digunakan untuk memberi nilai pada suatu variabel, yaitu melalui proses *assignment* dan *pembacaan*.

Pemberian nilai dengan cara assignment mempunyai bentuk umum sebagai berikut:

- *Variabel nilai;*
- *Variabell variabel2;*
- *Variabel ekspresi;*

Contoh assignment:

- Nama "Ali bin AbuThalib";
- Jarak 100.56;
- X Jarak;
- Rentang X+50-3*Y;

Pemberian nilai dengan cara pembacaan dapat dilakukan melalui instruksi pembacaan dengan bentuk umum sebagai berikut:

- **read**(variabel); atau
- **read**(variabel1, variabel2,...);

Contoh pembacaan data:

- **read**(Nama);
- **read**(Jarak, Rentang, X);

3.4 Menampilkan Nilai

Agar hasil pelaksanaan algoritma dapat dikomunikasikan atau ditayangkan maka nilai variabel yang telah diproses dalam algoritma dapat ditampilkan. Instruksi untuk menampilkan nilai variabel adalah: **write**(variabel,...);

Contoh penampilan nilai adalah sebagai berikut:

- **write**("nama anda :",Nama);
- **write**("nilai ujian = ", nilai);
- **write**("Jumlah variabel = ", X + Y + Z);

3.5 Ekspresi (Expression)

Ekspresi adalah transformasi data dan peubah dalam bentuk persamaan yang direlasikan oleh **operator** dan **operand**. Operand adalah data, tetapan, peubah, atau hasil dan suatu fungsi, sedangkan operator adalah simbol - simbol yang memiliki fungsi untuk menghubungkan operand sehingga terjadi transformasi. Jenis-jenis operator adalah sebagai berikut:

- **Operator aritmetika:** operator untuk melakukan fungsi aritmetika seperti: + (menjumlah), - (mengurangkan), * (mengalikan), / (membagi).
- **Operator relational:** operator untuk menyatakan relasi atau perbandingan antara dua operand, seperti: > (lebih besar), < (lebih kecil), >= (lebih besar atau sama), <= (lebih kecil atau sama), == (sama), != (tidak sama) atau > <, < >.
- **Operator logik:** operator untuk merelasikan operand secara logis, seperti && (and), (or), dan ! (not).
- **Operator string:** operator untuk memanipulasi string, seperti +

(concatenation), **len** (panjang string), dan **substr** (substring, mencuplik).

Berdasarkan jenis operator yang digunakan maka ada empat macam ekspresi, yaitu *ekspresi aritmetika*, *ekspresi relational*, *ekspresi logik*, dan *ekspresi string*.

- **Ekspresi Aritmetika:** ekspresi yang memuat operator aritmetika, contoh:
 - $T = 5 * (C + 32) / 9;$
 - $Y = 5 * ((a + b) / (c + d) + m / (e * f));$
 - $Gaji = GaPok * (I + JumNak * 0.05 + Lembur * 1.25);$
- **Ekspresi Relational:** ekspresi yang memuat operator relational, contoh:
 - $Nilai_A > Nilai_B$
 - $(A + B) < (C + D)$
 - $(x + 57) \neq (y + 34)$
- **Ekspresi Logik:** ekspresi yang memuat operator logik, contoh:
 - $m = (x > y) \& \& (5 + z)$
 - $n = (!A \& \& !(B \& \& C))$
- **Ekspresi String:** ekspresi dengan operator string, contoh:
 - $Alamat = "Jl. P. Kemerdekaan " + "Km 9 Tamalanrea"$
 - $Hasil = "Saudara:" + Nama + "adalah mahasiswa"$
 - $Tengah = Substr(Kalimat, 5, 10);$

Contoh Algoritma:

1. Susun algoritma yang menghitung pajak pertambahan nilai (ppn) 12.50% dengan meminta harga barang yang dibeli dari pengguna program.

Algoritma PPN

{ menghitung pajak pertambahan nilai 12.50% dari harga barang }

Definisi Variabel

real harga, pajak, total;

Rincian Langkah

```

write ("Masukkan harga barang : ");
read(harga);

pajak ( $- 0.125 * harga$ ;
total = harga + pajak;
write("Harga ", harga, " pajaknya=",pajak);
write("Total = total);
  
```

2. Susun algoritma yang meminta data dasar mahasiswa (mis: Nama, Alamat, e_mail, dan telepon) kemudian menampilkannya kembali secara tersusun.

Algoritma Data_dasar

{membaca dan menampilkan data dasar mahasiswa}

Definisi Variabel:

string nama, alamat, e-mail, telepon;

Rincian Langkah:

{proses input/permintaan data }

```
write ("Masukkan nama Anda: ");
read(nama);
write("Di mana alamatnya: ");
read(alamat);
write("No telepon: ");
read(telepon);
write("Alamat e-mail: ");
read(e_mail);
```

{proses tampilan hasil input }

```
write(nama);
write(alamat, telepon);
write(e_mail);
```

SOAL:

1. Definisikan sebuah record untuk data pegawai yang terdiri atas elemen Nomor_Pegawai (NIP), Nama, TgLahir, Tempat_Lahir, Jen_Kelamin, Agama, Status, Pangkat.
2. Tulis algoritma sederhana untuk membaca dan menampilkan kembali data pegawai yang sesuai dengan struktur record pada Soal 1.
3. Tulis algoritma yang meminta suhu dalam skala celcius dan menampilkan suhu tersebut dalam skala Fahrenheit. ($F = \frac{9}{5} * C + 32$).
4. Tulis algoritma yang membaca tiga data, yaitu nilai tugas, nilai mid, dan nilai final, kemudian menghitung nilai akhir = 20% nilai tugas + 30% nilai mid + 50% nilai final. Tampilkan nilai akhir ini.

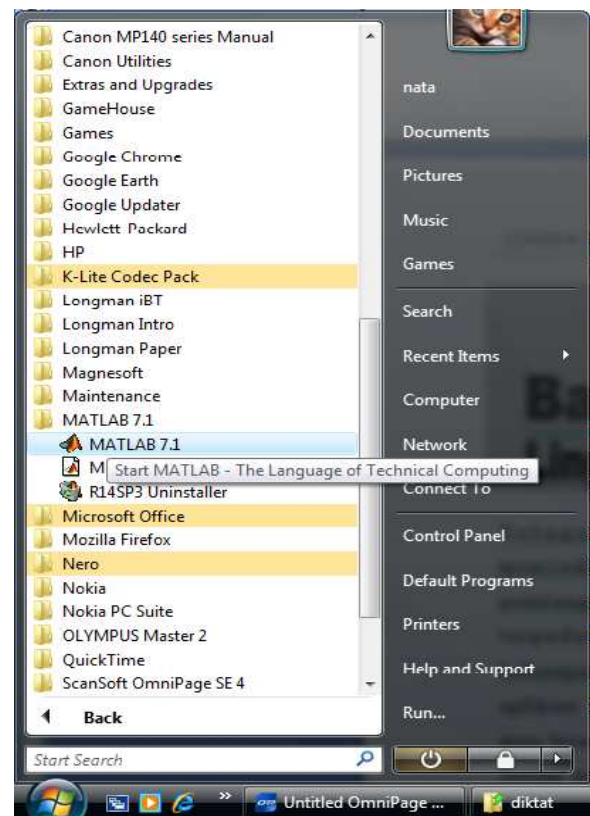
Bab IV

Lingkungan Kerja Matlab



4.1 Form/Window Matlab

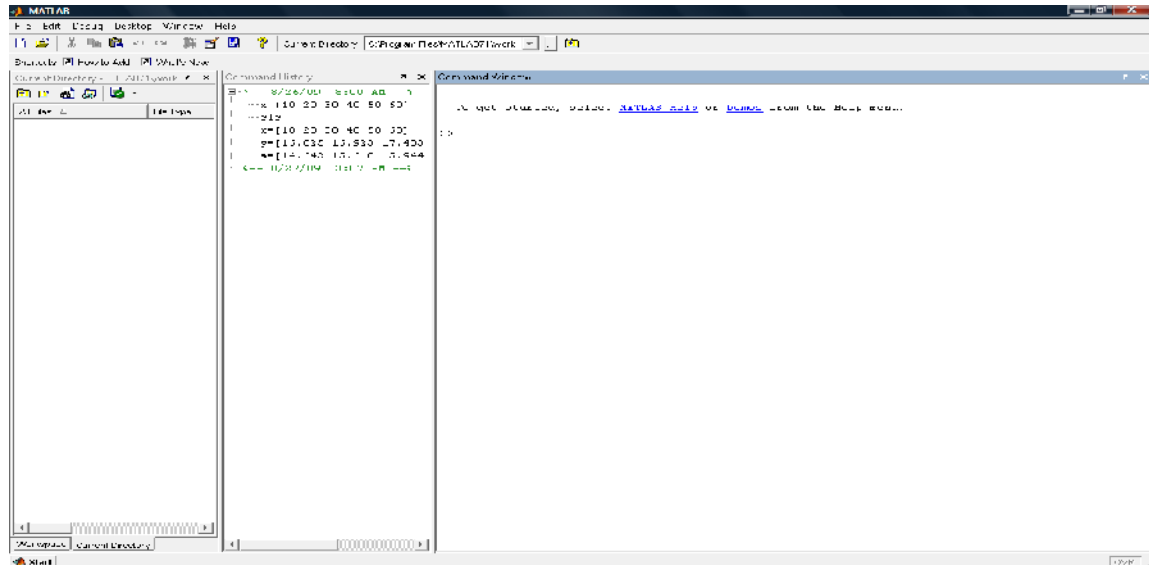
Sebagaimana bahasa pemrograman lainnya, MATLAB juga menyediakan lingkungan kerja terpadu yang sangat mendukung dalam pembangunan aplikasi. Pada setiap versi MATLAB yang terbaru, lingkungan terpadunya akan semakin dilengkapi. Lingkungan terpadu ini terdiri atas beberapa form/window yang memiliki kegunaan masing-masing. Untuk memulai aplikasi Matlab, anda hanya perlu mengklik ikon Matlab pada *Desktop window*, atau bisa juga dengan menu *start* seperti pada aplikasi-aplikasi lainnya.



Setiap pertama kali mulai membuka aplikasi Matlab, anda akan memperoleh beberapa form/window, yang sebenarnya menurut penulis hanya membuat *desktop* anda terlihat penuh.

Matlab akan menyimpan *mode/setting* terakhir lingkungan Kerja yang anda gunakan sebagai *mode/setting* lingkungan kerja pada saat anda membuka aplikasi Matlab di waktu berikutnya.

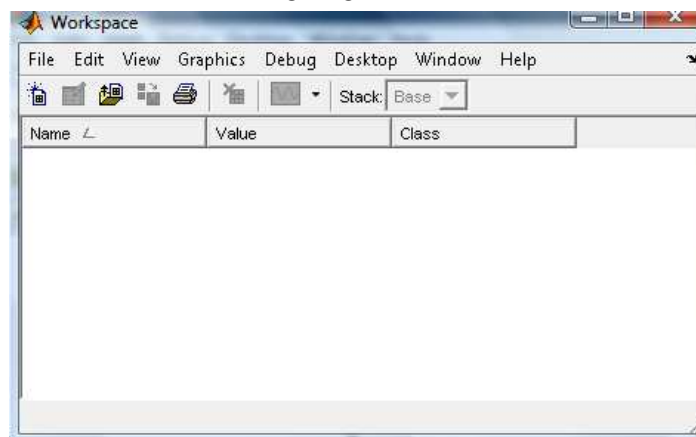
4.1.1. Windows Utama Matlab



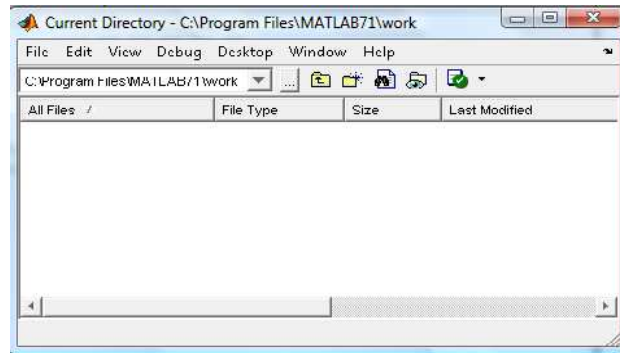
Window ini adalah *window* induk yang melingkupi seluruh lingkungan kerja MATLAB. Pada versi-versi pendahulu, *window* ini secara khusus belum ada namun terintegrasi dengan *Command Window*. Tidak ada fungsi utama yang ditawarkan oleh *window* ini selain sebagai tempat *dock-ing* bagi *form* yang lain.

4.1.2. Workspace Windows

Window ini juga baru diperkenalkan pada versi 6, berfungsi sebagai navigator bagi pemakai dalam penyediaan informasi mengenai variabel yang sedang aktif dalam *workspace* pada saat pemakaian. *Workspace* adalah suatu lingkungan abstrak yang menyimpan seluruh variabel dan perintah yang pernah digunakan selama penggunaan MATLAB berlangsung.

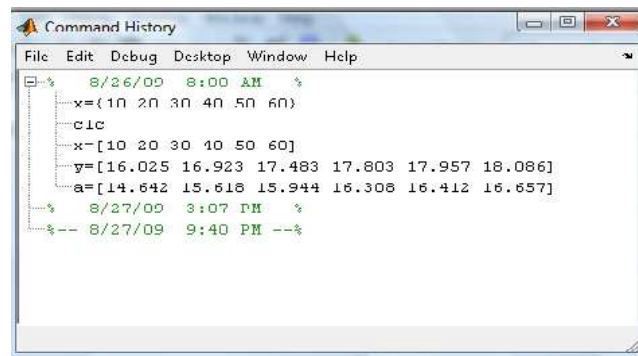


4.1.3. Current Directory Window



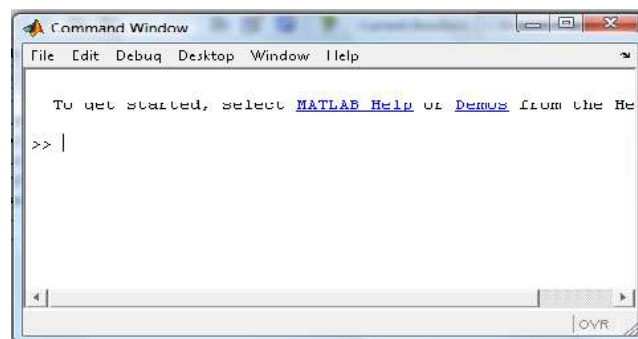
Window ini juga fasilitas yang diperkenalkan pada versi 6. Berfungsi sebagai *browser* direktori aktif, yang hampir sama dengan *window explorer*.

4.1.4. Command History Window



Window ini berfungsi sebagai penyimpan perintah-perintah yang pernah dikerjakan pada suatu *workspace*. Juga barn ada pada Matlab versi 6 keatas.

4.1.5. Command Window



Window ini berfungsi sebagai penerima perintah dari pemakai untuk menjalankan seluruh fungsi-fungsi yang disediakan oleh MATLAB. Pada dasarnya *window* inilah inti dari pemrograman MATLAB yang menjadi media utama satu-satunya bagi kita untuk berinteraksi dengan MATLAB.

4.2. Cara Bekerja Dengan MATLAB

Dalam melakukan pekerjaan pemrograman menggunakan bahasa MATLAB, anda dapat menggunakan salah satu cara yaitu :

Cara #1 : Lanjutkan di Command Window

Zara ini adalah yang paling sering dilakukan oleh pemula, namun masih sulit bagi anda untuk mengevaluasi perintah secara keseluruhan. Biasanya perintah hanya dilakukan baris perbaris.

Untuk membuat program, anda hanya perlu mengetikkan perintah pada prompt Matlab dalam Command Window, misalnya :

```
>> pjl = 5;
```

tekan tombol enter, lalu ketikkan :

```
>> lbr = 10;
```

tekan tombol enter, lalu ketikkan :

```
>> luas = pjl * lbr
```

untuk skrip terakhir sengaja tidak diberikan tanda (;) titik koma, sehingga anda bisa langsung melihat hasil akhir di layar *Command Window*.

Hasil akhir yaitu

```
>> luas =  
50
```

Program telah selesai.

Cara #2 : Menggunakan File M

Cara ini biasanya akan dipilih untuk digunakan oleh *programmer* yang lebih mahir (jangan khawatir dalam beberapa menit kedepan, anda pun akan menjadi salah satu dari kelompok ini (D)). Kelebihan cara ini adalah kemudahan untuk meng-evaluasi perintah secara keseluruhan. Terutama untuk program yang membutuhkan waktu pengerjaan yang cukup lama serta skrip yang cukup

panjang.

Untuk contoh dapat kita gunakan program yang sebelumnya anda kerjakan dengan cara pertama, dengan tahapan sebagai berikut :

1. Pada Command Window, ketikkan:

```
>>edit
```

2. Tekan *enter*, selanjutnya muncul Matlab *Editor* dan anda ketiklah program dibawah berikut :

```
% -----
% Program latihan 1
% Matlab Programing
% Oleh : Wayan Made Komang
% -----

clear all;
clc;

disp(' ----- ' )

disp(' Program Latihan 1 ' )
disp(' ----- ' )

pjpg = 5;
lbr = 10 ;
luas = pjpg * lbr;
disp([' luas -> ' num2str(luas)]);
```

Anda bingung dengan apa yang baru anda ketikkan? Tidak masalah, kata orang pintar "bingung pertanda mulai belajar", pada bab-bab berikutnya akan anda pelajari dan anda. Akan sangat mengerti dengan apa yang anda ketik. Untuk tahap ini anda ikuti dulu oke!

3. Setelah selesai mengetik program diatas, anda simpan di direktori c:\latihanku, dengan nama **latihan01.m**.
4. Anda kembali ke *Command Window*. Agar Matlab dapat mengenali lokasi tempat file **anda tersimpan, pada prompt** Matlab ketiklah direktori c:/latihanku pada prompt Matlab :

```
>>cd c:\latihanku
```

5. Tekan Enter, lalu ketiklah nama file latihan 01 tanpa ekstensi:


```
>>latihan01
```

6. Tekan Enter, selanjutnya program akan dijalankan dan menghasilkan sebagai berikut:

```
-----  
Program Latihan 1  
-----  
Luas -> 100
```

Jika tahap enam telah dicapai, maka program pertama telah sukses anda kerjakan. Selamat datang dan bergabung sebagai programmer Matlab.

Manajemen File dan Direktori

Matlab menggunakan metode *path searching* (pencarian direktori) untuk menemukan file dengan ekstensi M yang mengandung skrip dan fungsi. File M MATLAB terorganisir dengan rapi pada beberapa folder/direktori. Urutan pencarian MATLAB dalam menjalankan perintah pada Command Window secara bertahap adalah sebagai berikut, misalnya ketika diberi perintah 'kubus':

- MATLAB mencoba untuk mengenali apakah 'kubus' adalah variabel, jika ya, selesai. Jika tidak, maka MATLAB berasumsi bahwa 'kubus' adalah sebuah nama file dengan ekstensi M, lanjut ke tahap berikutnya.
- Matlab mencoba untuk mengenali apakah 'kubus' merupakan fungsi bawaan standar, jika ya, eksekusi. Jika tidak, lanjut ke tahap berikutnya.
- Matlab akan mencari file M yang bernama kubus.m pada direktori aktif (*current directory*), jika ditemukan, eksekusi. Jika tidak lanjut ke tahap berikutnya.
- Matlab akan mencari file M yang bernama kubus.m diseluruh direktori yang terdaftar pada daftar pencariannya, jika ditemukan, eksekusi. Jika tidak, Matlab akan menyampaikan pesan sebagai berikut:

```
>> kubus  
??? Undefined function or variabel 'kubus'
```

Jika pesan diatas muncul dihadapan anda, maka kesimpulannya hanya ada 2,

yaitu:

1. Anda salah menulis nama file, atau
2. File anda tidak berada dalam direktori yang diketahui oleh Matlab.

Jika anda yakin nama file yang anda ketikkan benar, maka yang harus anda lakukan juga ada 2 pilihan, yaitu:

1. **Memindahkan direktori aktif ke direktori tempat file anda berada.**

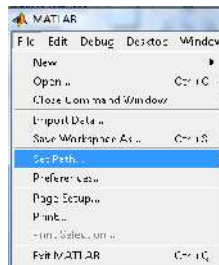
Misalkan direktori tempat anda menyimpan file adalah c:\latihanku. Maka caranya adalah dengan perintah sebagai berikut:

```
>>cd c:\latihanku
```

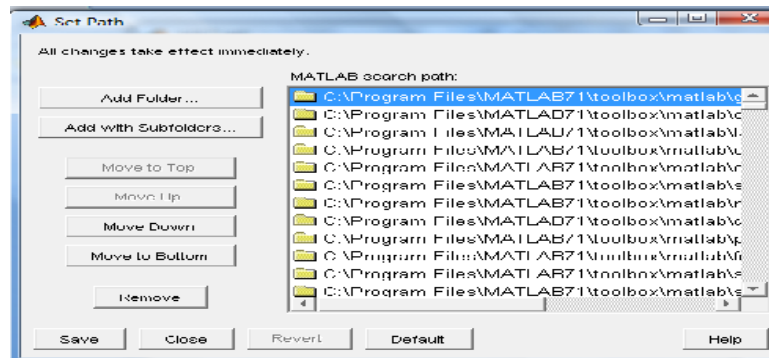
Tekan enter, maka selanjutnya direktori aktif adalah c:\latihanku.

2. **Menambahkan direktori anda ke dalam daftar pencarian direktori Matlab.**

Untuk membuka tool yang mengatur pencarian *path* ini dapat dilakukan dengan cara meng-klik **set path** pada menu **File** sebagai berikut:



Selanjutnya akan muncul dialog set path sebagai berikut:



Setelah melakukan pemilihan folder yang diinginkan dengan cara meng klik tombol **Add Folder**, lalu dilanjutkan dengan meng klik tombol **save** dan diakhiri dengan tombol **Close**. Maka direktori anda telah tersimpan didaftar pencarian direktori Matlab.

Jika anda memilih menggunakan cara pertama , maka setiap anda membuka aplikasi Matlab anda harus melakukannya lgi. Tetapi, jika anda memilih cara kedua, maka anda tidak perlu melakukannya lagi pada kesempatan lainnya, kecuali jika nama direktori anda telah berubah.

Penting

Pemahaman yang benar tentang cara kerja Matlab ketika melakukan eksekusi program ini sangat penting. Dari pengalaman penulis, permasalahan yang paling banyak diajukan pada saat penulis melayani konsultasi program adalah yang berhubungan dengan penempatan file program dan penamaan file yang overlap dengan fungsi standar yang dimiliki Matlab.

SOAL:

1. Buatlah dengan program matlab dari luas ruangan pertemuan yang memiliki panjang 12 meter dan lebar 8 meter. Buat dengan menggunakan command windows langsung dan M-file.
2. Buat dengan model M-file ! Suatu silinder memiliki jari-jari 60 cm, dan tinggi dari silinder tersebut 150 cm. hitung luas selimut silinder dan volume silinder jika nilai $\pi = 3,14$.

Bab V

Konsep Dasar Matlab



5.1 Fungsi Matematika Umum

Fungsi matematika umum dalam MATLAB terdiri atas fungsi trigonometri, fungsi eksponensial, fungsi berkait bilangan kompleks serta fungsi pembulatan dan sisa. Pada sub bagian berikut akan diuraikan fungsi-fungsi tersebut lengkap dengan *syntax*-nya dalam MATLAB. (note: tanda “ ” menunjukkan *syntax*nya dan “x” adalah bilangan konstan).

5.1.1 Fungsi trigonometri

Fungsi dasar trigonometri terdiri atas sinus, cosinus dan tangen. Fungsi tersebut dikembangkan menjadi secan, cosecan dan cotangen. Lebih lanjut fungsi-fungsi trigonometri dapat dikembangkan menjadi bentuk hiperbolik. Operasi yang dapat dilakukan pada fungsi tersebut adalah pencarian nilai dan invers (Arcus disingkat Arc). Dalam MATLAB, setiap fungsi trigonometri mempunyai *syntax* tersendiri. Fungsi trigonometri lengkap dengan *syntax*nya akan diperlihatkan pada uraian berikut:

- Fungsi **sinus**

Sinus $x \rightarrow \sin(x)$: Arc sinus $x \rightarrow \text{asin}(x)$

Sinus hiperbolik $x \rightarrow \sinh(x)$: Arc sinus hiperbolik $x \rightarrow \text{asinh}(x)$

- Fungsi **cosinus**

Cosinus $x \rightarrow \cos(x)$: Arc cosinus $x \rightarrow \text{acos}(x)$

Cosinus hiperbolik $x \rightarrow \cosh(x)$: Arc cosinus hiperbolik $x \rightarrow \text{acosh}(x)$

- Fungsi **Tangen**

Tangen $x \rightarrow \tan(x)$: rc tangen $x \rightarrow \text{atan}(x)$

Tangen hiperbolik $x \rightarrow \tanh(x)$: Arc tangen hiperbolik $x \rightarrow \text{atanh}(x)$

- Fungsi **Secan**

Secan $x \rightarrow \sec(x)$: Arc secan $x \rightarrow \text{asec}(x)$

Secan hiperbolik $x \rightarrow \operatorname{sech}(x)$: Arc secan hiperbolik $\rightarrow \operatorname{asech}(x)$

- Fungsi **Cosecan**

Cosecan $x \rightarrow \operatorname{csc}(x)$: Arc cosecan $x \rightarrow \operatorname{acsc}(x)$

Cosecan hiperbolik $x \rightarrow \operatorname{csch}(x)$: Arc cosecan hiperbolik $x \rightarrow \operatorname{acsch}(x)$

- Fungsi **Cotangen**

Cotangen $x \rightarrow \cot(x)$: Arc Cotangen $x \rightarrow \operatorname{acot}(x)$

Cotangen hiperbolik $x \rightarrow \operatorname{coth}(x)$: Arc cotan hiperbolik $x \rightarrow \operatorname{acoth}(x)$

5.1.2 Fungsi eksponensial

Fungsi eksponensial secara matematis mempunyai beberapa bentuk. Pada uraian berikut akan diperlihatkan beberapa fungsi eksponensial lengkap dengan *syntax*-nya.

- **Eksponensial** : $e^x \rightarrow \exp(x)$, fungsi ini digunakan untuk mencari nilai e^x , dengan e adalah bilangan natural ($e=2,718281824459\dots$).
- **Logaritma berbasis bilangan natural (e)**: ${}^e\log x$ atau $\ln x \rightarrow \log(x)$, fungsi ini digunakan untuk mencari nilai logaritma berbasis e^x .
- **Logaritma berbasis bilangan konstan**, misal $y: {}^y\log x \rightarrow \log_y(x)$, fungsi ini digunakan untuk mencari nilai logaritma berbasis y . Jadi, logaritma berbasis 10 tidak seperti biasanya yaitu ditulis dengan $\log x$ saja, tetapi dianggap sebagai bilangan y . Jadi, dalam MATLAB: ${}^{10}\log x$ ditulis $\log_{10}(x)$.
- **Akar pangkat dua**: $\sqrt{x} \rightarrow \operatorname{sqrt}(x)$, fungsi ini berguna untuk mencari akar pangkat dua dari x .

5.1.3 Fungsi yang berkait bilangan kompleks

Fungsi yang berkait bilangan kompleks, dalam MATLAB dituliskan dengan penambahan "i" atau "j" dibelakang bilangan pokok. Bilangan kompleks dapat dinyatakan dalam bentuk real atau sebaliknya. Pada uraian berikut diperlihatkan beberapa fungsi eksponensial lengkap dengan *syntax*-nya.

- **Nilai mutlak**: $|x| \rightarrow \operatorname{abs}(x)$, digunakan untuk mencari nilai mutlak bilangan x atau $-x$
- **Nilai riil** $\rightarrow \operatorname{real}(x)$, digunakan untuk mengambil nilai real dari bilangan kompleks
 $ix \text{ atau } jx \rightarrow \operatorname{imag}(x)$
- **Nilai imajiner** $\rightarrow \operatorname{imag}(x)$, digunakan untuk mengambil nilai imajiner dari

bilangan kompleks

5.1.4 Fungsi pembulatan dan sisa

Fungsi pembulatan dan sisa dalam MATLAB, digunakan untuk menuliskan bilangan riil dalam bentuk bilangan bulat positif atau negatif.

- **Pembulatan menuju integer terdekat:** $\rightarrow \text{round}(x)$, fungsi ini digunakan untuk membulatkan x ke nilai integer terdekat.

Misal:

$\text{round}(3.84)$ menghasilkan 4 dan $\text{round}(-3.84)$ menghasilkan -4

$\text{round}(0.52)$ menghasilkan 1 dan $\text{round}(-0.52)$ menghasilkan -1

- **Pendekatan menuju nol:** $\rightarrow \text{fix}(x)$, fungsi ini digunakan untuk membulatkan x ke nilai yang lebih dekat 0.

Misal:

$\text{fix}(3.84)$ menghasilkan 3 dan $\text{fix}(-3.84)$ menghasilkan -3

$\text{fix}(5.32)$ menghasilkan 5 dan $\text{fix}(-5.32)$ menghasilkan -5

$\text{fix}(0.52)$ menghasilkan 0 dan $\text{fix}(-0.52)$ menghasilkan 0

- **Pembulatan menuju $-\infty$:** $\rightarrow \text{floor}(x)$, fungsi ini digunakan untuk membulatkan x ke nilai yang lebih mendekati $-\infty$.

Misal:

$\text{floor}(3.84)$ menghasilkan 3 dan $\text{floor}(-3.84)$ menghasilkan -4

$\text{floor}(5.32)$ menghasilkan 5 dan $\text{floor}(-5.32)$ menghasilkan -5

$\text{floor}(0.52)$ menghasilkan 0 dan $\text{floor}(-0.52)$ menghasilkan -1

- **Pembulatan menuju $+\infty$:** $\rightarrow \text{ceil}(x)$, fungsi ini digunakan untuk membulatkan x ke nilai yang lebih dekat $+\infty$.

Misal:

$\text{ceil}(3.84)$ menghasilkan 4 dan $\text{ceil}(-3.84)$ menghasilkan -3

$\text{ceil}(5.32)$ menghasilkan 5 dan $\text{ceil}(-5.32)$ menghasilkan -5

$\text{ceil}(0.52)$ menghasilkan 1 dan $\text{ceil}(-0.52)$ menghasilkan 0

- **Sisa setelah pembagian:** $\rightarrow \text{rem}(x, y)$, fungsi ini digunakan untuk mengambil sisa dari x/y dengan tanda yang sama dengan x .

Misal:

$\text{rem}(15.2)$ menghasilkan 1 dan $\text{rem}(-15.2)$ menghasilkan -1

$\text{rem}(-13.5)$ menghasilkan -3 dan $\text{rem}(13.5)$ menghasilkan 3

5.2 Format Penampilan Angka

Format penampilan angka dalam MATLAB, digunakan untuk mengatur tampilan hasil perhitungan pada *Command Window*. Jika hasil perhitungan berupa bilangan bulat, MATLAB akan menampilkan sebagai bilangan bulat, sedangkan bilangan riil, ditampilkan dengan empat bilangan desimal. Tampilan ini dapat diatur dengan format penampilan angka dalam MATLAB. Misal: $y = 100/3$, dalam keadaan biasa (tanpa pengaturan format) y akan ditampilkan dengan format 33.3333. Berikut ini diuraikan beberapa perintah untuk mengatur penampilan angka pada *Command Window* lengkap dengan *syntax*-nya.

- format short: menghasilkan 33.3333
- format long: menghasilkan 33.333333333334
- format short e: menghasilkan 3.3333e+001
- format long e: menghasilkan 3.333333333334e+001
- format short g: menghasilkan 33.333
- format long g: menghasilkan 33.333333333333
- format hex: menghasilkan 4040aaaaaaaaab
- format bank: menghasilkan 33.33
- format +: menghasilkan +
- format rat: menghasilkan 100/3

Penting untuk diketahui bahwa format penampilan angka dalam MATLAB tidak mengubah representasi internal dari suatu bilangan.

5.3 Array dan Matriks

Array adalah kumpulan data-data scalar yang dinyatakan dalam bentuk baris, kolom dan gabungan antar keduanya. Kumpulan data dengan deret yang tidak teratur mengharuskan pemakai untuk menuliskan data satu per satu. Kumpulan data dengan deret yang teratur dapat diekspresikan dalam bentuk array, sehingga memungkinkan pemakai untuk tidak menuliskannya satu per satu.

Matriks adalah *array* yang dibangun dari kumpulan persamaan linear. Operasi matriks tidak seperti pada *array* biasa, melainkan sistem operasi aljabar matriks. Pada pembahasan selanjutnya, matriks diuraikan dengan *array* data yang teratur, karena akan

sering dipergunakan pada pemrograman tingkat lanjut.

5.3.1 Pembentukan array dan matriks

Array dan matriks ditampilkan dalam bentuk yang sama tetapi representasi internalnya berbeda. Berikut ini adalah perintah untuk membentuk *array*.

- **x=m: n**, membuat baris dengan elemen awal m , kenaikan 1 dan elemen akhir n atau sebelum n .

Misal:

Perintah $x=1:10$ akan membuat barisan dengan elemen dimulai dari 1, bertambah 1 dan berakhir pada 10.

- **x=m: k: n**, membuat baris dengan elemen awal m , kenaikan k dan elemen akhir n atau sebelum n .

Misal:

Perintah $x=1:2:20$ akan membuat barisan dengan elemen dimulai dari 1, bertambah 2 dan berakhir pada 19.

- **x=linspace (m, n, k)**, membuat baris dengan elemen awal m dan elemen akhir, dengan jumlah elemen sebanyak k .

Misal:

Perintah $x=\text{linspace}(1, 30, 10)$ akan menampilkan baris dengan elemen awal bernilai 1 dan elemen akhir bernilai 30 dengan jumlah elemen sebanyak 10.

- **x=logspace (m, n, k)**, membuat baris dengan elemen awal m dan elemen akhir n dengan jumlah elemen sebanyak k dalam skala logaritma.

Misal:

Perintah $x=\text{logspace}(1, 30, 10)$ akan menampilkan baris dengan elemen awal bernilai 10 dan elemen akhir bernilai 10 dengan jumlah elemen 10.

- **x=ones (m)**, membuat *array* segiempat ukuran $m \times m$ dengan semua elemennya bernilai 1.

Misal:

Perintah $x=\text{ones}(5)$ akan menampilkan *array* segiempat ukuran 5×5 dengan semua elemennya bernilai 1

- **x=ones (m, n)**, membuat *array* segiempat ukuran $m \times n$ dengan semua elemennya bernilai

1

Misal:

Perintah `x=ones (2, 5)` akan menampilkan *array* segiempat ukuran 2 x 5 dengan semua elemennya bernilai 1.

- **`x=zeros (m)`** , membuat *array* segiempat ukuran $m \times m$ dengan semua elemennya bernilai 0.

Misal:

Perintah `x=zeros (5)` akan menampilkan *array* segiempat ukuran 5 x 5 dengan semua elemennya bernilai 0.

- **`x=zeros (m, n)`** , membuat *array* segiempat ukuran $m \times n$ dengan semua elemennya bernilai 0.

Misal:

Perintah `x=zeros (2, 5)` akan menampilkan *array* segiempat ukuran 2 x 5 dengan semua elemennya bernilai 0.

- **`x=rand (m, n)`** , membuat *array* segiempat ukuran $m \times n$ dengan elemen-elemennya berupa bilangan random yang terdistribusi uniform dengan interval 0,0 sampai 1,0.

Misal:

Perintah `x=rand (2, 5)` akan menampilkan *array* segiempat ukuran 2 x 5 dengan elemen-elemennya berupa bilangan random yang terdistribusi uniform dengan interval 0,0 sampai 1,0.

- **`x=randn (m, n)`** , membuat *array* segiempat ukuran $m \times n$ dengan elemen-elemennya berupa bilangan random yang terdistribusi normal dengan *mean* = 0 dan *variasi* = 1.

Misal:

Perintah `x=randn (2, 5)` akan menampilkan *array* segiempat ukuran 2 x 5 dengan elemen-elemennya berupa bilangan random yang terdistribusi normal dengan *mean* = 0 dan *variasi* = 1.

Array dapat dinyatakan dalam bentuk matriks dengan memasukkan semua perintah pada ruas kanan ke dalam kurung komutasi. Misal: *array* `x=m: n` dapat dinyatakan dalam bentuk matriks dengan perintah `x= [m: n]` .

5.3.2. Operasi array

Operasi *array* merupakan operasi skalar terhadap elemen-elemennya. Misal diketahui suatu data:

$a = [a_1, a_2, a_3, \dots, a_n]$; $b = [b_1, b_2, b_3, \dots, b_n]$; $c = c$, operasi *array* yang dapat dilakukan adalah:

- Penambahan skalar: $a+c = [a_1+c, a_2+c, a_3+c, \dots, a_n+c]$
- Perkalian skalar: $a*c = [a_1*c, a_2*c, a_3*c, \dots, a_n*c]$
- Penambahan array: $a+b = [a_1+b_1, a_2+b_2, a_3+b_3, \dots, a_n+b_n]$
- Perkalian array: $a.*b = [a_1*b_1, a_2*b_2, a_3*b_3, \dots, a_n*b_n]$
- Pembagian kanan array: $a./b = [a_1/b_1, a_2/b_2, a_3/b_3, \dots, a_n/b_n]$
- Pembagian kiri array: $a.\backslash b = [a_1/b_1, a_2/b_2, a_3/b_3, \dots, a_n/b_n]$
- Pemangkatan array-skalar: $a.^c = [a_1^c, a_2^c, a_3^c, \dots, a_n^c]$
- Pemangkatan skalar-array: $c.^a = [c^{a_1}, c^{a_2}, c^{a_3}, \dots, c^{a_n}]$
- Pemangkatan array-array: $a.^b = [a_1^{b_1}, a_2^{b_2}, a_3^{b_3}, \dots, a_n^{b_n}]$

5.3.3 Operasi matriks

Operasi pada matriks dilakukan dengan menggunakan prinsip aljabar matriks. Berikut ini diuraikan beberapa operasi matriks.

- Transpose matriks: $X = A \Rightarrow A' \rightarrow X=A'$
- Penjumlahan matriks: $X = A \Rightarrow A' \rightarrow X=A+B.$
- Pengurangan matriks: $X = A \Rightarrow A' \rightarrow X=A-B.$
- Perkalian matriks: $X = A \Rightarrow A' \rightarrow X=A*B.$

Operasi matriks lainnya seperti invers, determinan dan sebagainya dapat dilakukan jika syarat-syarat aljabar matriks terpenuhi. Bentuk seperti ini sering digunakan dalam analisis numerik untuk fisika komputasi.

5.3.4 Manipulasi *array* dan matriks

Manipulasi terhadap *array* dan matriks dapat mereduksi kerumitan metode komputasi. Misal dalam perhitungan numerik diketahui *array* data dengan variabel induk A yang terdiri atas gabungan antara baris dan kolom. *Array* tersebut dapat dimodifikasi dengan perintah pengalamatan.

- $A(i, j)$, mengalami sub *array* A dengan indeks baris i dan kolom j .

- $A(i, :)$, mengalami sub *array* A pada semua kolom j .
- $A(:, j)$, mengalami sub *array* A dengan semua baris pada kolom j .
- $A(:)$, mengalami sub *array* A dengan semua baris pada semua kolom.

Pengalamatan yang dilakukan terhadap elemen *array* dapat digunakan untuk pangalamatan elemen matriks dengan ketentuan *array* tersebut sebelumnya telah dinyatakan dalam bentuk matriks.

5.4 Konsep Dasar Kontrol Program

Perhitungan komputasi dengan deret teratur dalam MATLAB dapat dilakukan dengan statement tertentu. Statement tersebut untuk menyatakan kondisi, pemberhentian operasi dan iterasi. Statement-statement ini sering dikenal dengan nama kontrol program.

Kontrol program sangat berguna, karena memungkinkan perhitungan sebelum kondisi tertentu mempengaruhi perhitungan selanjutnya dalam satu kesatuan program. Kontrol program mempunyai fungsi dan keunggulan tersendiri bergantung pada kebutuhan pemrograman.

5.4.1 Statement if

Statement *if* akan mengeksekusi sekumpulan instruksi apabila suatu kondisi yang diisyaratkan bernilai benar. Statement *if* diakhiri dengan *end*. Jika ada kondisi yang berlawanan (*false*), statement *if* dapat diikuti dengan statement *else* atau *elseif*.

Contoh 1:

```
x=input('Masukan bilangan bulat x=')
if rem(x,2)==0
    x=' == GENAP == '
else
    x=' == GANJIL == '
end
```

Contoh 1 akan mengidentifikasi sifat bilangan X , apakah termasuk bilangan ganjil atau genap.

5.4.2 Statement switch

Statement *switch* akan mengeksekusi sekumpulan instruksi didasarkan pada nilai dari suatu ekspresi atau variabel. Statement *switch* diikuti dengan *case*

dan otherwise untuk menunjukan suatu *group* dalam *looping*. Statement ini diakhiri dengan end.

Contoh 2:

```
clear;
X=input('masukkan bilangan bulat X=');
Y=input('masukkan pembagi Y=');
Sisa=rem(X,Y)
switch Sisa
case 10
    NilaiSisa='A'
case 11
    Nilaisisa='B'
case 12
    Nilaisisa='C'
case 13
    Nilaisisa='D'
case 14
    Nilaisisa='E'
otherwise
    NilaiSisa=int2str(sisa)% konversi integer ke string end;
```

Contoh 2 mengambil sisa pembagi dari X/Y kemudian dikelompokkan dalam bentuk abjad. Jika program tersebut dijalankan, pada *Command Window* akan tampil 6 kemungkinan, yaitu:

- Jika Sisa=10, maka Nilaisisa=A
- Jika Sisa=11, maka Nilaisisa=B
- Jika Sisa=12, maka Nilaisisa=C
- Jika Sisa=13, maka Nilaisisa=D
- Jika Sisa=14, maka Nilaisisa=E
- Jika Sisa=bilangan lain, Nilaisisa=Sisa

5.4.3 Statement for

Statement `for` digunakan untuk mengulang sekumpulan instruksi hingga n kali (notasi n adalah bilangan integer yang telah diidentifikasi sebelumnya). Statement `for` diakhiri dengan `end`.

Contoh 3:

```
X=rand(50,1)
Xmax=realmin
for i=1:50
if X(i)>Xmax
Xmax=X(i)
end;
end;
```

Contoh 3 memperagakan cara mencari nilai terbesar dari *array* X . Nilai tersebut akan disimpan dalam variabel X_{\max} .

5.4.4 Statement while

Statement `while` mengerjakan sekelompok perintah yang diulang secara tidak terbatas. Statement `while` diakhiri dengan `end`. Perintah antara *loop* `while` dan `end` dieksekusi berulang kali selama semua elemen dalam ekspresi bernilai benar.

Contoh 4:

```
hitungan=0; X=1;
while (1+X)>1
X=X/2
hitungan=hitungan+1
end
```

Contoh 4 memperagakan cara menghitung nilai terkecil yang dapat ditambah pada 1 sedemikian hingga hasilnya lebih besar dari dengan menggunakan presisi hingga. Selama $(1 + X) > 1$ benar, perintah yang terdapat dalam *loop* `while` akan dikerjakan terus menerus. Karena X terus menerus dibagi 2, maka X semakin kecil hingga didapat suatu kondisi yang unik, yaitu penambahan X pada 1 tidak mendapatkan hasil yang lebih besar dari 1 menurut hitungan komputasi. Jika kondisi tersebut terpenuhi, maka *loop* `while` berhenti bekerja atau kondisi $(1+X) > 1$ bernilai salah. Kondisi

tersebut dicapai pada hitungan ke-53.

5.4.5 Statement break

Statement break digunakan untuk keluar lebih awal dari suatu *loop for* dan *while* jika kondisi yang diinginkan sudah tercapai. Melanjutkan contoh sebelumnya untuk mencari nilai terkecil yang tidak merubah nilai 1 jika ditambah X, persoalan tersebut dikerjakan pada *loop for* dan *if* dengan urutan 1 sampai 1000. sebelumnya sudah diketahui bahwa kondisi yang diinginkan akan tercapai pada hitungan ke-53, tetapi komputer tidak berhenti karena *loop for* menginstruksikan perhitungan sampai pada step ke-1000. Statement break digunakan untuk menghentikan *looping for*.

Contoh 5:

X=1

for hitungan=1:1000

X=X/2;

if (1+X)<=1

X = X + 2

hitungan=hitungan

break

end

end

Dalam kasus ini hitungan hanya melompati struktur yang ditempati. Jadi, pada *loop* bertingkat statement break harus dibuat tersendiri.

Bab VI

Teknik Dasar manipulasi Data



6.1 Membangun Data

Dalam beberapa kasus program sering ditemui penggunaan data inisial dalam bentuk matrik atau array, misalnya matrik nol, matrik identitas dan lain-lain.

Secara sederhana MATLAB menyediakan beberapa teknik untuk membangun data dengan cepat, sebagai berikut:

- **Membangun data dengan elemen yang telah ditentukan**

Misalnya anda akan membangun data x dengan nilai yang telah diketahui, maka cara penulisannya sebagai berikut:

```
>> x = [2 1 6]
x =
     2     1     6
```

Untuk data vektor baris

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

Untuk data vektor kolom

```
>> x = [2 4 6;1 1 5]
x =
     2     4     6
     1     1     5
```

Untuk data bentuk matrik

- **Membangun data dengan batas awal dan batas akhir**

Misalnya anda ingin membuat data sudut dari sudut 30 derajat samapi 35 derajat, maka cara penulisannya adalah sebagai berikut:

```
>> sdt= [30:35]

sdt =

    30    31    32    33    34    35
```

- **Membangun data dengan batas awal, increment dan batas akhir**

Misalnya anda ingin membuat data sudut dari sudut 30 derajat samapi 90 derajat dengan pertambahan 10, maka cara penulisannya adalah sebagai berikut:

```
>> sdt= [30:10:90]

sdt =

    30    40    50    60    70    80    90
```

- **Membangun data dengan batas awal dan batas akhir, tetapi jumlah data ditentukan.**

Misalnya anda ingin membuat data 5 buah sudut dalam interval sudut 30 derajat sampai 90 derajat, maka cara penulisannya adalah sebagai berikut:

```
>> sdt= linspace(30,90,5)

sdt =

    30    45    60    75    90
```

- **Membangun data logaritma dengan batas awal dan batas akhir, tetapi jumlah data ditentukan.**

Misalnya anda ingin membuat data 5 nilai yang berada dalam interval 10^2 dan 10^4 , maka caranya adalah sebagai berikut:

```
>> x= logspace (2,4,5)

x =

1.0e+004 *

    0.0100    0.0316    0.1000    0.3162    1.0000
```


- **Membangun data menggunakan standar matrik MATLAB.**

Cara membuat data matrik dengan semua elemen bernilai 1:

```
>> x=ones (3,4)

x =

     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Cara membuat data matrik dengan semua elemen bernilai 0:

```
>> x=zeros (3,4)

x =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Membuat data matrik identitas, caranya sebagai berikut:

```
>> x=eye (4,4)

x =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

- **Membangun data Random**

Data random sangat penting digunakan dalam pemrograman, khususnya bidang pemodelan matematika. MATLAB menyediakan cara cepat untuk membangkitkan data random sebagai berikut:

```
>> x=rand (4,4)

x =

    0.9501    0.8913    0.6214    0.9218
    0.2151    0.7521    0.4447    0.7382
    0.6060    0.4565    0.6154    0.1700
    0.4949    0.1190    0.7081    0.4183
```

Terlihat bahwa data random yang dihasilkan berada dalam interval 0 dan 1. Lalu bagaimana caranya untuk membangkitkan data random dengan interval lainnya, misalnya interval 5 dan 7?

Berikut ini adalah sintak membangkitkan data random dengan interval:

```
Variabel = (akhir-(rand()*(akhir-awal)))
```

Contoh pemakaian dalam program adalah sebagai berikut:

```
>> a=(7-(rand(4)*(7-5)));

a =

    5.1291    6.8843    6.7232    6.4556
    5.1662    6.2943    6.5345    6.5024
    6.1795    5.3737    6.6326    6.9695
    5.2127    6.5803    5.7924    5.5064
```

6.2 Orientasi dan Augmentasi data

Merubah orientasi dan menempelkan data (*data augmented*) sangat umum digunakan dalam program. Pada bahasa pemrograman yang lain mungkin hal tersebut menjadi gampang-gampang sulit. Tetapi tentu saja pada MATLAB hal tersebut menjadi sangat-sangat mudah.

- Mengubah Orientasi data dengan Transpos

```
>> a=[2 3 6; 6 4 7]

a =

     2     3     6
     6     4     7

>> a=a'

a =

     2     6
     3     4
     6     7
```

- Menempelkan data pada baris

```
>> a=[2 3 6; 6 4 7]

a =

     2     3     6
     6     4     7

>> b=[2 2 2]

b =

     2     2     2

>> aug=[a;b]

aug =

     2     3     6
     6     4     7
     2     2     2
```

- Menempelkan data pada kolom

```
>> a = [2 3 5; 6 4 7]

a =

     2     3     5
     6     4     7

>> b = [1;2]

b =

     1
     2

>> aug = [a b]

aug =

     2     3     5     1
     6     4     7     2
```

6.3 Pengurutan Data

Teknik pengurutan data sangat sering digunakan dalam program pengolahan data. MATLAB menyediakan fungsi khusus yaitu **sort** untuk melakukan pengurutan. Menggunakan **sort** dapat dengan dua cara. Cara pertama digunakan untuk mengurutkan data pada arah kolom, sintaknya sebagai berikut:

```
var2=sort (var1,1)
```

var1 adalah matrik atau vektor yang akan diurutkan. Berikut adalah cara menggunakannya dalam program:

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

a =

     2     3     1
     9     6     7
     7     9     5

>> b=sort(a,1)

b =

     2     3     1
     7     6     5
     9     9     7
```

Cara kedua digunakan untuk mengurutkan data pada arah baris, sintaknya sebagai berikut:

```
var2= sort(var1,2)
```

Var1 adalah matrik atau vektor yang akan diurutkan. Berikut adalah cara menggunakannya dalam program:

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

a =

     2     3     1
     9     6     7
     7     9     5

>> b=sort (a,2)

b =

     1     2     3
     6     7     9
     5     7     9
```

6.4 Menyeleksi Data

Melakukan pemrograman untuk pengolahan data, pasti tidak luput dari seleksi data. Menyeleksi data berarti menggunakan sebagian data dari sebuah data yang lengkap tanpa merusak ukuran maupun nilai data tersebut. Untuk melakukan seleksi data, anda dapat mempergunakan ekspresi matematika =, >, >=, < dan <=.

Cara menyeleksi data untuk mengambil nilai elemen dari sebuah matrik, adalah sebagai berikut:

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

a =

     2     3     1
     9     6     7
     7     9     5

>> b=(a>7) .*a

b =

     0     0     0
     9     0     0
     0     9     0
```

Program diatas maksudnya adalah menyeleksi elemen pada data a, dengan syarat elemen tersebut lebih besar dari 7, lalu hasilnya disimpan pada variable b.

Cara kerja seleksi data adalah membangkitkan matrik bernilai 0 dan 1, sesuai kondisi yang diberikan, jika memenuhi kondisi, elemen matrik bernilai 1, jika tidak, maka elemen matrik bernilai 0.

```
>> c = (a ~= 7)
```

```
C =
```

```
1    1    1
```

```
1    1    0
```

```
0    1    1
```

Maka untuk mendapatkan nilai yang terseleksi, kalikan hasil diatas dengan data asal. Gunakan perkalian elemen!!

```
>> c = (a ~= 7) * a
```

```
c =
```

```
2    3    1
```

```
9    6    0
```

```
0    9    5
```

Banyak kemudahan yang ditawarkan MATLAB, namun efektifitas pemrogramannya hanya akan tercapai jika kita mampu menerapkan metode-metode manipulasi data dalam skrip program. Umumnya programmer yang memiliki dasar pemrograman procedural mengabaikan hal ini, sehingga efektifitas penulisan skrip dan kecepatan proses runtime tidak diperoleh secara maksimal.

Dengan menggunakan teknik-teknik manipulasi diatas, sebisa mungkin hindari pemakaian *control flow* (perulangan dan kondisional) yang melibatkan operasi elemen-elemen data seperti pada pemrograman konvensional. Karena walaupun MATLAB menyediakan fasilitas untuk itu, MATLAB tidak didesain untuk hal tersebut. Akibatnya waktu yang digunakan MATLAB untuk menjalankan program anda akan cukup lama.

Bab VII

Menggambar Grafik Dalam Matlab



7.1 Perintah Dasar menggambar Grafik

Program MATLAB mempunyai fasilitas menggambar hasil perhitungan komputasi secara grafis. Instruksi yang bisa digunakan untuk menggambar grafik, sebagian diantaranya telah dibahas pada tutorial Bab 1. Pada bab ini akan dibahas berbagai Instruksi untuk menggambar grafik dalam MATLAB.

Elemen dasar yang dibutuhkan untuk menggambar grafik adalah data dalam bentuk *array*. Misal, MATLAB diberi perintah membuat array dalam bentuk sudut dari 0 sampai 2 sebanyak 30 data. Dalam MATLAB, *array* tersebut dinyatakan dengan variabel Theta untuk menentukan nilai $x = \sin$. Persoalan seperti ini dapat dibuat dalam MATLAB dengan *listing* program:

```
Theta=linspace(0,2*pi,30)
X=sin(Theta)
```

Program ini dapat dilengkapi dengan *syntax* untuk menggambar grafik fungsi sinus. Instruksi yang boleh digunakan adalah:

- Plot (Theta): menggambar grafik terhadap indeksinya.
- Plot (Theta, x) : menggambar grafik x terhadap , dengan syarat panjang *array* data dan x sama.
- Plot (Theta, x, bt-cing). menggambar grafik x, terhadap , dengan karakter berupa string. Karakter *string* boleh berupa: **warna**, **tipe titik data** dan tipe **garis penghubung titik data** dari grafik yang *di-plot*.

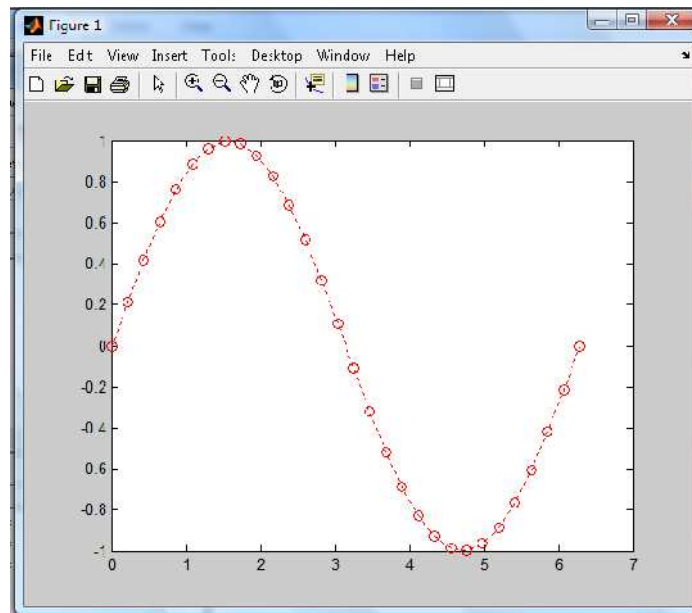
Sebagai contoh, akan digambar fungsi sinus 1 periode terhadap data dengan karakteristik: warna: merah ($r=red$), tipe titik data: lingkaran (o) dan tipe garis penghubung data: titik-titik (:). Program sederhana untuk problem ini adalah:

```
Theta=linspace(0,2*pi,30)
```

```
X=sin(Theta)
```

```
Plot(Theta,x,'ro:')
```

Grafik yang dihasilkan dipaparkan pada Gambar dibawah:



Gambar 7.1 Contoh plot grafik sinus

Karakteristik grafik pada Gambar 5.1 dapat dimodifikasi dengan *string* seperti pada Tabel dibawah:

Tabel 7.1 Karakter *string* pada grafik

Warna	Simbol	Titik data	Simbol	Garis penghubung data	Simbol
merah	r	titik	.	garis lurus	-
biru	b	lingkaran	o	titik-titik	:
kuning	y	tanda silang	x	garis-titik	-.
hijau	g	tanda plus	+	garis putus-putus	--
ungu	m	tanda bintang	*		
putih	w	tanda kotak	s		
hitam	k	tanda berlian	d		
biru muda	c	tanda pentagram	p		
		tanda heksagram	h		

7.2 Memberi Judul dan Label Pada Grafik

Grafik pada Gambar 5.1 belum mempunyai judul pada bagian atas dan label pada setiap sumbu. Perintah untuk memberi judul dan label adalah:

- `title('text','fontsize',size,'fontname','name')`
- `xlabel('text','fontsize',size,'fontname','name')`
- `ylabel('text','fontsize',size,'fontname','name')`
- `zlabel('text','fontsize',size,'fontname','name')`

Dengan text adalah nama judul atau label, fontsize adalah perintah pengaturan ukuran huruf, size adalah ukuran huruf, fontname adalah perintah pengaturan jenis huruf, name adalah jenis huruf. Sebagai contoh, grafik pada Gambar 5.1 dapat dilengkapi dengan instruksi berikut:

```
Theta=linspace(0,2*pi,30)
```

```
x=sin(Theta)
```

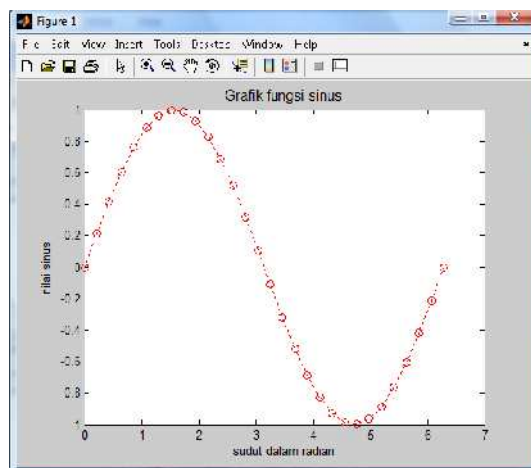
```
plot(Theta,x,'ro:')
```

```
title('Grafik fungsi sinus','fontsize',12,'fontname','Arial')
```

```
xlabel('Sudut dalam radian','fontsize',10,'fontname','Arial')
```

```
ylabel('Nilai sinus','fontsize',10,'fontname','Arial')
```

Dengan menjalankan program ini, akan diperoleh sebuah grafik lengkap dengan keterangannya seperti pada Gambar dibawah:



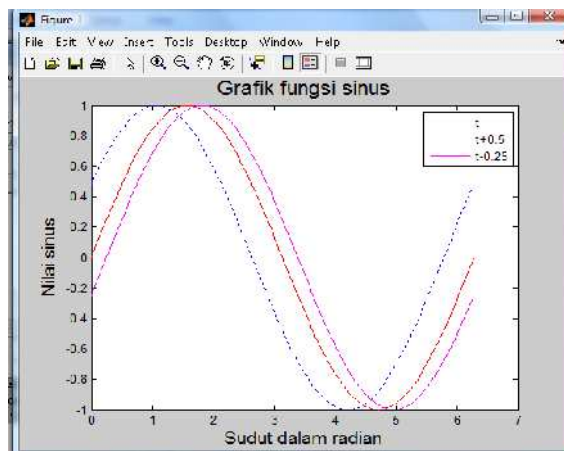
Gambar 7.2 Grafik fungsi sinus 1 periode lengkap dengan judul dan label

7.3 Memberi Grid dan Legend

Perintah grid dan legend digunakan untuk memberi *grid* dan *legend* pada grafik. Misal, diketahui fungsi sinus 1 periode $y_1 = \sin t$. Fungsi tersebut mengalami pergeseran sejauh $y_2 = \sin(t+1/2)$ dan $y_3 = \sin(t+1/4)$. Ketiganya dapat digambarkan dalam satu grafik yang dilengkapi *grid* dan *legend*.

```
clear;
t=0:pi/100:2*pi
y1=sin(t)
y2=sin(t+0.5)
y3=sin(t-0.25)
plot(t,y1,'r-',t,y2,'b:',t,y3,'m-')
title('Grafik fungsi
sinus','fontsize',16,'fontname','Arial')
xlabel('Sudut dalam
radian','fontsize',14,'fontname','Arial')
ylabel('Nilai
sinus','fontsize',14,'fontname','Arial')
grid
legend('t','t+0.5','t-0.25')
```

Jika program tersebut dijalankan, akan diperoleh grafik seperti Gambar dibawah:



Gambar 7.3 Grafik fungsi sinus lengkap dengan grid dan legend

7.4 Membentuk Subplot

Perintah subplot digunakan untuk menggambar lebih dari 1 grafik dalam 1 plot. Dalam MATLAB, perintah ini ditulis subplot (mnp). Elemen m dan n menunjukkan banyaknya baris dan kolom dalam ruang *plot*, sedangkan p menunjukkan posisi dalam ruang *plot*. Cara termudah untuk memahami posisi dalam ruang *plot* adalah menganggapnya sebagai elemen matriks A_{ij} . Misal, diketahui 4 buah fungsi, yaitu:

- $y_1 = x + 10$
- $y_1 = x^2 + x + 10$
- $y_1 = x^3 + x^2 + x + 10$
- $y_1 = x^4 + x^3 + x^2 + x + 10$

Fungsi-fungsi ini terdiri atas 4 buah. Jadi, harus di-plot dalam ruang 2 x 2 dengan perintah dasar subplot (22p). Posisi masing-masing subplot adalah p, dengan p=1, 2, 3, 4. Setiap posisi p terletak pada elemen-elemen matriks secara berturut-turut dari baris 1 kolom 1 ke kolom berikutnya, dilanjutkan dengan baris berikutnya dengan urutan yang sama dimulai pada baris 1.

Fungsi-fungsi y_i ($i = 1, 2, 3, 4$) pada kasus ini, mempunyai posisi: p=1: A_{11} ; p=2: A_{12} ; p=3: A_{21} ; p=4: A_{22} .

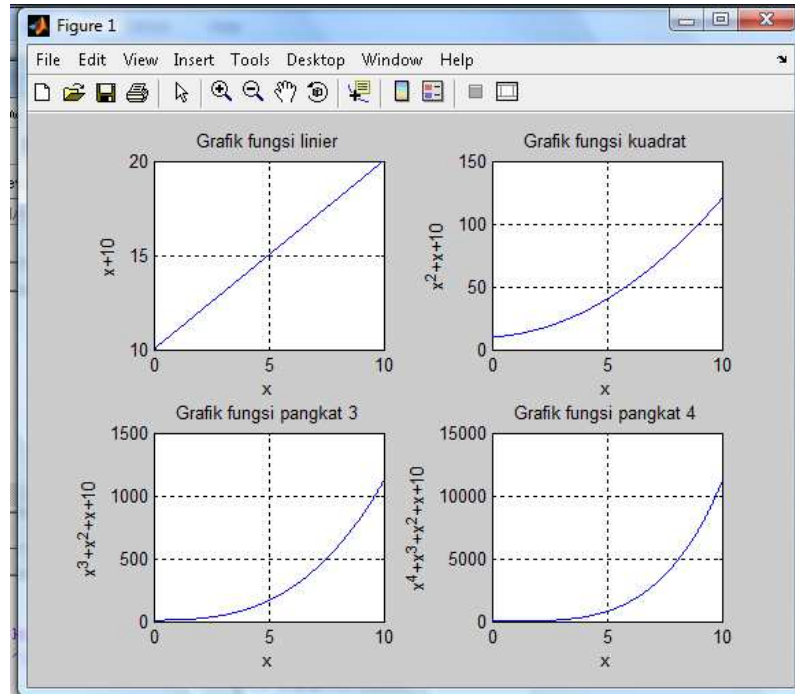
```
%Program dengan subplot
%By: NAMA ANDA
clear;
x = 0 : 0.01 : 10
y1=x+10*ones(l,length(x))
Y2=x.^2+x+10*ones(l,length(x))
Y3=x.^3+x.^2+x+10*ones(l,length(x))
y4=x.^4+x.^3+x.^2+x+10*ones(l,length(x))
%Grafik y1 subplot(221) plot(x,y1);grid;
title('Grafik fungsi linear') xlabel(lx);ylabel(lx+101)
%Grafik y2 subplot(222) plot(x,y2);grid;
title('Grafik fungsi kuadrat')
xlabel(lx);ylabel('x^2+x+101')
%Grafik y3 subplot(223) plot(x,y3);grid;
```

```

title('Grafik fungsi pangkat 3') xlabel('x');ylabel('x^3+x^2+x+101')
%Grafiky4
subplot(224) plot(x,y4);grid;
title('Grafik fungsi pangkat 4') xlabel('x');ylabel('x^4+x^3+x^2+x+101')

```

Dengan menjalankan program tersebut, akan diperoleh 1 grafik dalam 1 *Figure Window* seperti pada Gambar dibawah:



Gambar 7.4 Contoh Sub Grafik

7.5 Menggambar Grafik Tiga Dimensi

Bagian sebelumnya dalam bab ini, menguraikan cara menggambar grafik 2 dimensi. Selain graft 2-dimensi dikenal pula graft 3-dimensi. Syarat utama untuk menggambar grafik 3-dimensi adalah *array* yang terdiri atas 3 pasangan data dengan panjang yang sama. Berikut ini akan diuraikan beberapa *syntax* dasar untuk menggambar grafik 3--dimensi, yaitu:

- *plot (x, y, z)* : menggambar grafik yang merupakan pasangan *array* data pada vektor *x*, *y* dan *z*.
- *plot (x, y, z, S)* : menggambar grafik yang merupakan pasangan *array* data pada vektor *x*, *y* dan *z*, serta menggunakan karakter *s*.

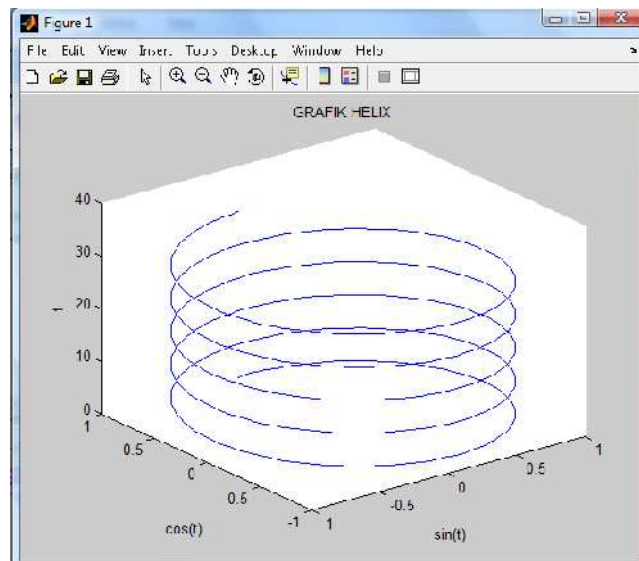
- *plot (X,Y,Z)* : menggambar grafik yang merupakan pasangan *array* data berupa kolom pada matriks X, Y dan Z.
- *plot (X,Y,Z,S)*: menggambar grafik yang merupakan pasangan *array* data berupa kolom pada matriks X, Y dan Z, serta menggunakan karakter S.

Karakter s adalah warna, tipe titik data dan tipe garis penghubung data. Uraian selengkapnya tentang karakter s diperlihatkan pada Tabel 7.1.

7.5.1 Grafik garis tiga dimensi

Grafik garis 3 dimensi merupakan pengembangan perintah dasar plot, dari 2-satuan data menjadi 3-satuan data. Program berikut akan menghasilkan grafik seperti gambar 5.5

```
clear
t=0:pi/50:10*pi;
x=sin(t);
y=cos(t);
plot3(x,y,t);
title('GRAFIK HELIX','fontsize',20,'fontname','Arial');
xlabel('sin(t)','fontsize',16,'fontname','Arial');
ylabel('cos(t)','fontsize',16,'fontname','Arial')
xlabel('t','fontsize',16,'fontname','Arial')
```

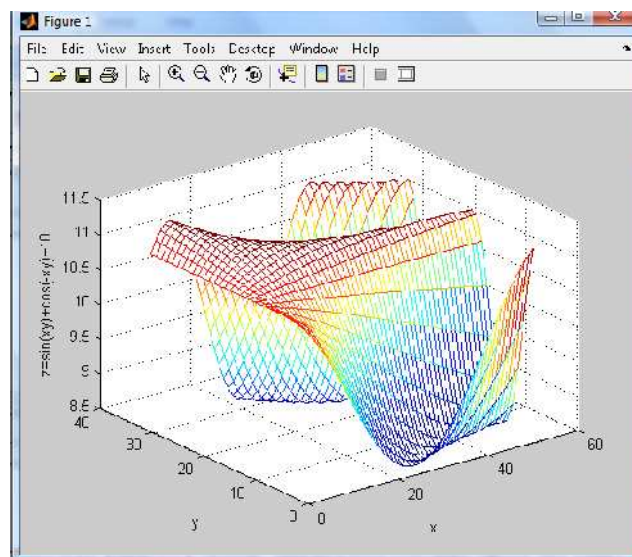


Gambar 7.5 Contoh Grafik Helix

7.5.2 Grafik jala dan permukaan

Grafik jala merupakan koordinat z dari sebuah titik diatas grid segiempat pada bidang x - y yang menyerupai jala. Grafik jala dihasilkan dengan syntax `mesh (z)`. Array data yang divisualisasikan adalah vector x berukuran n , vector y berukuran m dan matriks z berukuran $m \times n$. Titik data yang tergambar merupakan pasangan $\{x(i),y(j),z(i,j)\}$, dengan x berhubungan dengan kolom-kolom z dan y berhubungan dengan baris-baris z . Sebagai contoh, program berikut akan menghasilkan grafik seperti pada Gambar 5.6

```
x=-1:1:2;
y=0:1:5;
for i=1:length(x)
for j=1: length (y)
Z(i,j)=sin (x(i)* y(j))+cos(-x(i)* y(j))+10;
end
end
surf(x,y,Z)
mesh(Z)
xlabel (' x ', ' fontsize', 14, ' fontname', ' Arial' );
ylabel (' y ', ' fontsize', 14, ' fontname', ' Arial' )
zlabel (' z=sin(xy)+cos(-xy)+10' )
```



Gambar 7.6 Grafik permukaan hasil `surf(Z)`

SOAL:

1. Lakukan modifikasi terhadap program pada tutorial nomor 1, kemudian buat program untuk menggambarkan fungsi sinus, cosinus, tangen dan cotangen untuk kasus 2 periode dalam 1 *Figure Window*.
2. Spektroskopi nuklir merupakan instrumen yang berfungsi untuk mengetahui karakteristik spektrum energi radiasi. Pengaruh energi yang dipancarkan nuklida radioaktif terhadap tegangan keluaran sistem spektroskopi nuklir diketahui melalui tinggi pulsa yang tampak pada osiloskop. Misal, diketahui data pengamatan seperti pada Tabel 5.2, sedangkan data energi pada Tabel 5.2 merupakan standar kalibrasi dari *Laboratory Education and Training, National Atomic Energy Agency-From JRIA Book*. Perkiraan kesetaraan energi disesuaikan dengan kecenderungan bahwa pulsa tertinggi pada osiloskop sama dengan tingkat energi tertinggi. Pengamatan selanjutnya dilakukan pada unsur yang belum diketahui (Tabel 5.3). Besaran yang diamati adalah energi yang dipancarkan nuklida akibat perubahan tegangan keluaran sistem spektroskopi nuklir. Perkiraan kesetaraan ditentukan berdasarkan posisi puncak energi pada *display layer* terhadap tegangan yang terbaca pada osiloskop. Jika diketahui data pengamatan seperti Tabel 5.3, buatlah grafik hubungan antara besar energi nuklida radioaktif terhadap tegangan listrik yang ditimbulkan (Tabel 5.2) dan hubungan antara tegangan terhadap energi radiasi (Tabel 5.3). Gunakan perintah stem (**X, Y**) untuk menggambar kedua grafik tersebut. Gambarlah kedua grafik dalam 1 *figure window* dengan bantuan perintah hold on untuk menampilkan *array* data dalam 1 grafik. Lakukan analisis lebih lanjut untuk menentukan unsur yang belum diketahui (*unknown*). Tebaklah unsur apakah yang sedang diamati?. (Petunjuk: Gunakan X: Tegangan dan Y. Energi.!).

Tabel 5.2: Pengaruh energi terhadap tegangan

No	Nuklida	Energi (keV)	Tegangan (Volt)
1	Ba-133	302	1,8
2	Cs-137	662	3,6
3	Na-22	1275	5,8
4	Co-60	1332	6,0

Tabel 5.3: Pengaruh Tegangan terhadap energy radiasi

No	Tegangan (Volt)	Energi (keV)
1	1,0	25
2	1,5	49
3	2,0	214
4	2,5	276
5	3,0	340
6	3,5	390
7	4	457
8	5	571
9	6	679
10	7	819
11	8	911

Bab VIII

Akar-Akar Persamaan



8.1 Akar persamaan

Akar-akar persamaan adalah bilangan yang merepresentasikan titik potong kurva polinomial terhadap variabel bebasnya. Pada bagian ini akan diuraikan beberapa metode penentuan akar-akar persamaan. Polinomial berderajat dua atau biasa disebut persamaan kuadrat dinyatakan dalam bentuk

$$y = ax^2 + bx + c \dots\dots\dots (8.1)$$

Akar-akar persamaan (4.1) dapat ditentukan secara analitik dengan formula

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \dots\dots\dots (8.2)$$

sedangkan persamaan dengan derajat lebih tinggi, akar-akarnya tidak dapat ditentukan dengan persamaan (4.2), misalnya:

$$f(x) = x^3 + x^2 - 3x - 3$$

$$f(x) = x^5 + 2x^4 + 3x^3 + 4x^2 - 3x - 1$$

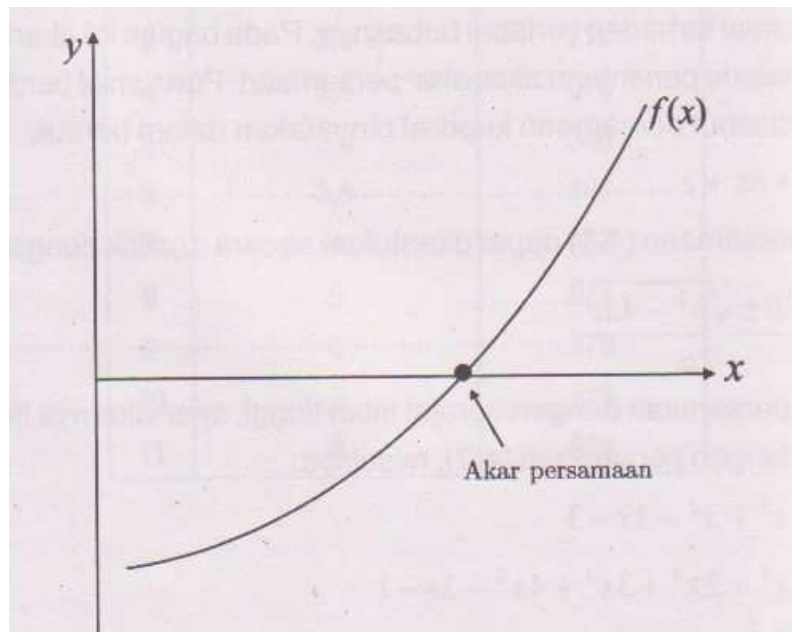
$$f(x) = e^x - 3x$$

$$f(x) = 3x + \sin x - e^x$$

Persamaan-persamaan tersebut hanya bisa diselesaikan dengan metode numerik. Metode ini merupakan cara penyelesaian dengan teknik perkiraan tertentu sampai diperoleh hasil yang mendekati penyelesaian eksak. Penyelesaian numerik diselesaikan dengan pendekatan berurutan (iterasi) sedemikian hingga hasil yang diperoleh pada setiap *step* lebih teliti dari sebelumnya. Dengan melakukan sejumlah prosedur iterasi yang dianggap cukup, akan didapat hasil perkiraan yang mendekati nilai eksak dengan toleransi kesalahan yang diperkenankan.

Metode paling sederhana untuk mendapatkan penyelesaian perkiraan adalah dengan menggambarkan fungsi $f(x)$ kemudian menentukan titik potongnya terhadap sumbu x . Titik

potong tersebut menunjukkan akar persamaan dari fungsi yang digambarkan (Gambar 4.1). Metode lain yang lebih sederhana untuk menyelesaikan persamaan kuadrat adalah metode coba banding. Prosedur metode ini adalah mencoba nilai x sebarang, kemudian dievaluasi apakah $f(x) = 0$. Jika nilai x tidak sama dengan nol, maka dicoba dengan nilai yang lain sedemikian hingga diperoleh nilai $f(x) = 0$. Nilai yang membuat $f(x) = 0$ merupakan akar persamaan. Kedua cara sederhana yang diuraikan sepintas tidak efisien dan sistematis. Ada beberapa metode yang juga merupakan perkiraan, tetapi lebih sistematis untuk menghitung akar-akar persamaan. Metode-metode tersebut akan diuraikan pada bagian-bagian berikutnya dalam bab ini.



Gambar 8.1: Akar persamaan dari fungsi $f(x)$

8.2 Metode Setengah Interval

Metode setengah interval biasa disebut metode bagi dua atau metode *bisection*. Prosedur yang dilakukan untuk menyelesaikan persamaan dengan metode setengah interval adalah:

1. Menghitung fungsi pada interval yang sama dari x sampai diperoleh perubahan tanda untuk fungsi $f(x)$ dan $f(x_{n+1})$, yaitu $f(x) \times f(x_{n+1}) < 0$.
2. Melakukan estimasi pertama terhadap akar x , yang dihitung dengan formula

$$x_1 = \frac{x_n + x_{n+1}}{2} \dots\dots\dots (8.3)$$

3. Membuat evaluasi untuk menentukan sub interval (Gambar 4.2) tempat akar persamaan berada dengan kriteria:

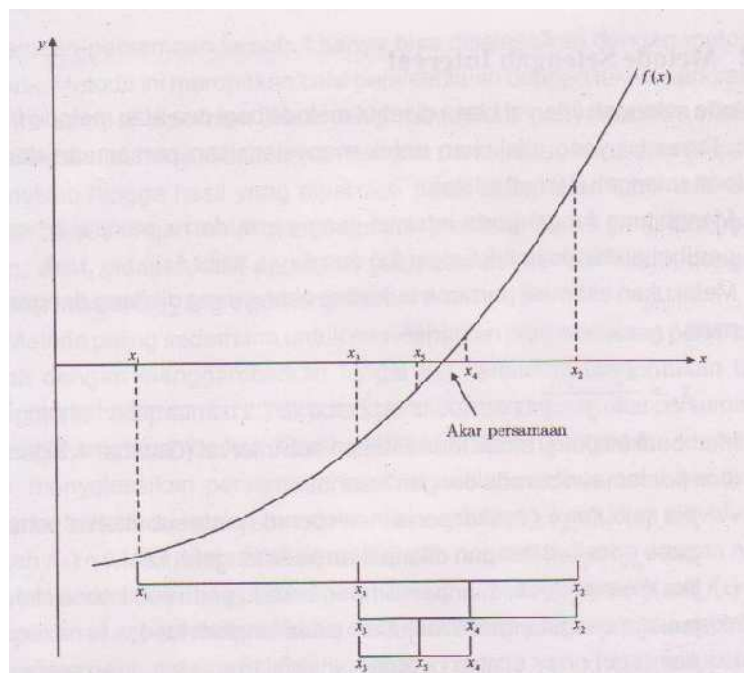
- Jika $f(x) \times f(x_{n+1}) < 0$, akar persamaan berada pada sub interval pertama, jadi $x_{n+1} = x_t$ hitungan dilanjutkan pada langkah ke-4.
- Jika $f(x) \times f(x_{n+1}) > 0$, akar persamaan berada pada sub interval kedua, jadi $x_n = x_t$ hitungan dilanjutkan pada langkah ke-4.
- Jika $f(x) \times f(x_{n+1}) = 0$, akar persamaan adalah x_t hitungan selesai.

4. Menghitung perkiraan akar baru dengan formula¹

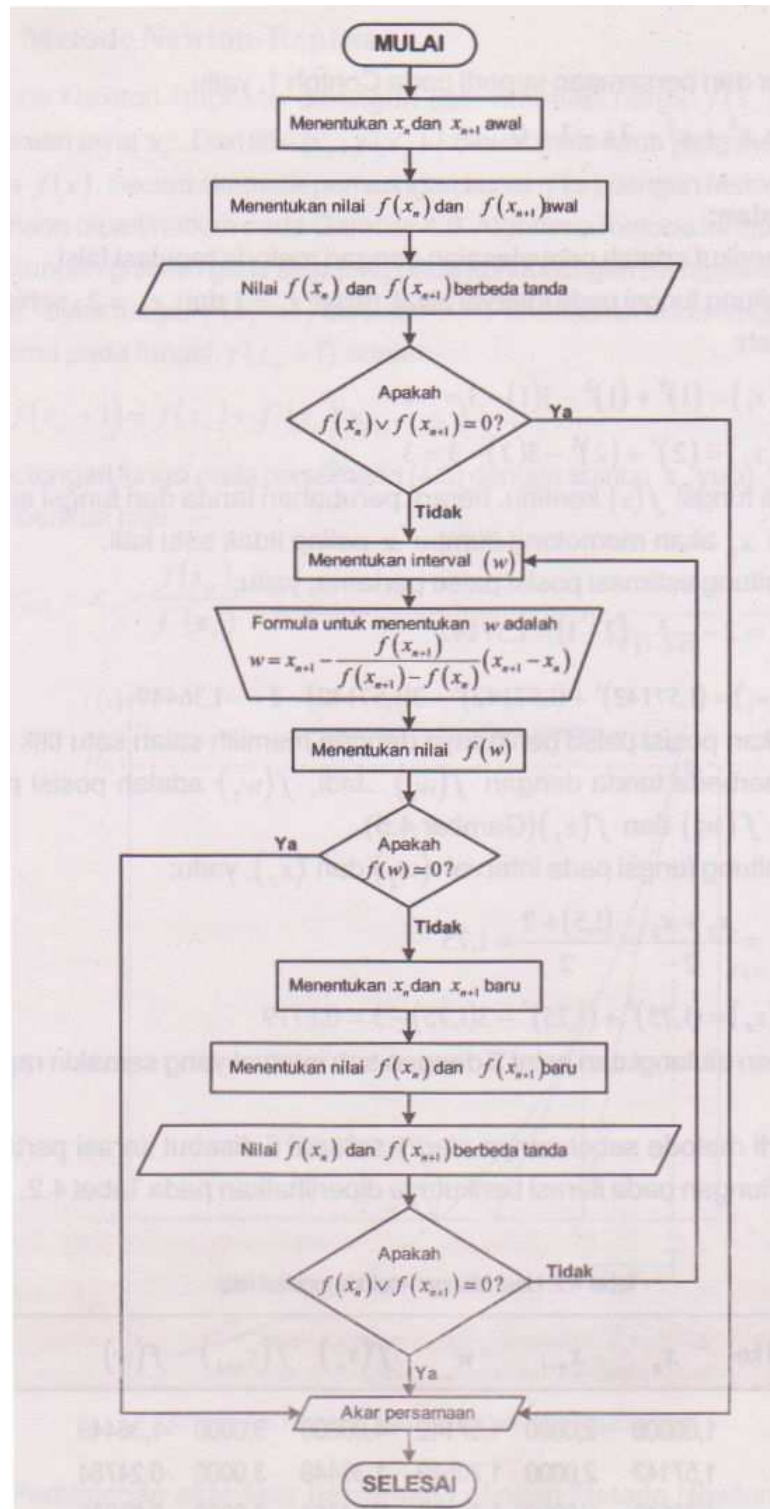
$$x_t^* = \frac{x_n^* + x_{n+1}^*}{2} \dots\dots\dots(8.4)$$

5. Jika perkiraan akar baru cukup kecil atau sesuai dengan target awal dalam batasan yang dapat diterima, hitungan dianggap selesai dengan x , adalah akar persamaan. Jika perkiraan belum kecil, hitungan diulang dari langkah ke-3 sampai diperoleh hasil yang sesuai dengan target awal.

Prosedur hitungan dalam bentuk graft dan *flow chart* diperlihatkan pada Gambar 6.2 dan Gambar 6.3.



Gambar 8.2: Prosedur hitungan metode setengah interval



Gambar 8.3: Bagan alir metode setengah interval

Contoh 1:

Hitung salah satu akar persamaan polynomial orde 3 berikut.

$$f(x) = x^3 + x^2 - 3x - 3 = 0$$

Penyelesaian:

Akar-akar Persamaan ini dihitung dengan prosedur berikut:

1. Menghitung fungsi pada interval awal, misal $x_1 = 1$ dan $x_2 = 2$, sehingga diperoleh:

- $f(x) = (1)^3 + (1)^2 - 3(1) - 3 = -4$
- $f(x_2) = (2)^3 + (2)^2 - 3(2) - 3 = 3$

Karena fungsi $f(x)$ kontinu, berarti perubahan tanda antara x_1 dan x_2 pada fungsi tersebut akan memotong sumbu x paling tidak 1 kali.

2. Menghitung estimasi sub interval pertama, yaitu:

- $x_3 = \frac{x_1 + x_2}{2} = \frac{-4 + 3}{2} = 1,5000$

- $f(x_3) = (1,5)^3 + (1,5)^2 - 3(1,5) - 3 = 0,18750$

3. Menentukan sub interval berikutnya dengan memilih salah satu titik awal yang berbeda tanda dengan $f(x_3)$. Jadi $f(x_4)$ adalah sub interval antara $f(x_3)$ dan $f(x_4)$ (Gambar 4.2).

4. Menghitung fungsi pada interval (x_3) dan (x_2) , yaitu:

- $x_2 = \frac{x_1 + x_2}{2} = \frac{-4 + 3}{2} = 1,75$

- $f(x_4) = (1,75)^3 + (1,75)^2 - 3(1,75) - 3 = 0,1719$

5. Hitungan diulangi dari point 3 dengan sub interval yang semakin rapat.

Step 1 sampai 5 disebut 1 iterasi. Prosedur perhitungan yang telah dilakukan dengan hasil $f(x_4) = 0,1719$ disebut iterasi pertama. Dari prosedur ini terlihat bahwa nilai $f(x_4)$ belum kecil atau belum mendekati nol. Nilai seperti ini dalam perhitungan dengan metode setengah interval dianggap belum merepresentasikan akar persamaan, sehingga perlu dilakukan perhitungan lebih lanjut berdasarkan bagan alir pada Gambar 8.3. Hasil perhitungan yang diperoleh pada prosedur tersebut diperlihatkan pada Tabel 8.1. Prosedur perhitungan ini sangat mudah dilakukan tetapi tidak efisien, karena membutuhkan iterasi yang cukup panjang.

Tabel 8.1 Prosedur Perhitungan

Iterasi ke-	x_n	x_{n+1}	x_r	$f(x_n)$	$f(x_{n+1})$	$f(x_r)$
1	1,0000	2,0000	1,5000	-1,0000	3,0000	-1,8750
2	1,5000	2,0000	1,7500	-1,8750	3,0000	0,1719
3	1,5000	1,7500	1,6250	-1,8750	0,1719	-0,9434
4	1,6250	1,7500	1,6875	-0,9434	0,1719	0,4094
5	1,6875	1,7500	1,7188	-0,4094	0,1719	-0,1248
6	1,7188	1,7500	1,7344	-0,1248	0,1719	-0,0220
7	1,7188	1,7344	1,7266	-0,1248	0,1719	-0,0220
8	1,7266	1,7344	1,7305	-0,0518	0,0220	-0,0150
9	1,7305	1,7344	1,7324	-0,0150	0,0220	0,0035
10	1,7305	1,7324	1,7344	-0,0150	0,0035	-0,0057
11	1,7314	1,7324	1,7319	-0,0057	0,0035	-0,0011
12	1,7319	1,7324	1,7322	-0,0011	0,0035	0,0012
13	1,7319	1,7321	1,7321	-0,0011	0,0012	0,0000

8.3 Metode Regulasi Falsi

Metode regulasi falsi boleh diartikan **metode posisi palsu** karena metode ini memberikan posisi palsu akar w berdasarkan titik perpotongan garis lurus yang melalui $(x_1, f(x_1))$ dan $(x_2, f(x_2))$ dengan tanda yang berbeda dan kontinu. Dari titik tersebut dapat dilakukan interpolasi linear, sehingga metode ini juga sering disebut metode **interpolasi linear**. Dengan metode ini akar-akar persamaan fungsi yang ditinjau lebih cepat diperoleh tanpa iterasi yang cukup panjang. Metode ini lebih efektif jika dibandingkan dengan metode setengah interval.

Perhitungan akar-akar persamaan dengan metode regulasi falsi dilakukan dengan prosedur berikut:

1. Menentukan fungsi pada interval x yang sama sampai diperoleh perubahan tanda dari fungsi $f(x)$ dan $f(x_{n+1})$ atau $f(x)x(x_{n+1}) < 0$ (Gambar 8.4).
2. Dari kedua nilai fungsi $f(x)$ dan $f(x_{n+1})$ ditarik garis lurus sehingga terbentuk segitiga sebangun yang memenuhi Persamaan (8.5) dan (8.6).

$$\frac{x_{n+1} - w}{x_{n+1} - x_n} = \frac{f(x_{n+1})}{f(x_{n+1}) - f(x_n)} \quad \dots\dots\dots(8.5)$$

$$w = x_{n+1} - \frac{f(x_{n+1})}{f(x_{n+1}) - f(x_n)}(x_{n+1} - x_n) \quad \dots\dots\dots(8.6)$$

Jadi langkah ke-2 adalah menentukan w menggunakan persamaan (8.6) dan nilai fungsi $f(w)$.

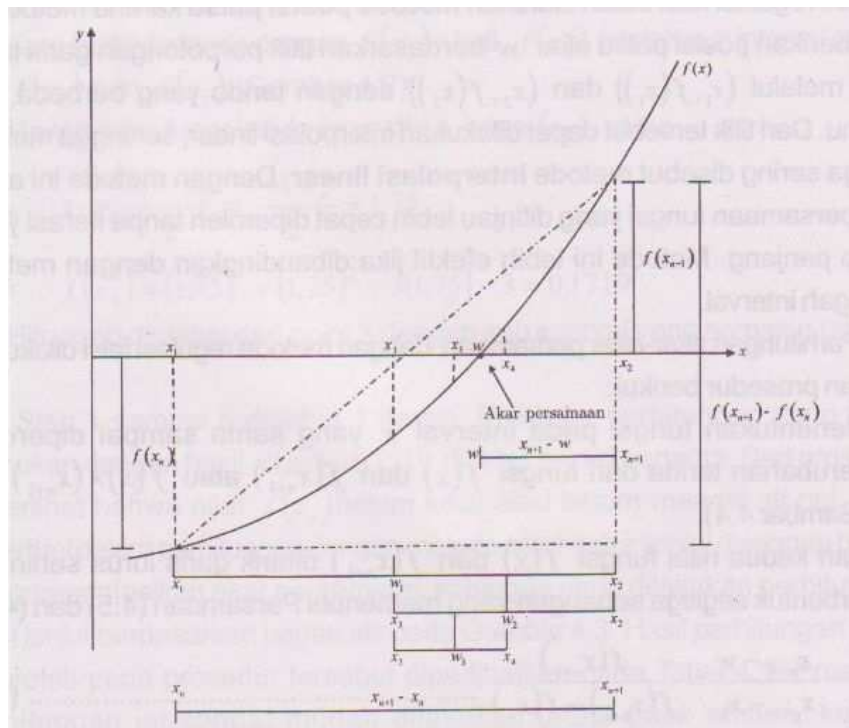
3. Melakukan evaluasi terhadap fungsi $f(w)$ dan $f(x_{n+1})$ untuk menentukan sub interval tempat w berada (Gambar 6.4) dengan kriteria:

- jika $f(w) \times f(x_{n+1}) < 0$, akar persamaan berada pada sub interval pertama, jadi $x_{n+1} = w$ hitungan dilanjutkan pada langkah ke-4.
- jika $f(w) \times f(x_{n+1}) > 0$, akar persamaan berada pada sub interval kedua, jadi $x_n = w$ hitungan dilanjutkan pada langkah ke-4.
- jika $f(x) \times f(x_{n+1}) = 0$, akar persamaan adalah w hitungan selesai.

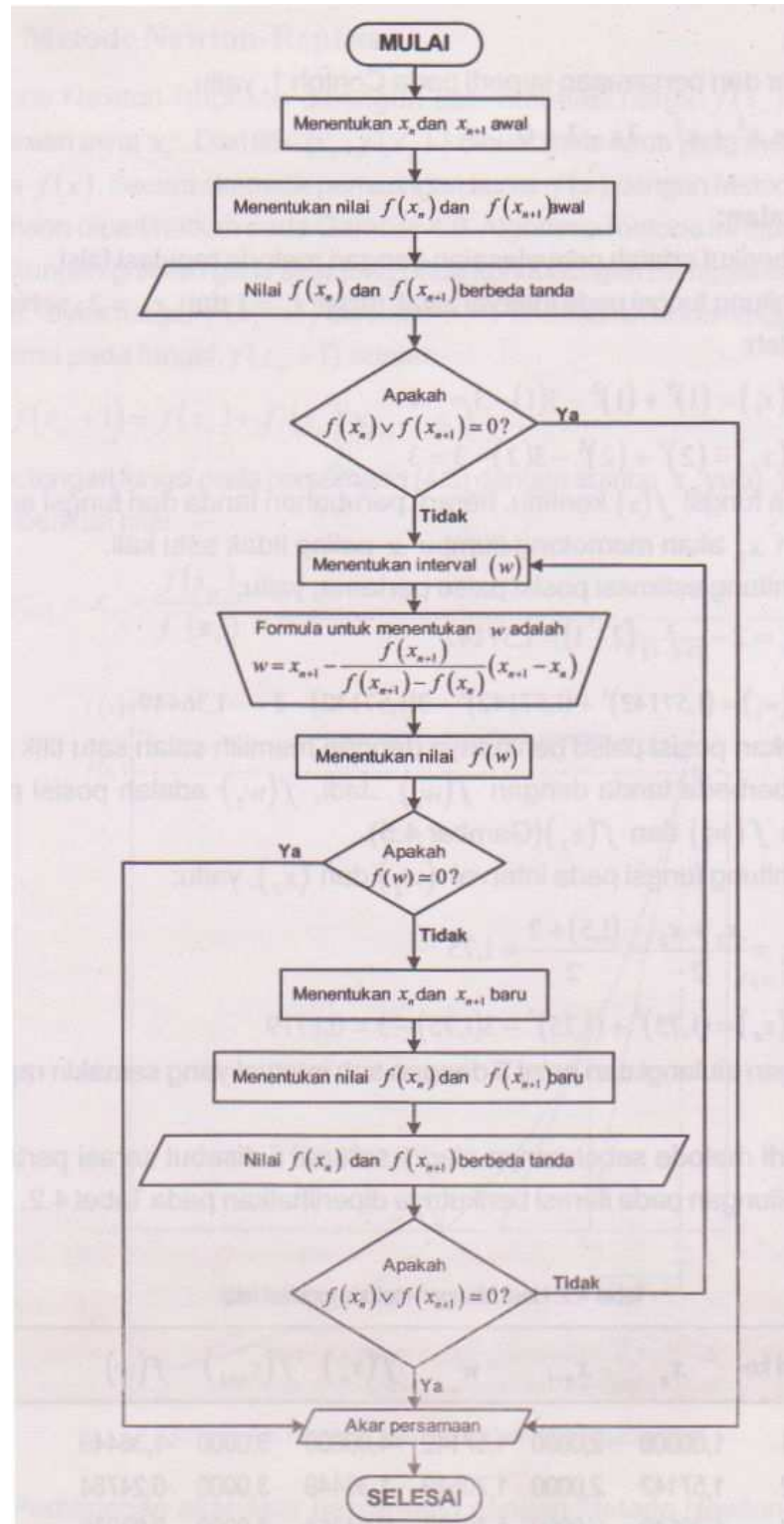
4. Melakukan interpolasi baru menggunakan persamaan:

$$w^* = x_{n+1} - \frac{f(x_{n+1})}{f(x_{n+1}) - f(x_n)} (x_{n+1} - x_n) \quad \dots\dots\dots(8.7)$$

5. Jika $f(w^*)$ cukup kecil atau $f(w^*) = 0$ perhitungan dinyatakan selesai dengan w^* adalah akar persamaan. Jika $f(w^*)$ belum kecil, perhitungan diulangi dari langkah ke -3.



Gambar 8.4. Prosedur hitungan metode regulasi falsi



Gambar 8.5. Bagan alir metode regulasi falsi

Contoh 2:

Hitung akar dari persamaan seperti pada Contoh 1, yaitu

$$f(x) = x^3 + x^2 - 3x - 3 = 0$$

Penyelesaian:

Prosedur berikut adalah penyelesaian dengan metode regulasi falsi.

1. Menghitung fungsi pada interval awal, misal $x_1 = 1$ dan $x_2 = 2$, sehingga diperoleh:

- $f(x_1) = (1)^3 + (1)^2 - 3(1) - 3 = -4$
- $f(x_2) = (2)^3 + (2)^2 - 3(2) - 3 = 3$

Karena fungsi $f(x)$ kontinu, berarti perubahan tanda dari fungsi antara x_1 dan x_2 akan memotong sumbu x paling tidak satu kali.

2. Menghitung estimasi posisi palsu pertama, yaitu:

- $w_1 = 2 - \frac{3}{|3 - (-4)|} (2 - 1) = 1,57142$
- $f(w_1) = (1,57142)^3 + (1,57142)^2 - 3(1,57142) - 3 = -1,36449$

3. Menentukan posisi palsu berikutnya dengan memilih salah satu titik awal yang berbeda tanda dengan $f(w_1)$. Jadi, $f(w_2)$ adalah posisi palsu antara $f(w_1)$ dan $f(x_2)$ (Gambar 6.5).

4. Menghitung fungsi pada interval (w_1) dan (x_2) , yaitu:

- $w_2 = \frac{x_3 + x_2}{2} = \frac{(1,5) + 2}{2} = 1,75$

4. $f(x_4) = (1,75)^3 + (1,75)^2 - 3(1,75) - 3 = 0,1719$

5. Hitungan diulangi dari point 3 dengan sub interval yang semakin rapat.

Seperti metode sebelumnya *step* 1 sampai 5 disebut iterasi pertama. Hasil perhitungan pada iterasi berikutnya diperlihatkan pada Tabel 8.2.

Tabel 8.2: Hasil hitungan metode regulasi falsi

Iterasi ke-	x_n	x_{n+1}	w	$f(x_n)$	$f(x_{n+1})$	$f(w)$
1	1,00000	2,0000	1,57142	-4,00000	3,0000	-1,36449
2	1,57142	2,0000	1,70540	-1,36449	3,0000	-0,24784
3	1,70540	2,0000	1,72788	-0,24754	3,0000	-0,03936
4	1,73140	2,0000	1,73140	-0,03936	3,0000	-0,00615

8.4 Metode Iterasi

Metode iterasi adalah metode perkiraan akar persamaan dengan menggunakan hasil transformasi dari persamaan itu sendiri. Misal sebuah fungsi $f(x) = 0$ ditransformasikan dengan menambahkan variabel bebas pada fungsi itu sendiri. Misal, variabel x ditambahkan pada fungsi $g(x)$ sehingga diperoleh persamaan baru dengan bentuk umum

$$x = g(x) \dots\dots\dots (8.12)$$

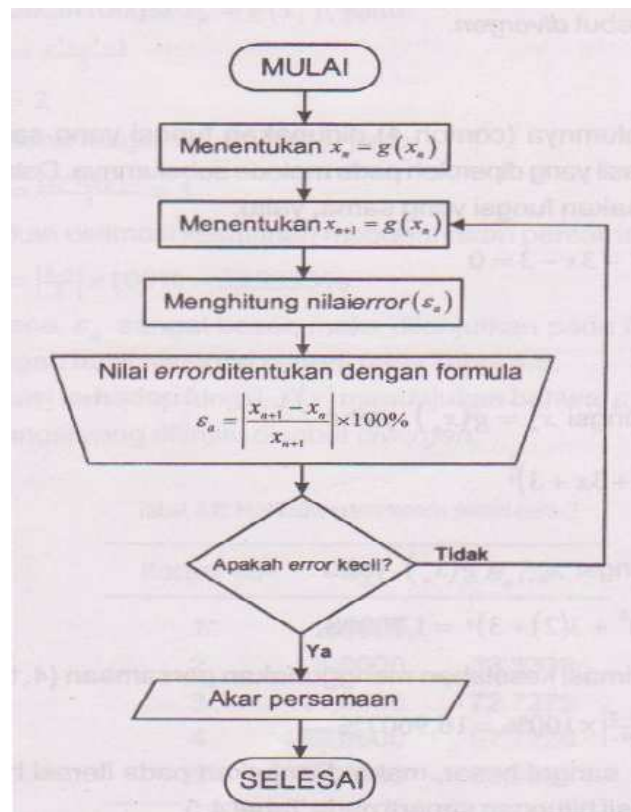
Persamaan (8.12) menunjukkan bahwa nilai x merupakan fungsi dari x . Dengan memberikan nilai perkiraan awal yaitu x_n , perkiraan akar selanjutnya yaitu x_{n+1} , dapat dihitung dengan rumus iterasi

$$x_{n+1} = g(x_n) \dots\dots\dots (8.13)$$

Besarnya kesalahan (*error*) dihitung dengan formula

$$e_a = \left| \frac{x_{n+1} - x_n}{x_{n+1}} \right| \times 100\% \dots\dots\dots (8.14)$$

Bagan alir metode iterasi diperlihatkan pada Gambar dibawah:



Gambar 8.6 Diagram Alir Metode Iterasi

Penentuan akar-akar persamaan menggunakan metode iterasi dilakukan dengan prosedur berikut:

1. Menentukan fungsi $x = g(x)$,
2. Menghitung fungsi $x_{n+1} = g(x_n)$,
3. Melakukan estimasi kesalahan menggunakan persamaan (4.14) dengan kriteria:
 - jika c_n sangat kecil ($\epsilon \cdot 0$) dengan $x_n \rightarrow 0$, perhitungan selesai dan x_n adalah akar persamaan.
 - jika tidak, maka perhitungan diulangi dari langkah ke-2 dengan perkiraan baru, yaitu $x_{n+1}^* = g(x_n)$ -
4. Melakukan identifikasi terhadap fungsi dengan kriteria:
 - jika c_n semakin kecil pada iterasi berikutnya, maka fungsi yang ditinjau disebut *konvergen*.
 - jika c_n semakin besar pada iterasi berikutnya, maka fungsi yang ditinjau disebut *diverges*.

Contoh :

Pada contoh sebelumnya digunakan fungsi yang sama untuk membandingkan hasil yang diperoleh pada metode sebelumnya. Dalam contoh ini juga akan digunakan fungsi yang sama, yaitu:

$$A^x = x^3 + x' - 3x - 3 = 0$$

Penyelesaian:

Cara-1

1. Menentukan fungsi $x = g(x)$, yaitu:
 - $x_{n+1} = x_n^2 + 3x_n + 3$
 - $x_1 = 2$
2. Menghitung fungsi $x_{n+1} = g(x_n)$, yaitu :
 - $x_2 = (2)^2 + 3(2) + 3 = 1,70998$
3. Melakukan estimasi kesalahan menggunakan persamaan (6.14):

$$\frac{11,70998 - 21}{21} \times 100\% = 16,9607\% \quad 7 \quad 0 \quad \% \quad S$$
 - Karena c_0 sangat besar, maka dilanjutkan pada iterasi berikutnya dengan hasil hitungan seperti pada Tabel 6.5.

Identifikasi terhadap fungsi $f(x)$ menunjukkan bahwa --, semakin kecil, maka fungsi yang ditinjau disebut *konvergen*.

Cara-2

1. Menentukan fungsi $x_{n+1} = g(x_n)$, yaitu:

$$x_{n+1} = \frac{x_n + x_n^2 - 3}{3}$$

$$x_1 = 3$$

$$x_2 = 2$$

2. Menghitung fungsi $x_{n+1} = g(x_n)$, yaitu:

$$x_2 = (2)' + (2)^2 - 3 = 3$$

3. Melakukan estimasi kesalahan menggunakan persamaan (6.14): $\frac{1}{3} \times 100\% = 33,3333\%$

- karena c_n sangat besar, maka dilanjutkan pada iterasi berikutnya dengan hasil hitungan seperti pada Tabel 6.6.

4. Identifikasi terhadap fungsi $f(x)$ menunjukkan bahwa c_n semakin besar, maka fungsi yang ditinjau disebut *divergen*.

Tabel 8.3: Hasil hitungan metode iterasi cara-2

Iterasi ke-	x_n	$\varepsilon_a \%$
1	2,0000	—
2	3,0000	33,3333
3	11,0000	72,7273
4	483,0000	97,7226
5	37637290	99,9987

Hasil perhitungan yang diperoleh melalui cara-1 dan cara-2 menunjukkan bahwa cara ini kurang efektif karena dibutuhkan fungsi, nilai awal dan persentase kesalahan yang belum tentu valid. Tetapi metode ini mampu mengidentifikasi suatu fungsi konvergen atau divergen.

SOAL:

1. Buat program untuk menghitung akar-akar persamaan berikut, dengan metode setengah interval.

$$f(x) = x^2 + 2x - 5 = 0$$

2. Buat program untuk menghitung akar-akar persamaan berikut, dengan metode Regulasi Falsi

$$f(x) = x^2 + 2x - 5 = 0$$

3. Buat program untuk menghitung akar-akar persamaan berikut, dengan metode Iterasi.

$$f(x) = x^2 + 2x - 5 = 0$$

Bab IX

Membuat Fungsi Dengan Matlab



Fungsi Dalam M-File

Pada dasarnya semua tools yang disediakan oleh MATLAB dibuat dalam format fungsi dan dikelompokkan dalam folder-folder toolbox. Selain menggunakan fungsi-fungsi yang telah disediakan oleh MATLAB, kita juga dapat membuat fungsi-fungsi sendiri sesuai kebutuhan.

Keuntungan membuat program dalam format fungsi adalah kemudahannya untuk digunakan kembali pada program yang lain. Untuk membangun sebuah fungsi, MATLAB memberikan kita satu pola penulisan untuk diikuti. Yaitu sebagai berikut:

Bagian 1	Function [out1, out2,...] = Nama (un1,in2,...)
Bagian 2	% Penjelasan fungsi
Bagian 3	---Statement fungsi---
	---Statement fungsi---

Ada 3 bagian pokok dalam penulisan fungsi, yaitu:

1. Bagian deklarasi fungsi
2. Bagian penjelasan fungsi
3. Bagian program utama

Dari ketiga bagian tersebut hanya bagian 2 yang bersifat optional, artinya boleh ada boleh juga tidak.

Variabel out1, out2, in1 dan in2 adalah arguman output dan input fungsi. Dan jumlah input dan output yang dapat digunakan untuk fungsi tidak terbatas.

Nah, untuk memperjelas cara pembuatannya, mari kita kerjakan latihan berikut:

1. Pada command Window, ketikkan:

```
>>edit
```

2. Tekan enter, selanjutnya muncul MATLAB Editor dan anda ketiklah program dibawah berikut:

```
function y=fungsiku(x)
% -----
% Program latihn 16
```

```
% MATLAB Programming
% Oleh : gunay
%
% Fungsi y=x^3+12x^2-15x+34
%
% Cara menggunakan:
% y=fungsiku (12.7)
% -----
Y=x.^3+12*x.^2-15*x+34;
```

3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **fungsiku.m**. Perhatikan secara default MATLAB akan menyimpan nama file anda sama dengan nama fungsinya.
4. Pastikan direktori penyimpanan file anda sudah terdapat didalam daftar pencarian direktori MATLAB (lihat bab 2 Penting Untuk Pemula). Untuk menggunakan fungsi yang telah dibuat ketiklah sebagai berikut:

```
>> y=fungsiku(12)
```

5. Tekan enter, selanjutnya program akan dijalankan dan menghasilkan sebagai berikut:

```
>>y=fungsiku(12)
y =
    3310
```

6. Fungsi anda telah berhasil dijalankan, untuk melihat penjelasan fungsi, ketiklah sebagai berikut:

```
>> help fungsiku
-----
Program latihan 16
MATLAB Programming
Oleh: Gunay

Fungsi y=x^3+12x^2-15x+34

Cara menggunakan:
Y=fungsiku (12.7)
```

7. Selesai

Pada program sebelumnya kita telah membuat sebuah fungsi sederhana dengan jumlah argument input dan output tunggal. Nah, untuk lebih memahami program fungsi dengan jumlah argument yang lebih dari satu, mari kita kerjakan latihan berikut:

1. Pada Command Window, ketikkan:

```
>>edit
```

2. Tekan enter, selanjutnya muncul MATLAB Editor dan anda ketiklah program dibawah ini:

```
function [luas,isi]=balok(p,1,t)
% -----
% Program latihan 17
% MATLAB Programming
% Oleh: Gunay
%
% Menghitung luas dan Isi balok
% Luas= 2p1+2pt+ 2lt
% Isi = plt
% Cara menggunakan :
% [luas,isi] = balok (p,j,lbr,tg)
% -----
Luas= 2* p. * 1+2* p. * t+2* 1. * t;
Isi= p. * 1. * t;
```

3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **balok.m**. Perhatikan secara detail MATLAB akan menyimpan nama file anda sama dengan nama fungsinya.
4. Buatlah sebuah fungsi lagi dan ketiklah program berikut:

```
Function [luas,isi]=silinder (jari,tinggi)
% -----
% Program latihan 17
% MATLAB Programming
% Oleh Gunay
%
% menghitung luas dan Isi silinder
% luas = (2(pi) x r x t) +2(pi x r^2)
% Isi = pi x r^ 2 x t;
%
% Cara menggunakan:
% [luas,isi]=silinder (jari,tinggi)
% -----
Luas= (2* pi* jari* tinggi) + 2* (pi* jari^2)
Isi= pi* jari^2* tinggi;
```

5. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **silinder.m**. Perhatikan secara default MATLAB akan menyimpan nama file anda sama dengan nama fungsinya.
6. Selesai

Kedua fungsi (balok dan silinder) sekarang sudah menjadi fungsi yang kita definisikan sendiri dan telah siap untuk digunakan sebagaimana fungsi-fungsi MATLAB lainnya.

Untuk mengilustrasikan cara menggunakan fungsi yang telah kita buat pada sebuah program yang utuh, kerjakanlah latihan berikut ini.

1. Pada command Window, ketikkan:

```
>>edit
```

2. Tekan enter, selanjutnya muncul MATLAB Editor dan anda ketiklah program dibawah berikut:

```
% -----
% Program latihan 18
% MATLAB Programming
% Oleh: Gunay
% -----

clear all;
clc;

disp(' ----- ');
disp(' Program latihan 18 ');
disp(' ----- ');

disp(' Pilihan Rumus Perhitungan ');
disp(' 1.Kotak ');
disp(' 2.Silinder ');
disp(' ');

Pilih=input(' pilihan anda (1-2) -> ');
Switch pilih
Case 1
    disp(' Hitung Luas dan Isi Kotak ');
    disp(' ----- ');
    pjg=input(' panjang kotak= ');
    lbr=input(' lebar kotak= ');
    tg=input(' tinggi kotak= ');
    [luas,isi]=balok(pjg,lbr,tg);
    disp([' luas kotak= ' num2str(luas)]);
    disp([' volume kotak= ' num2str(isi)]);
Case2
    disp(' Hitung Luas dan Isi Silinder ');
    disp(' ----- ');
    r=input(' jari-jari silinder= ');
    tg=input(' tinggi silinder= ');
    [luas,isi]=silinder(r,tg);
    disp([' luas silinder= ' num2str(luas)]);
    disp([' volume silinder= ' num2str(isi)]);
otherwise
    disp(' pilihan anda ngawuur !!!! ')
end;
```


3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **latihan18.m**.
4. Pastikan direktori penyimpanan file anda sudah terdapat didalam daftar pencarian direktori MATLAB (lihat bab 2 Penting Untuk Pemula). Lalu ketiklah nama file latihan18 tanpa ekstensi:

```
>> latihan18
```

5. Tekan enter, selanjutnya program akan dijalankan, masukkan parameter input yang sesuai sehingga akan menghasilkan sebagai berikut:

```
-----  
Program latihan18  
-----  
Pilihan Rumus Perhitungan  
1. Kotak  
2. Silinder  
  
pilihan anda(1-2) - > 2  
hitung luas dan Isi silinder  
-----  
jari-jari silinder =12  
tinggi silinder =12  
luas silinder = 1809.5574  
volume silinder =5428.6721  
>>
```

6. Selesai

Bab X

Pemrograman Window



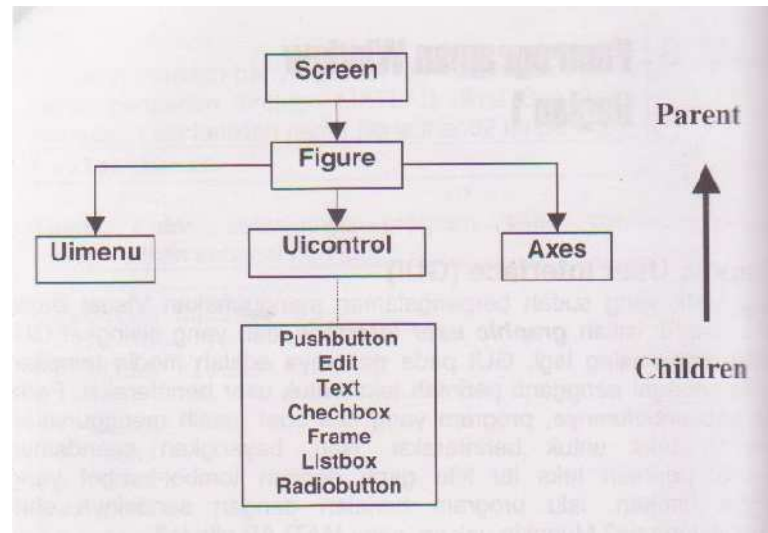
10.1 Graphic User Interface (GUI)

Bagi anda yang sudah berpengalaman menggunakan Visual Basic atau Delphi, istilah *graphic user interface* atau yang disingkat GUI sudah tidak asing lagi. GUI pada dasarnya adalah media tampilan grafis sebagai pengganti perintah teks untuk user berinteraksi. Pada bab-bab sebelumnya, program yang kita buat masih menggunakan perintah teks untuk berinteraksi.

10.1.1 Komponen Window Standar

Untuk keperluan pemrograman *Window*, MATLAB telah menyediakan komponen-komponen standar, seperti *pushbutton*, *edit*, *text*, *combo*, *checkbox* dan lain-lain untuk kita gunakan. Tetapi sebelum dapat menggunakan komponen-komponen tersebut dengan benar, kita harus memahami konsep **Pemrograman Berbasis Objek (PBO)** di MATLAB dengan benar.

Pada PBO, setiap komponen di artikan sebagai objek yang dapat diberikan pekerjaan maupun melakukan pekerjaan tertentu. Selain itu setiap objek dalam PBO pasti memiliki *property* untuk berinteraksi dengan objek lainnya. Dalam konteks pemrograman MATLAB sendiri, setiap objek tersebut memiliki hirarki objek yang dijabarkan dalam konsep *parent-children*. Berikut adalah diagramnya :



Gambar 10.1 Konsep Parent Children

Maksud dari diagram tersebut adalah setiap objek yang digunakan harus diposisikan pada objek *parent*-nya. Misalnya *pushbutton* harus diletakkan pada objek *figure* sebagai *parent*-nya. Seandainya anda masih bingung, tidak masalah kok. Nanti saat kita mulai menuliskannya dalam skrip pasti anda akan paham.

Objek paling tinggi dalam hirarki MATLAB adalah **Screen**, tetapi objek ini bertipe abstrak, dan pemrograman MATLAB tidak dapat langsung menyentuhnya. Maka objek tertinggi dalam pemrograman MATLAB kita fokuskan pada objek *figure*. Sintak umum menggunakan **objek** dalam pemrograman MATLAB adalah sebagai berikut :

```
ObjHandle = objek ([Property objek],[Property Value]);
```

Tentang objek-objek apa saja yang dapat kita gunakan akan dibahas setelah ini. Tetapi khusus untuk properti objek, sengaja tidak panjang lebar kita perbincangkan, karena jumlahnya yang banyak **dan** kesulitan penulis untuk mendeskripsikan satu persatu. **Menurut** penulis, lebih baik kita lihat saja hasil dari penggunaan setiap properti objek tersebut. Sehingga anda dapat mendefinisikannya dalam bahasa yang lebih mudah anda mengerti.

Pada umumnya objek yang paling sering digunakan dalam pemrograman MATLAB adalah sebagai berikut :

Objek Figure

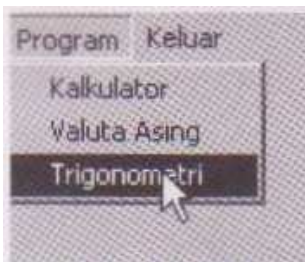
Figure adalah objek tertinggi yang dapat kita gunakan dalam pemrograman *Window*. Objek ini dapat diakses dengan beberapa properti penting menggunakan sintak sebagai berikut :

```
Nama = figure(...
'Color',[RGB],
'MenuBar',<'figure' 'None'>,
'Units',<'points' 'pixels'>,
'Position',[Left Top Width Height],
'Resize',<'on' 'off'>, ...
'NumberTitle',<'on' 'off'>,
'Name',[Teks Window], ...
'WindowStyle',<'normal' 'modal'>);
```

Contoh potongan skrip pemakaiannya adalah sebagai berikut :

```
win1=figure( ...
'units','points',...
'position',[100 150 500 300],...
'color',[.8 .8 .9],...
'menubar','none',...
'resize','off' ....
'numbertitle','off' ....
'name','Latihan WindowProgramming');
```

Hasil eksekusi program akan seperti berikut



Objek Uicontrol

Objek **Uicontrol** adalah objek yang paling kita butuhkan untuk berinteraksi dengan program. *Uicontrol* berisi komponen-komponen yang kita butuhkan untuk mendesain *form* untuk media interaksi. Objek ini dapat

diakses dengan beberapa properti penting menggunakan sintak sebagai berikut :

```
Nama = uicontrol( ...
'Parent',[NamaFigure],
'Style',[Komponen], ...
'Units','points' 'pixel'>,
'ListboxTop',0, ...
'Position',[Left Top Width Height],
'String',[Text pada Object],...
'Callback',[MATLAB Skrip]);
```

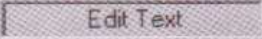
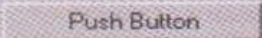


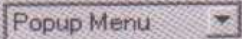
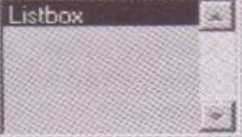
Dimulai dari sini, untuk seterusnya pada setiap penulisan objek yang baru kita harus mendefinisikan objek *parent* dari objek tersebut. Berikut adalah cuplikan skrip yang menggunakan *uicontrol*.

```
edit1=uiicontrol('parent',win1 ....
'units','points' ....
'position',[110 140 100 15],...
'backgroundcolor',[1 1 1],...
'style','Pushbutton' ....
'string','Exit',...
'fontname','arial',...
'fontsize',10,...
'Callback','Close');
```

perhatikan properti '*Parent*' yang diarahkan pada objek bernama 'win1' (penulisan nama objek tanpa tanda petik). Artinya objek anda yang bernama 'edit1' diletakkan pada objek 'win1'.

Lalu perhatikan pula properti '*Style*', nilai yang dituliskan pada properti ini adalah nama-nama komponen yang disediakan didalam objek *uicontrol*. Berikut ini adalah tabel nama-nama komponen yang dapat kita gunakan, berikut skripnya:

Tabel 10.1 Tabel Unicontrol

Skrip Style	Komponen Grafis
'style','Text',...	Static Text
'style','Edit',...	
'style','Pushbutton',...	
'style','Checkbox',...	<input type="checkbox"/> Checkbox
'style','Radiobutton',...	<input type="radio"/> Radio Button
'style','Slider',...	
'style','Frame',...	
'style','Popupmenu',...	
'style','Listbox',...	

Obiek Uimenu

Objek *uimenu* pada dasarnya mirip dengan *uicontrol* khususnya pada komponen *pushbutton*. Kalau anda melihat pada aplikasi-aplikasi *Window*, dipojok kiri atas selalu ada daftar menu yang dapat digunakan dengan cara meng-klik pada menu yang disorot. Nah *uimenu* digunakan untuk membuat daftar menu seperti itu. Berikut ini adalah sintak umumnya dalam pemrograman MATLAB:

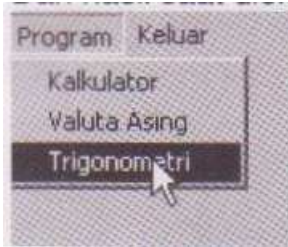
```
nama=uimenu('parent',[NamaFigure],...
'Label',[Teks Menu],...
'Callback',[MATLAB Skrip]);
```

Berikut ini adalah contoh potongan skrip yang digunakan dalam

pemrogramannya :

```
menu1=uimenu('parent',win1 ....
    'Label','Program');
menull= uimenu('parent',menul,...
    'Label','Kalkulator',...
    'Callback','latihan23');
```

Dan hasil saat dieksekusi adalah daftar menu seperti berikut :



Obiek Axes

Objek *axes* dalam pemrograman MATLAB sangat penting untuk melakukan visualisasi data. Tanpa menggunakan objek *axes*, kita tidak dapat menampilkan hasil eksekusi fungsi *plot*, *mesh*, *contour* dan lain-lain, karena objek *axes* adalah medianya. Sintak umum menggunakan objek *axes* adalah sebagai berikut :

```
nama=axes('parent',[objek parent],...
    'units','<'points' 'pixel'>',...
    'position',[Left Top Width Height],...
    'xgrid','<'on' 'off'>' ....
    'ygrid', '<'on' 'off'>' ....
    'xcolor',[R G B],...
    'ycolor', [R G B],...
    'fontsize',[numerik],...
    'color', [R G B]);
```

Berikut adalah potongan program yang menggunakan skrip axes :

```
grafikl=axes('parent',winl,...
    'units','points',...
    'position',[250 80 240 180],...
    'xgrid','on',...
    'ygrid','on',...
    'xcolor',[0.4 0 .15],...
```

```
'ycolor',[0.4 0 .15],...
'fontsize',8,...
'color',[1 1 1]);
```

Properti Callback Sebagai Media Interaksi

Agar objek-objek yang kita buat dapat digunakan untuk mengerjakan perintah-perintah pemrograman sebagaimana mestinya, ada media yang disediakan disetiap objek untuk itu. Medianya adalah melalui properti *callback*. Dimana nilai properti *callback* akan dieksekusi sebagai program Matlab ketika objek pemilikinya dikenai sesuatu (pada *pushbutton* misalnya diklik). Pada properti *callback* ini kita buat skrip MATLAB sebagaimana biasa (skrip MATLAB) atau kita tuliskan nama file MATLAB yang akan dijalankan. **Mengetahui fungsi *callback*** adalah **kunci pertama** pemrograman *Window* menggunakan MATLAB.

Berikut adalah cuplikan contoh skrip untuk *callback* :

```
edit1=uicontrol('parent',win1,...
'units' 'points',...
'position',[110 140 100 15],...
'backgroundcolor',[1 1 1],...
'style','Pushbutton',... 'string','Exit',...
'fontname','arial',...
'fontsize',10 ....
'Callback', ['x [0:10:180];',...
'y sin(x*pi/180);',...
'plot(x,y,'-r')'l];
```

Bentuk lainnya, adalah sebagai berikut :

```
Menu14=uimenu('parent',menul,...
'Label','Tes',...
'Callback','latihan20');
```

Pada contoh terakhir, 'latihan20' adalah nama file MATLAB, perhatikan cara penulisannya tidak boleh dengan ekstensinya (latihan20.m).

Interaksi Antar Objek Visual (Fungsi Get dan Set)

Mengambil nilai properti dari satu objek dan menggunakannya untuk mengisi nilai properti pada objek lain, itulah hakekat pemrograman *Window*.

Mengetahui metoda Interaksi antar objek *visual* dalam pemrograman MATLAB adalah **kunci kedua** untuk pemrograman *Window*. MATLAB menyediakan dua buah fungsi untuk itu, yaitu **get** dan **set**. Bayangkan, kita memiliki 3 buah objek *edit* sebagai berikut :



Kita memberikan input angka pada edit1 dan edit2, kemudian angka tersebut dijumlahkan dan hasilnya dimunculkan ke edit3.

Fungsi **get** kita gunakan untuk mengambil nilai properti dari suatu objek. Fungsi ini dapat dipadukan dengan fungsi konversi *string* ke *numeric* atau sebaliknya, sesuai kebutuhan pengolahan datanya. Sintak dasarnya adalah sebagai berikut :

```
X=get([NamaObjek],[Property]);
```

Maka skrip untuk mengambil input dari edit1 dan edit2 adalah

```
a=str2num(get(edit1,'String'));  
b=str2num(get(edit2,'String'));  
c=a+b;
```

Sedangkan fungsi **set** kita gunakan untuk memberikan suatu nilai pada properti objek tertentu. Fungsi ini juga dapat dipadukan dengan fungsi konversi sebagai mana fungsi **get**. Sintak dasarnya adalah sebagai berikut :

```
set([NamaObjek],[Property],[Nilai Baru]);
```

Maka skrip untuk menampilkan output pengolahan kita ke edit3 adalah sebagai berikut :

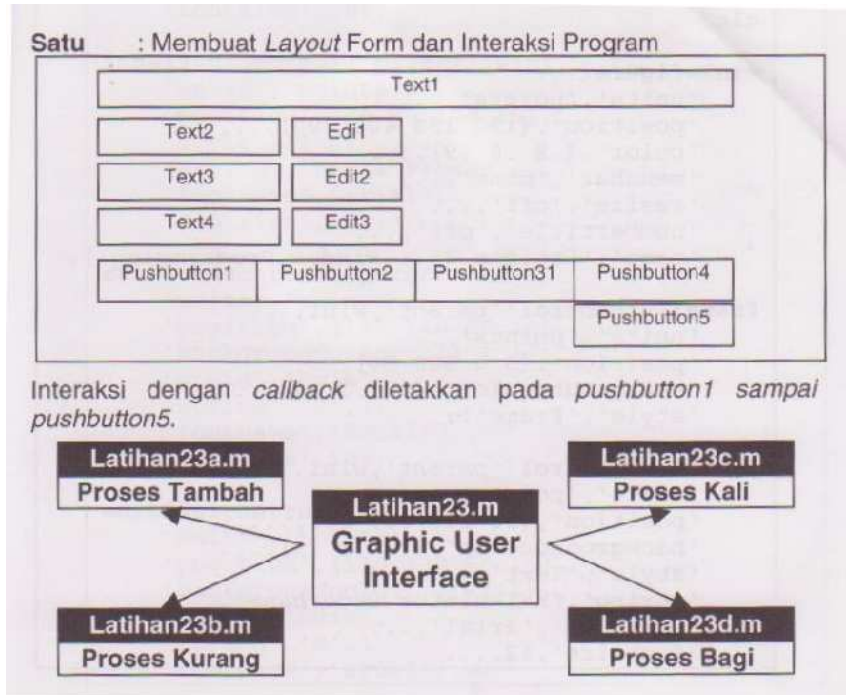
```
set(edit3,'String',num2str(c));
```

Sekarang kita sudah mendapatkan cara penulisan dan konsep-konsep untuk membangun aplikasi *Window* dengan menggunakan MATLAB. Tiba

saatnya kita lanjutkan untuk menggunakan skrip-skrip tersebut dalam satu program yang utuh.

Tutorial #1 : Program Kalkulator Sederhana

Program latihan pertama kita adalah membuat **Program Kalkulator Sederhana**. Untuk membuat program ini, kita rencanakan beberapa tahap.



Gambar 10.2 Contoh pembuatan layout form

Dua : Membuat Program MATLAB dalam 5 bush file terpisah seperti pada diagram

Kita mulai dengan membuat *file* pertama (*Graphic User Interface*). ikuti langkah-langkah berikut ini.

1. Pada *Command Window*, ketikkan :

```
>> edit
```

2. Tekan *Enter*, selanjutnya muncul *MATLAB Editor* dan anda ketiklah program dibawah berikut :

```
% -----  
% Program Latihan 23
```

```
% MATLAB Window Programming
% Oleh : gunay
clear all;
cic;

win1=figure( ...
    'units' 'points' ...
    'position',[130 190 400 200],...
    'color',[.8 .8 .91] ....
    'menubar','none',...
    'resize','off' ....
    'numbertitle','off',...
    'name','Latihan 23 : Window Programming');

framel=uicontrol('parent',win1,...
    'units','points',...
    'position',[0 0 500 60],...
    'backgroundcolor',[.3 .3 .4],...
    'style','Frame');

labell=uicontrol('parent',win1 ....
    'units','points',...
    'position',[30 170 300 20] ....
    'backgroundcolor',[.8 .8 .9],...
    'style','Text',...
    'string','Kalkulator Sederhana' ....
    'fontname','arial',...
    'fontsize',12,...
    'fontweight','bold',...
    'foregroundcolor',[0 0 0]);

label2=uicontrol('parentl,win1,...
    'units','points',...
    'position',[30 140 100 15],...
    'style','Text' ....
    'string','Data ke 1',...
    'fontname','arial',...
    'fontsize',10);
```

```
label3=uicontrol('parenL',winl ....
    'units','points' ....
    'position',[30 120 100 15],...
    'style','Text',...
    'string','Data ke 2',...
    'fontname','arial',...
    'fontsize',10);
```

```
label4=uicontrol('parent',winl ....
    'units' 'points' ....
    'position',[30 90 100 15],...
    'style','Text',...
    'string','hasil Proses',...
    'fontname','arial',...
    'fontsize',10);
```

```
edit1=uicontrol('parent',winl....
    'units','points',...
    'position',[130 140 60 15],...
    'backgroundcolor',[1 1 1],...
    'style','Edit',...
    'string','0',...
    'fontname','arial',...
    'fontsize',10);
```

```
edit2=uicontrol('parent',winl,...
    'units','points',...
    'position',[130 120 60 15],...
    'backgroundcolor',[1 1 1],...
    'style','Edit',...
    'string','0',...
    'fontname','arial',...
    'fontsize',10);
```

```
edit3=uicontrol('parent',winl,...
    'units','points',...
    'position',[130 90 60 15],...
    'backgroundcolor',[1 1 1],...
```

```

'style','Edit',...
'string','0' ....
'fontname','arial',...
'fontsize',10);

tomtambah=uicontrol('parent',winl...
'units','points' ....
'position',[30 40 80 15],...
'style','pushbutton',...
'callback','latihan23a',...
'string','Tambah',...
'fontname','arial',...
'fontsize',10);

tomkurang=uicontrol('parent',winl...
'units','points',...
'position',[110 40 80 15],...
'style','pushbutton' ....
'callback','latihan23b' ....
'string','Kurang' ....
'fontname','arial',...
'fontsize',10);

tomkali=uicontrol('parent',winl ....
'units','points' ....
'position',[190 40 80 15],...
'style','pushbutton' ....
'callback','latihan23c',...
'string','Kali',...
'fontname','arial',...
'fontsize',10);

tombagi=uicontrol('parent',winl ....
'units','points',...
'position',[270 40 80 15],...
'style','pushbutton' ....
'callback','latihan23d',...
'string','Bagi',...
'fontname','arial',...
'fontsize',10);

tomtutup=uicontrol('parent',winl,...
'units','points',...
'position',[270 20 80 15],...
'style','pushbutton',...
'string','Tutup',...

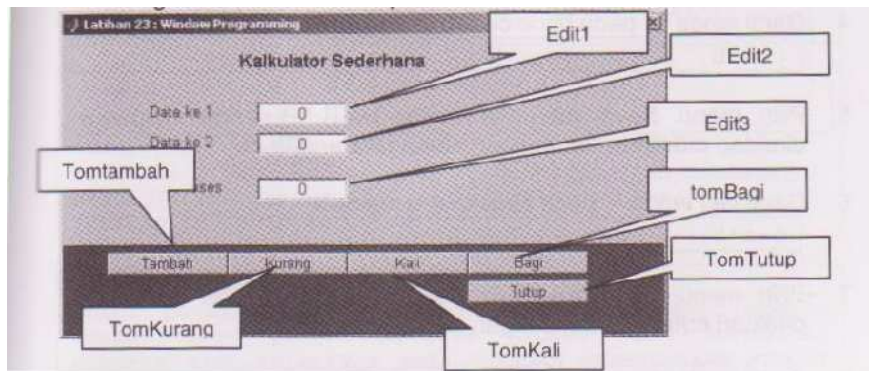
```

```
'fontname','arial',...
'fontsize',10...
'callback','close');
```

- Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **latihan23.m**.
- Pastikan direktori penyimpanan *file* anda sudah terdapat didalam daftar pencarian direktori MATLAB (lihat Bab 2 Penting Untuk Pemula). Lalu ketiklah nama file latihan23 tanpa ekstensi :

```
»latihan23
```

- Tekan *Enter*, selanjutnya program akan dijalankan dan menghasilkan *Window* seperti berikut ini :



- Selesailah program pertama.

Nah, untuk program-program selanjutnya adalah program yang berisi skrip interaksi antar objek yang telah kita definisikan pada *file* latihan23.m. Perhatikan nama-nama objeknya.

Sekarang kita lanjutkan dengan membuat *file* program-program proses. Ikuti langkah-langkah berikut :

- Pada *Command Window*, ketikkan :

```
>> edit
```

- Tekan *Enter*, selanjutnya muncul *MATLAB Editor* dan anda ketiklah program dibawah berikut :

```
% -----
```

```
% Program Tambah Untuk Latihan 23
% -----

a=str2num(get(edit1,'String'));
b=str2num(get(edit2,'String'));
c=a+b;
```

```
set(edit3,'String',num2str(c));
```

3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **latihan23a.m**.
4. Ganti tanda '+' pada baris `c=a+b` menjadi

```
c=a-b;
```

5. Pilih menu Save As pada editor MATLAB, anda simpan di direktori **c:/latihanku**, dengan nama **latihan23b.m**.
6. Ganti lagi tanda '-' pada baris `c=a-b` menjadi

```
c=a*b;
```

7. Pilih menu Save As pada editor MATLAB, anda simpan di direktori **c:/latihanku**, dengan nama **latihan23c.m**.
8. Ganti tanda '*' pada baris `c=a*b` menjadi

```
c=a/b;
```

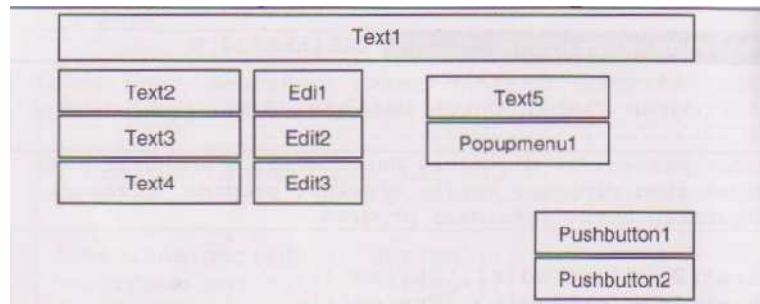
9. Pilih menu Save As pada editor MATLAB, anda simpan di direktori **c:/latihanku**, dengan nama **latihan23d.m**.
10. Pastikan direktori penyimpanan file anda sudah terdapat Jaem daftar pencarian direktori MATLAB (lihat Bab 2 Penting Pemula).
11. Selesailah pembuatan 4 program untuk proses.

Tiga : Jalankan Program Utama (latihan23.m). Selesai.

Tutorial #2 : Program Valuta Asing -> Rupiah

Program latihan kedua kita adalah membuat **Program Valuta Asing -> Rupiah**. Untuk membuat program ini, kita rencanakan beberapa tahap.

Satu : Membuat Layout Form dan Interaksi Program



Interaksi dengan *callback* diletakkan pada *pushbutton1* dan *pushbutton2*.



Dua: Membuat Program MATLAB dalam 2 buah *file* terpisah seperti pada diagram

Dari pengalaman tutorial sebelumnya, mungkin anda bisa langsung mencoba sendiri. Tapi kalau belum terbiasa, mari kita ikuti langkah-langkahnya. Berikut adalah langkah-langkah membuat *file* pertama (*User Interface*).

1. Pada *Command Window*, ketikkan :

```
>> edit
```

2. Tekan *Enter*, selanjutnya muncul *MATLAB Editor* dan anda ketiklah program dibawah berikut :

```
% -----
% Program Latihan 24
% MATLAB Window Programming
% Oleh : gunay
% -----
clear all;
clc;

win1=figure( ...
    'units','points' ....
    'position',[130 190 400 200],...
    'color',[.8 .8 .9],...
```



```

        'menubar','none',...
        'resize','off',...
        'numbertitle','off',...
        'name','Latihan 24 : Window Programming');

frame1=uicontrol('parent',win1 ....
    'units','points' ....
    'Position',[0 0 500 60],...
    'backgroundcolor',[.3 .3 .4],...
    'style','Frame');

label1=uicontrol('Parent',win1 ....
    'units','points',...
    'position',[30 170 300 20],...
    'backgroundcolor',[.8 .8 .9],...
    'style','Text',...
    'string','Valuta Asing -- > Rupiah',...
    'fontname','arial',...
    'fontsize',12,...
    'fontweight','bold',...
    'foregroundcolor',[0 0 0]);

label2=uicontrol('parent',win1 ....
    'units' 'points',...
    'position',[220 140 100 15],...
    'style','Text',...
    'string','Mata Uang Asing',...
    'fontname','arial' ....
    'fontsize',10);

popu1=uicontrol('parent',win1 ....
    'units','points' ....
    'position',[220 130 100 10],...
    'backgroundcolor',[1 1 1],...
    'style','popupmenu',...
    'string','US. Dollar MLY. Ringgit SGP

```

```
Dollar JPN. Yen',...
'fontname','arial',...
'fontsize',10);

label2=uicontrol('parent',winl,...
'units','points',...
'position',[30 140 100 15],...
'style','Text',...
'string','Jumlah Uang',...
'fontname','arial',...
'fontsize',10);

label3=uicontrol('parent',winl ....
'units','points',...
'position',[30 120 100 15],...
'style','Text',...
'string','Kurs',...
'fontname','arial',...
'fontsize',10);

label4=uicontrol('parent',winl,...
'units','points',...
'position',[30 90 100 15],...
'style','Text',...
'string','Jumlah Rupiah',...
'fontname','arial',...
'fontsize',10);

edit1=uicontrol('parent',winl ...
'units','points',...
'position',[130 140 60 15],...
'backgroundcolor',[1 1 1],...
'style','Edit' ....
'string','0',...
'fontname','arial',...
'fontsize',10);
```

```

edit2=uicontrol('parent',winl,...
    'units','points',...
    'position',[130 120 60 15],...
    'backgroundcolor',[1 1 1],...
    'style','Edit',...
    'string','0',...
    'fontname','arial',...
    'fontsize',10);

edit3=uicontrol('parent',winl,...
    'units','points',...
    'position',[130 90 60 15],...
    'backgroundcolor',[1 1 1],...
    'style','Edit' ....
    'string','0',...
    'fontname','arial',...
    'fontsize',10);

tomhitung=uicontrol('parent',winl ....
    'units','points' ....
    'position',[270 40 80 15],...
    'style','pushbutton' ....
    'callback','latihan24a',...
    'string','Hitung',...
    'fontname','arial',...
    'fontsize',10);

tomtutup=uicontrol('parent',winl,...
    'units','points',...
    'position',[270 20 80 15],...
    'style','pushbutton' ....
    'string','Tutup',..
    'fontname','arial',...
    'fontsize',10,...
    'callback','close');

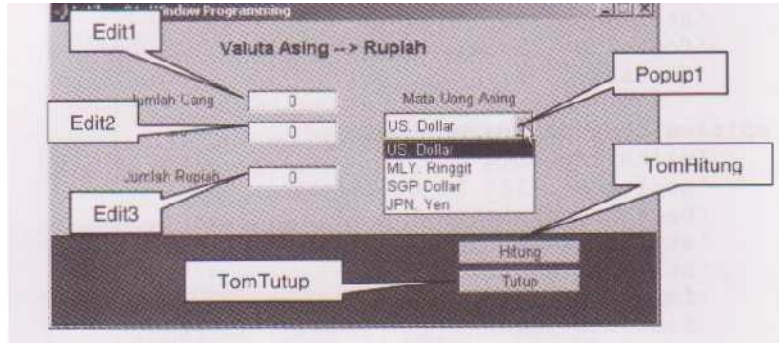
```

3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **latihan24.m**.
4. Pastikan direktori penyimpanan file anda sudah terdapat didalam daftar pencarian direktori MATLAB (lihat Bab 2 Penting Untuk Pemula). Lalu ketiklah nama file

latihan24 tanpa ekstensi:

```
>>latihan24
```

5. Tekan *Enter*, selanjutnya program akan dijalankan dan menghasilkan sebagai berikut



6. Selesai pembuatan program pertama.

Kita lanjutkan dengan membuat program kedua dengan memperhatikan nama objek yang telah kita definisikan, mengikuti langkah-langkah berikut :

1. Pada *Command Window*, ketikkan :

```
>> edit
```

2. Tekan *Enter*, selanjutnya muncul *MATLAB Editor* dan anda ketiklah program dibawah berikut :

```
%-----
% Program Hitung Valuta Untuk Latihan 24
% -----
uangl= str2num(get(editl,'String'));

pilihan = get(popupl,'Value');
switch pilihan
case 1
```

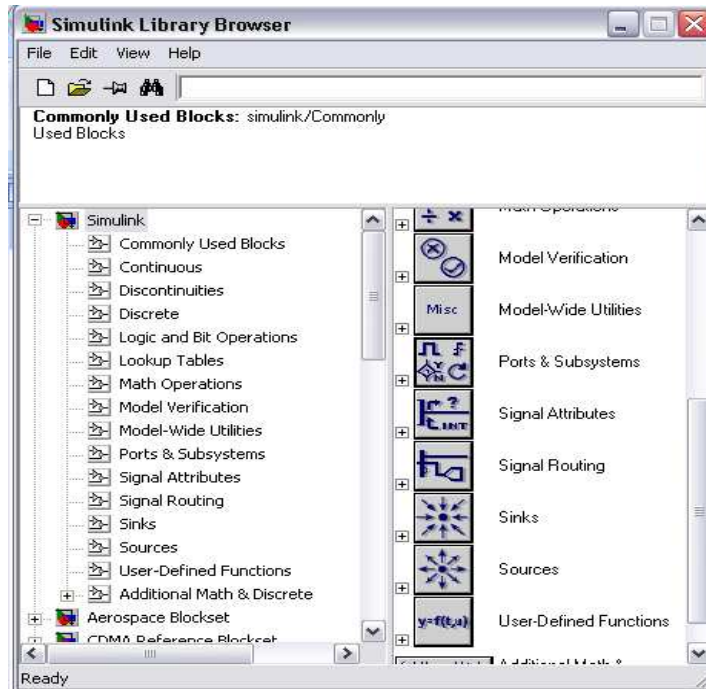
```
% US dollar
kurs=9900;
uang2 = uang1 * kurs ;
set(edit2,'String',num2str(kurs));
set(edit3,'String',num2str(uang2));
case 2
% Mly Ringgit
kurs=1450;
uang2 = uang1 * kurs ;
set(edit2,'String',num2str(kurs));
set(edit3,'String',num2str(uang2));
case 3
% Sgp Dollar
kurs=4600;
uang2 = uang1 * kurs ;
set(edit2,'String',num2str(kurs));
set(edit3,'String',num2str(uang2));
case 4
% Jpn Yen
kurs=3500;
uang2 = uang1 * kurs ;
set(edit2,'String',num2str(kurs));
set(edit3,'String',num2str(uang2));
end;
```

3. Setelah selesai mengetik program diatas, anda simpan di direktori **c:/latihanku**, dengan nama **latihan24a.m**.
4. Pastikan direktori penyimpanan *file* anda sudah terdapat didalam daftar pencarian direktori MATLAB (lihat Bab 2 Penting Untuk Pemula).
5. Selesailah pembuatan program kedua.

Tiga : Jalankan Program Utama (latihan24.m). Selesai.

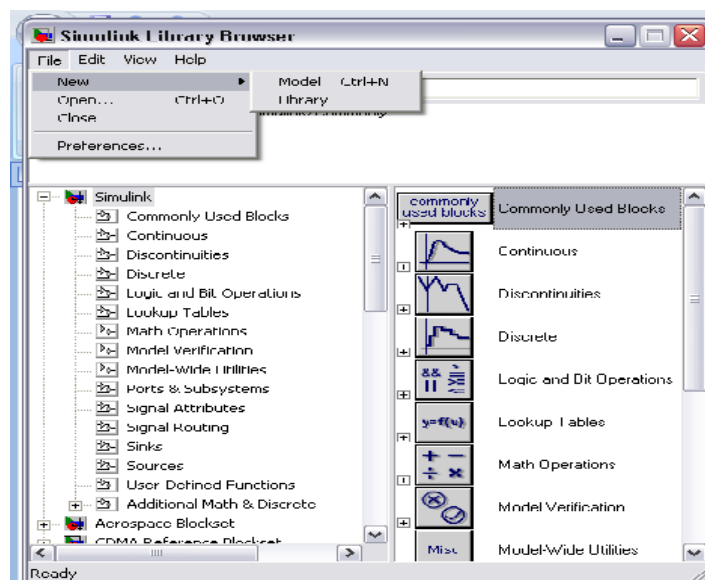
11.3 Simulink Library Browser

Setelah menekan icon simulink maka akan muncul jendela library browser yang merupakan tempat dari bagian simbol-simbol pemrograman.



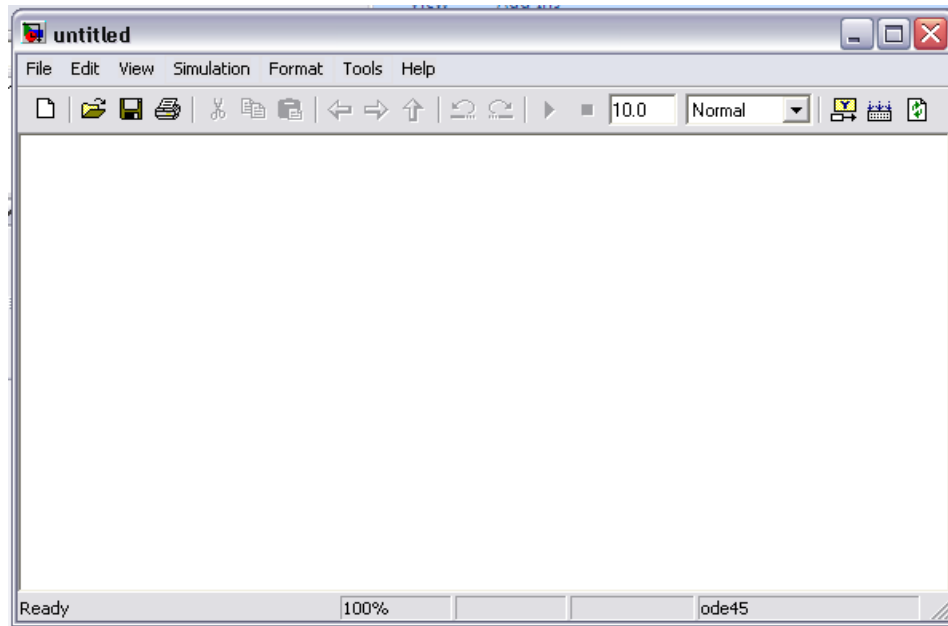
Gambar 11.2 Library Browser

Model dapat dibuka melalui file – new – model seperti berikut:



Gambar 11.3 Membuka Model

Model merupakan command/ruang kerja dari simulink, yakni dengan men drag/menknarik simbol-simbul pada simulink library yang diperlukan ke dalam command model. Akan tetapi disini sangat diperlukann kemahiran dalam memodelkan persamaan matematis sebelum menyusun dalam bentuk model simulasi dengan simulink. Adapun command model sebagai berikut:

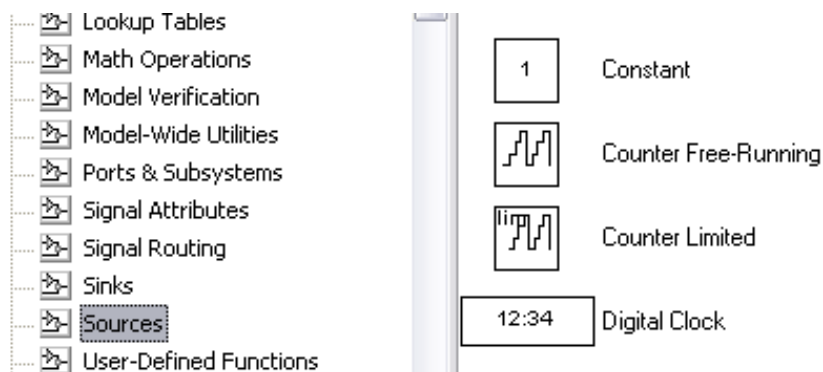


Gambar 11.4 Comand Model

11.4 Tool Simulink Library Browser

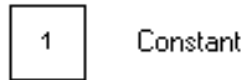
Pada library browser terdapat banyak tool yang memiliki fungsi dan kegunaan yang berbeda-beda. Seperi: Math Operations, Sinks, Sources, Continous dan lain sebagainya.

Untuk lebih jelasnya dapat dilihat pada [demos- matlab-simulink](#).

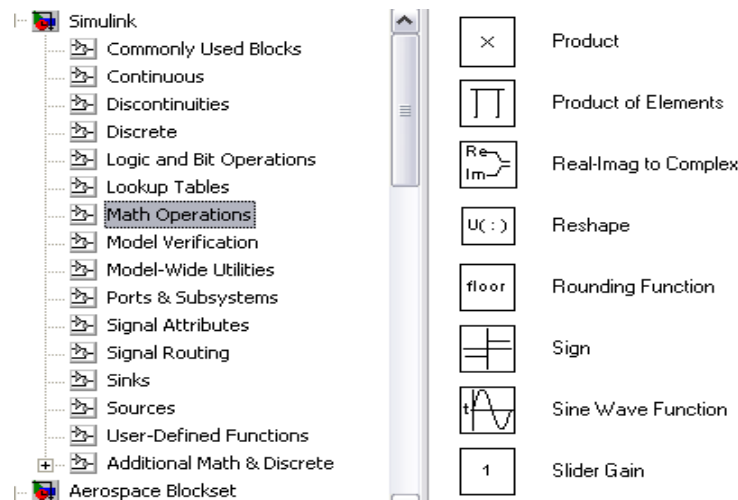


Gambar 11.5 Contoh tool sources.

Diatas merupakan contoh beberapa isi dari tool sources. Dimana yang paling sering digunakan adalah simbol “**constant**”.

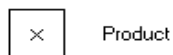


Simbul ini digunakan sebagai tempat menginput nilai-nilai daripada constanta-constant matematika.



Gambar 11.6 Contoh tool math operation

Math operations merupakan tool operasi matematika, baik itu simbol perkalian, penjumlahan, pengurangan, pembagian, akar dan lain sebagainya.



Product merupakan simbol matematis untuk perkalian.



Sum merupakan simbol matematis untuk penjumlahan dan pengurangan.

Masih banyak simbol-simbol yang lain dalam library browser, silakan anda pelajari lebih lanjut. Parameter-parameter dalam simbol-simbol tersebut dapat diubah dengan mengklik kanan kemudian akan keluar command parameter. Dari command tersebut anda dapat merubah parameter-parameter sesuai yang diperlukan.

Contoh:

Suatu tekanan pada dasar tangki ditentukan dengan rumusan matematis seperti:

$$P_2 = \gamma_{H_2O} \cdot h_{H_2O} + P_1$$

Dimana:

P_1 = tekanan pengukuran

$$= 721 \text{ lb/ft}^2$$

γ = Berat jenis air

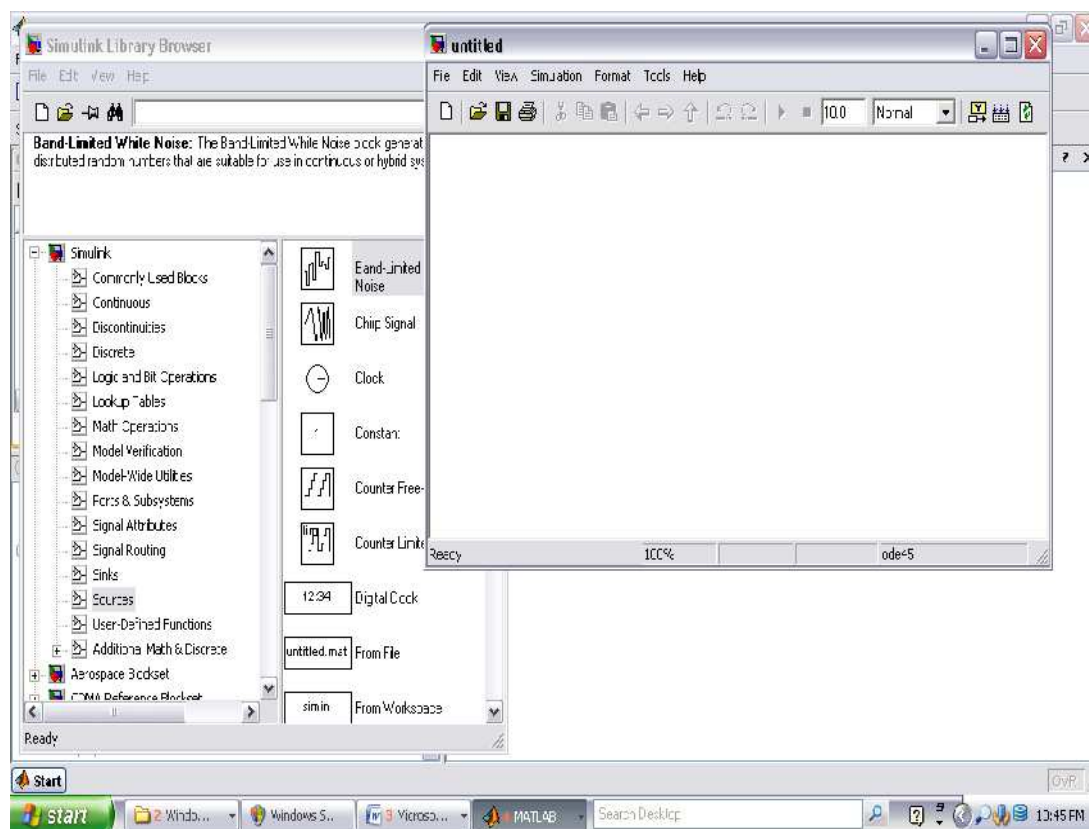
$$= 62,4 \text{ lb/ft}^3$$

h = tinggi air

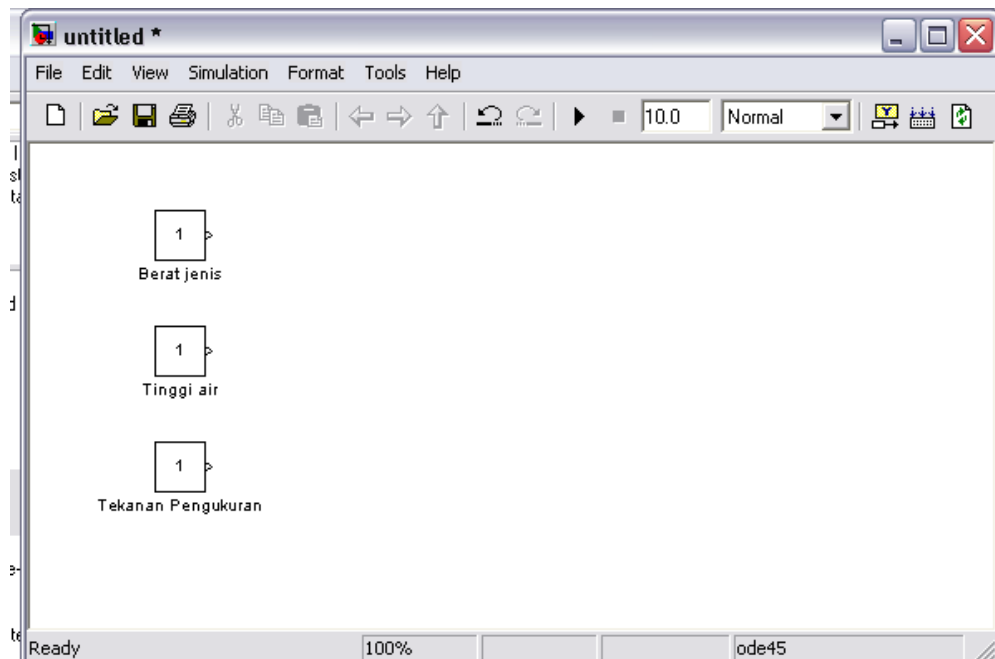
$$= 3 \text{ ft}$$

Maka persamaan matematis tersebut dapat diselesaikan dengan simulink, sebagai berikut:

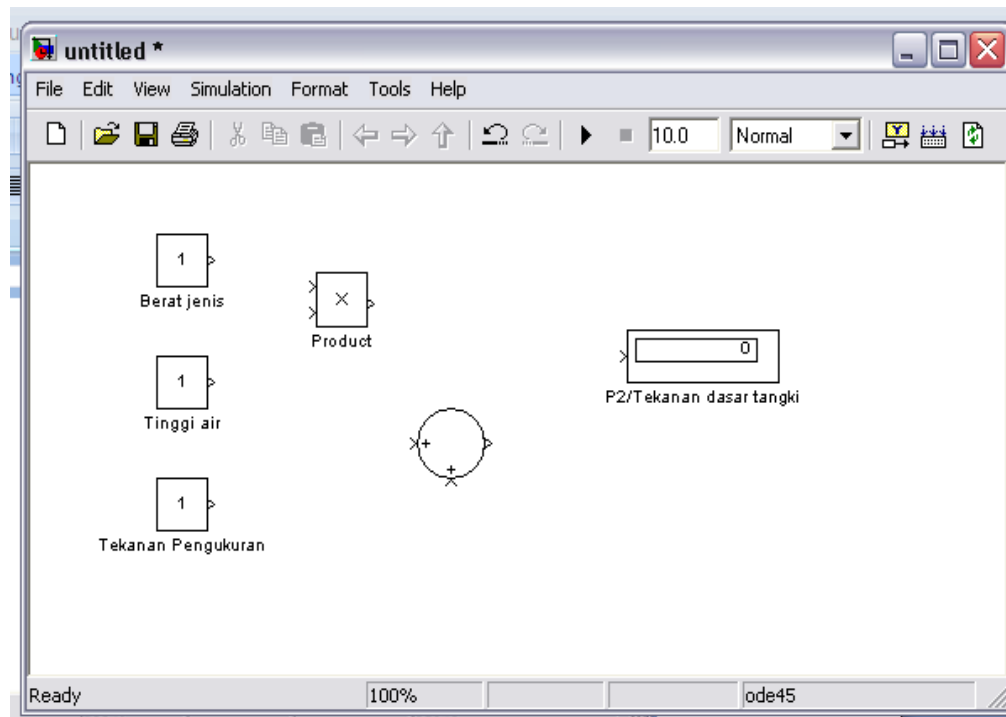
Buka simulink kemudian library browser dan command model.



Setelah itu ambillah 3 simbol constant dari sources dan ditarik ke command model dan beri nama masing-masing constants seperti dibawah:

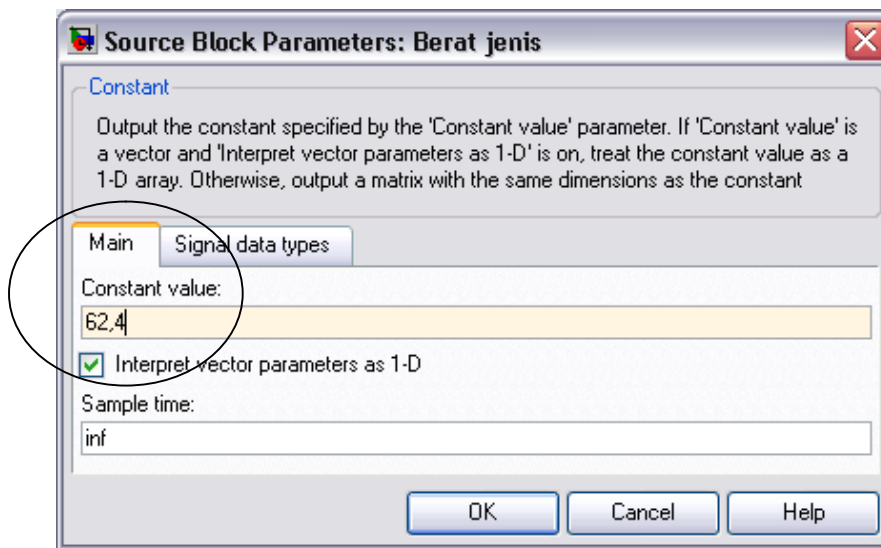
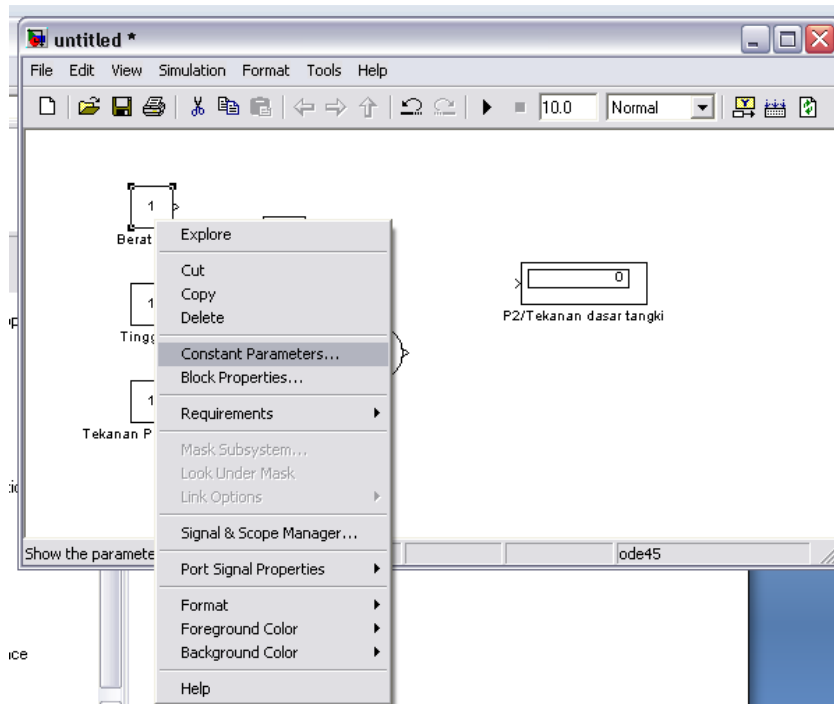


Setelah itu kita memerlukan sum dan product untuk penjumlahan dan perkalian konstanta (ambil di tool math operation), ambil pula simbol display untuk memunculkan nilai P2 (ambil pada tool sink):



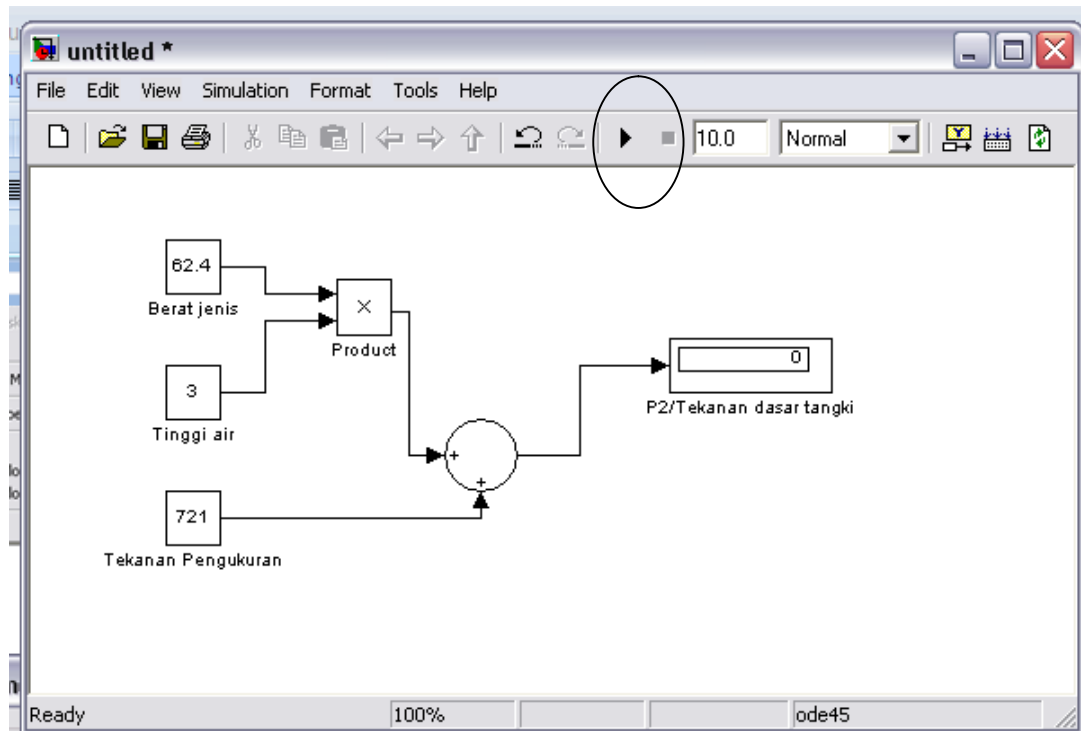
Ganti parameter constant sesuai dengan soal:

Misalnya:

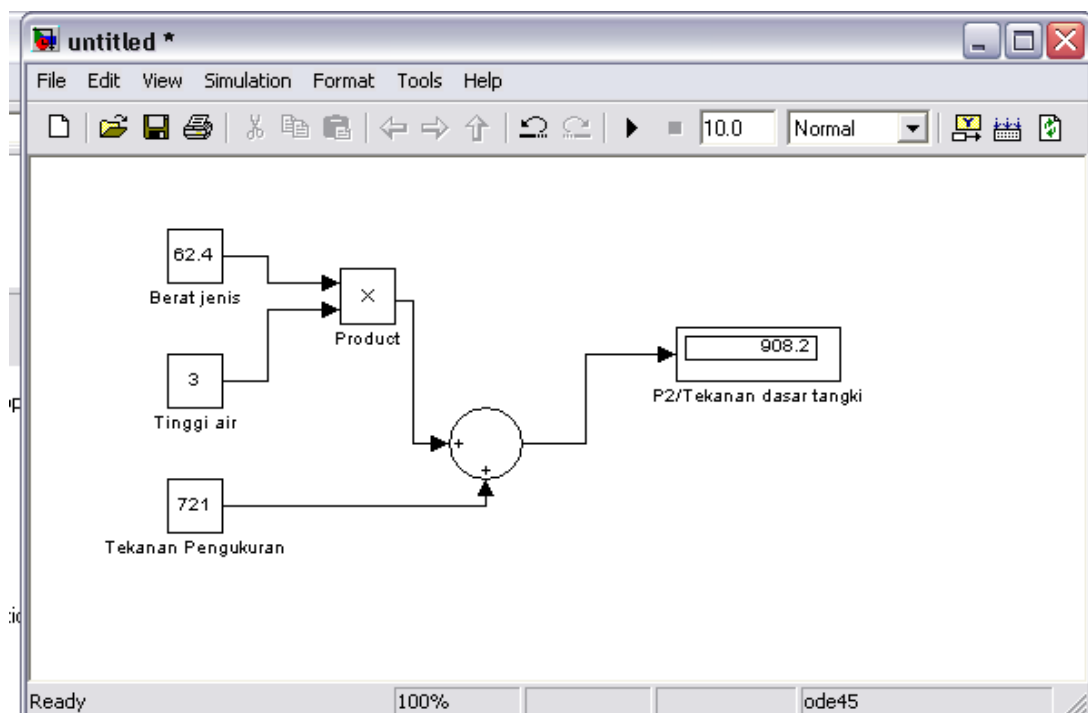


Setelah muncul source blok parameter diatas, ganti nilai parameternya sesuai nama constant yang ada lalu tekan ok. Maka nilai constant akan bernilai sesuai dengan soal.

Setelah itu hubungkan sesuai dengan persamaan matematisnya:

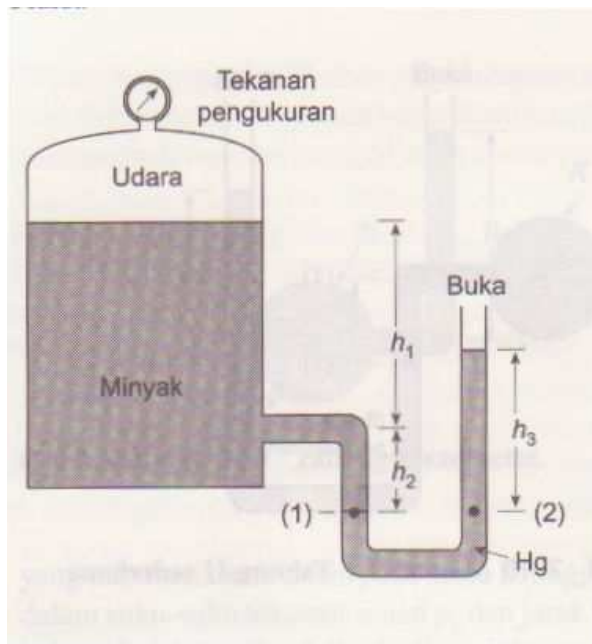


Untuk memperoleh nilai dari P2 maka perlu menekan tool run seperti yang dilingkari diatas. Maka akan muncul nilai P2 seperti dibawah:



Soal:

Sebuah tangki tertutup berisi udara bertekanan dan minyak ($SG_{\text{minyak}} = 0,9$) seperti pada gambar: Sebuah barometer tabung U yang menggunakan air raksa ($SG_{\text{raksa}} = 13,6$) dihubungkan ke tangki tersebut seperti pada gambar:



Tentukan bacaan tekanan dari alat ukur!!!! (Selesaikan dengan simulink)

Dimana

$$P_1 = P_{\text{udara}} + \text{Berat Jenis minyak} + (h_1 + h_2)$$