Bagian 1 PENDAHULUAN

1. Karakter

a. Numerik (Angka) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
b. Alfabet (Huruf) : A, B, C, D, ……., Z
a, b, c, d, ……., z
c. Khusus : %, ’, +, -, *, /, \, dll.

2. Ekspresi

Merupakan gabungan dari karakter yang dibedakan atas:

1. ekspresi numerik
   gabungan dari karakter numerik, misalnya: 12, 4567, 0.25, dll.
2. ekspresi string
   gabungan dari karakter alfabet; atau gabungan dari karakter alfabet dan numerik, misalnya: metanol, reaktor1, A25, dll.

3. Variabel

→ nama yang diberikan untuk mewakili suatu data, baik berupa masukan data atau merupakan hasil perhitungan

→ aturan penulisan variabel:

1. maksimal terdiri dari 31 karakter
2. harus diawali dengan huruf
3. tidak boleh mengandung spasi dan tanda baca (karakter khusus)
4. tidak boleh menggunakan *key words*
5. dibedakan antara huruf besar dan huruf kecil, misalnya: Nama, NAMA, NamA, NaMa, dsb. merupakan variabel yang berbeda.

4. Memasukkan Data

Terdapat dua cara untuk memasukkan data yaitu:

a. inisialisasi data, dimana data diberikan secara langsung pada saat pembuatan program
   Bentuk umum:
   variabel = ekspresi
b. menggunakan perintah *input*, dimana data diberikan pada saat program di-*running* sehingga bersifat interaktif
   Bentuk umum:
   variabel = input(‘teks’) ← numerik dan string

   atau

   variabel = input(‘teks’, ‘s’) ← string

pada layar akan tampil apa yang tertulis di antara tanda petik (teks), kemudian komputer menanti masukan data yang diikuti dengan menekan *enter* melalui *keyboard*

Contoh 1.1 dan 1.2 berikut ini menampilkan program perhitungan sederhana dimana langkah memasukkan data dilakukan dengan menggunakan kedua cara tersebut. Kedua contoh tersebut dijalankan pada layar utama Matlab yang disebut *command window*.

Contoh 1.1: Perhitungan luas persegi panjang A = p.l dengan cara inisialisasi data.

```
>> data = 'persegi panjang'
data =
persegi panjang
>> panjang = 4
panjang =
     4
>> lebar = 3
lebar =
     3
>> luas = panjang * lebar
luas =
    12
```

Contoh 1.2: Perhitungan luas persegi panjang A = p.l dengan cara input data.

```
>> data = input('Masukkan jenis bangun geometri yang akan dihitung: ')
Masukkan jenis bangun geometri yang akan dihitung: 'persegi panjang'
data =
persegi panjang

>> panjang=input('Masukkan data panjang : ')
Masukkan data panjang : 4
panjang =
     4

>> lebar=input('Masukkan data lebar : ')
Masukkan data lebar : 3
lebar =
     3

>> luas = panjang * lebar
luas =
    12
```

Kedua cara di atas dapat diakhiri dengan tanda (;). Penggunaan tanda (;) akan meniadakan tampilnya data setelah penekanan *enter* atau pada saat *running program*.

5. Tampilan Hasil

a. Bila ingin menampilkan nilai dari suatu variabel

disp(A)

→ dimana A adalah variabel; hasil yang ditampilkan adalah nilai yang tersimpan dalam variabel A

b. Bila ingin menampilkan teks atau string

disp('teks')

→ hasil yang ditampilkan adalah apa yang tertulis di antara tanda petik (teks)

c. Bila ingin menampilkan gabungan teks dan nilai dari suatu variabel, gunakan tanda kurung siku dimana nilai numerik harus dikonversi ke bentuk string terlebih dahulu dengan menggunakan fungsi *num2str* (*number to string*)

Contoh 1.3:

```
x = 2.678;
disp(['Nilai x adalah = ', num2str(x), 'pada iterasi ini'])
```

Dapat pula menggunakan perintah **fprintf**

Bentuk umum:

fprintf('file name', 'format string', list)

- *file name* bersifat *optional* (dapat ditulis atau tidak)
- *list* adalah daftar nama variabel yang dipisahkan dengan tanda koma (,)
- *format string* adalah format/bentuk tampilan:

  %P.Qe untuk bentuk eksponensial
  %P.Qf untuk bentuk *fixed point*
  \n untuk membentuk baris baru (kelang)

  dimana P dan Q merupakan *integer* (bilangan bulat).

Contoh 1.4:

```
x = 1007.46;
y = 2.1278;
k = 17;
fprintf('\n x  = %8.2f  y = %8.6f  k = %2.0f \n', x, y, k)
```

Latihan 1.1: Tambahkanlah tanda (;) pada akhir semua baris program pada Contoh 1.1. Kemudian aturlah tampilan hasilnya dengan menggunakan perintah disp atau fprintf.

6. Penggabungan *Input Data*

Beberapa data yang dimasukkan yang ditulis dalam beberapa baris program dapat digabungkan dengan memberikan tanda koma (,) atau titik koma (;) sebagai pemisah.

Contoh 1.5: *Input data* x, y, dan k pada contoh di atas yang ditulis dalam 3 baris program dapat digabungkan menjadi 1 baris program dengan salah satu cara berikut:

```
x = 1007.46, y = 2.1278, k = 17
x = 1007.46; y = 2.1278; k = 17;
```

7. Komentar

Semua teks yang diawali dengan tanda persen '%' dianggap sebagai pernyataan atau komentar atau keterangan atau catatan.

Tujuannya adalah untuk memberi keterangan agar lebih mudah memahami maksud dan kegunaan suatu bagian program.

Sejauh ini, semua pekerjaan dilakukan melalui sebuah layar yang disebut *command window* dimana perintah dapat dieksekusi secara langsung satu per satu. Ada pula layar lain yang dapat menyimpan semua perintah yang dibuat untuk kemudian dieksekusi secara keseluruhan. Layar tersebut adalah layar *M-File*.

Contoh 1.6: Hendak ditentukan volume suatu gas berdasarkan persamaan gas ideal P.V = n.R.T, dimana P = tekanan (atm), V = volume (ltr), n = jumlah mol (gmol), T = temperatur (K), dan R = konstanta gas (0,08206 ltr.atm/gmol.K).

```
% Perhitungan volume gas
% Berdasarkan persamaan gas ideal
% PV = nRT

% Masukan Data
nama_gas = 'metanol'
P = 1
T = 373
n = 10
R = 0.08206

%Perhitungan
V = n*R*T/P
```

Program di atas disimpan dalam sebuah *m-file* yang diberi nama ideal.m. Untuk mengeksekusinya dapat dilakukan dengan salah satu cara berikut:

1. Tekanlah tombol F5 pada layar *m-file*, kemudian pindahlah ke *command window* untuk melihat hasil eksekusi (*running program*)
2. Pindahlah ke *command window*, kemudian ketikkanlah nama *file* yang akan dieksekusi (di-*running*).

Running program berikut menggunakan cara yang kedua:

```
>> ideal
nama_gas =
metanol

P =
     1

T =
   373

n =
    10
```

```
R =
   0.0821

V =
  306.0838
```

Catatan: - Data yang dimasukkan haruslah konsisten dalam penggunaan satuan
- Analisis hasil diserahkan kepada pengguna program, dimana pada program di atas, diketahui bahwa volume gas metanol yang terdapat sejumlah 10 gmol pada temperatur 373 K dan tekanan 1 atm adalah 306,0838 ltr.

Latihan 1.2: Ubahlah program pada Contoh 1.6 di atas
- Berilah komentar berupa satuan dari masing-masing masukan data
- Gunakan bentuk *input* dalam memasukkan data
- Aturlah tampilan hasil agar lebih representatif.

Latihan 1.3: Buatlah program perhitungan luas persegi panjang pada layar *m-file*.

8. *Key Words* atau *Reserved Words*

→ kata-kata yang mempunyai arti khusus dalam MATLAB:
merupakan variabel khusus atau fungsi

a. Variabel Khusus

Misalnya:	pi	π, 3.14
	ans	nama variabel untuk hasil apapun (*default*)
	inf	bilangan tak berhingga, 1/0
	NaN atau nan	*not a number*, 0/0
	i dan j	i = j = √-1

b. Fungsi

Matlab banyak menyediakan fungsi (*function*) yang dapat digunakan untuk berbagai perhitungan. Fungsi-fungsi yang telah disediakan oleh Matlab (*built-in*), seperti contoh berikut ini:

- **Fungsi-fungsi trigonometri dan matematika dasar**

sqrt(x)	akar pangkat dua dari x
abs(x)	bilangan mutlak (nilai positif) dari x
sin(x)	sinus dari x → cos(x), tan(x), sinh(x), cosh(x), tanh(x)
log(x)	logaritma natur dari x
log10(x)	logaritma basis 10 dari x
exp(x)	eksponensial dari x, dll.

- **Fungsi-fungsi untuk analisis data**

min(x)	nilai minimum dari x
max(x)	nilai maksimum dari x
mean(x)	nilai rata-rata dari x
std(x)	standar deviasi dari x
sum(x)	penjumlahan dari x, dll.

- **Fungsi-fungsi untuk polinom**

poly(x)	membentuk polinom dari x, dimana x merupakan vektor dari akar persamaan
roots(x)	menentukan akar persamaan polinom dari x, dimana x merupakan vektor dari koefisien polinom dari pangkat tertinggi hingga terendah
polyfit (x,y)	membentuk polinom dari pasangan data yang terdapat pada vektor x dan y, digunakan untuk pencocokan kurva
conv(x,y)	menghitung perkalian antara polinom x dan y

Ada pula fungsi yang tidak tersedia pada Matlab *function library*, sehingga harus dibuat sendiri. Fungsi seperti ini sangat bermanfaat untuk perhitungan yang berulang-ulang (repetitif).

Contoh 1.6: Diinginkan untuk menghitung luas persegi panjang secara berulang-ulang dengan membuat programnya dalam bentuk fungsi.

Tuliskanlah program berikut pada *M-file*, dan simpanlah dengan nama **luas.m**.

```
function A = luas (p,l)
% menghitung luas persegi panjang

A = p*l;
```

Untuk menjalankan fungsi di atas, ketikkanlah pada *command window* nama fungsi yang diikuti dengan 2 buah variabel yang dibutuhkan:

```
>> luas(4,3)
ans =
   12
```

- Data variabel dapat dimasukkan dalam bentuk matriks
- Pemakaian fungsi *built-in* juga dibenarkan di dalam fungsi yang dibuat sendiri. Misalkan untuk berbagai perhitungan yang melibatkan bentuk logaritma, eksponensial, dsb.

Bagian 2 VEKTOR DAN MATRIKS

Vektor dan matriks merupakan konsep dasar perhitungan dalam Matlab. Berbagai perhitungan dapat diselesaikan dengan lebih mudah, ringkas, dan cepat bila bentuknya dikonversi ke dalam bentuk vektor/matriks. Untuk itu, harus dipahami benar dasar operasi dengan menggunakan vektor/matriks.

1. Skalar

Di dalam Matlab, skalar adalah sebuah data dengan satu baris dan satu kolom. Variabel-variabel yang memuat data skalar tersebut dapat mengalami operasi penjumlahan, pengurangan, perkalian, dan pembagian.

Contoh 2.1: Skalar dan operasinya

```
>> x = 1;
>> y = 2;
>> z = x + y
z =
       3
```

2. Vektor

Di dalam Matlab, vektor adalah sekumpulan data yang membentuk hanya satu baris atau satu kolom.

Penulisan elemen dilakukan di dalam kurung siku [ ] yang diantarai dengan spasi atau titik koma. Pengecualian berlaku hanya untuk penulisan data yang berbentuk deret dengan pola tertentu. Vektor dapat mengalami operasi dengan skalar juga dengan vektor lain asalkan mempunyai dimensi yang sama.

Contoh 2.2: Vektor dan operasinya

(i) Bentuk deret sederhana

Bentuk umum penulisan data dengan pola tertentu atau deret yang sederhana:

variabel = n : m

dimana n = nilai awal, m = nilai akhir

```
>> a = 1:3
a =
    1    2    3

>> b = 2 * a
b =
    2    4    6

>> c = [1:3]
c =
    1    2    3
```

```
>> d = 2 * c
d =
    2     4     6
```

Terdapat pesan kesalahan bila penulisan vektor yang tidak berbentuk deret ditulis tanpa kurung siku:

```
>> e = 1 3 4
??? e = 1 3 4
          |
Error: Missing operator, comma, or semicolon.
```

(ii) Penggunaan increment

Bentuk umum penulisan data dengan pola tertentu atau deret:

variabel = n : i : m

dimana n = nilai awal, m = nilai akhir, dan i = *increment*/langkah; bila i tidak didefinisikan, maka Matlab akan menggunakan *default*-nya yaitu 1, seperti yang ditunjukkan pada butir (i) di atas.

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

>> B = 0:2:10
 B =
    0    2    4    6    8   10

>> C = 10:-1:1
 C =
   10    9    8    7    6    5    4    3    2    1

>> D = 3:3:14
 D =
    3    6    9   12
```

(iii) Penggunaan kurung siku

```
>> x = [1 2 3]          % vektor baris
x =
    1    2    3

>> x = [1:3]            % mengikuti pola penulisan seperti deret
x =
    1    2    3

>> y = x'               % transposisi vektor
```

```
y =
    1
    2
    3

>> z = [4              % ada dua cara penulisan vektor kolom
       5
       6]
z =
    4
    5
    6
>> z = [4; 5; 6]
z =
    4
    5
    6

>> a = y+z            % penjumlahan 2 vektor berorde 3
a =
     5
     7
     9

>> b= x*y          % perkalian vektor baris dengan vektor kolom berorde 3
b =
    14

>> c=y*z              % perkalian 2 vektor kolom
??? Error using ==> *
Inner matrix dimensions must agree.
```

Vektor dapat mengalami operasi penjumlahan, pengurangan, perkalian, dan pembagian. Operasi penjumlahan dan pengurangan dapat dilakukan bila vektor-vektor yang akan dijumlahkan atau dikurangkan mempunyai orde (dimensi) yang sama. Perkalian 2 buah vektor x dan y mempunyai bentuk: $\sum x_i * y_i$ diman kedua vektor juga harus berde sama, tetapi 1 vektor kolom dan yang lainnya vektor baris.

Latihan 2.1:
Lihatlah pengaruh penggunaan (;) pada akhir penulisan.

3. Matriks

Matriks merupakan himpunan data yang membentuk beberapa baris dan kolom. Matriks dapat terbentuk dari gabungan 2 vektor atau lebih yang berdimensi sama. Dengan demikian, aturan operasi penjumlahan dan pengurangan yang berlaku pada vektor juga berlaku untuk matriks. Perkalian antara 2 buah matriks harus memenuhi aturan bahwa banyaknya kolom pada matriks pertama harus sam dengan benyaknya baris pada matriks kedua.

Contoh 2.3: Matriks dan Operasinya

```
>> r = [1 2 3; 2 3 4];
>> s = [3 4 5; 4 5 6];
>> t = r + s
t =
    4    6    8
    6    8   10
>> u = s - r

u =
    2    2    2
    2    2    2

>> a = 2*r
a =
    2    4    6
    4    6    8

>> b = s/4
b =
   0.7500    1.0000    1.2500
   1.0000    1.2500    1.5000

>> c = r*s
??? Error using ==> *
Inner matrix dimensions must agree.

>> c = r*s'              % jumlah baris r harus sama dengan jumlah kolom s
c =
   26   32
   38   47

>> d = a^2
??? Error using ==> ^
Matrix must be square.

>> d = a.^2
d =
    4   16   36
   16   36   64
```

Khusus untuk pemangkatan, operasi hanya dapat berlangsung secara elementer artinya masing-masing elemen dari matriks tersebut dipangkatkan.

4. Pengalamatan

Merupakan cara penulisan yang digunakan untuk menampilkan atau mendefinisikan ulang suatu data atau sekumpulan data pada vektor atau matriks, ditulis dalam bentuk umum:

variabel(i,j), dimana i menunjukkan baris dan j menunjukkan kolom

Contoh 2.4: Pengalamatan Vektor atau Matriks

x(2)	menunjukkan elemen kedua vektor x
z(3)	menunjukkan elemen ketiga vektor y
r(2,1)	menunjukkan elemen matriks r pada baris kedua kolom pertama
t(3,2)	menunjukkan elemen matriks t pada baris ketiga kolom kedua
s(:,2)	menunjukkan semua elemen matriks s pada kolom kedua
u(1,:)	menunjukkan semua elemen matriks u pada baris pertama

5. Bentuk-bentuk Khusus Vektor dan Matriks

Beberapa fungsi seperti **ones, zeros, linspace, logspace**, dsb. dapat digunakan untuk menciptakan vektor atau matriks dengan ukuran tertentu.

Contoh 2.5:

```
>> x = ones(3)              % menciptakan matriks 3x3, semua elemennya 1
x =
    1    1    1
    1    1    1
    1    1    1

>> y = ones(1,3)            % matriks dengan 1 baris 3 kolom, semua elemennya 1
y =
    1    1    1

>> z = zeros(2,3)           % matriks dengan 2 baris dan 3 kolom, semua
elemennya 0
z =
    0    0    0
    0    0    0

>> a = linspace(1,10,5)     % linearly spaced, dari 1 sampai 10 sebanyak 5 data
a =
   1.0000    3.2500    5.5000    7.7500   10.0000

>> b = logspace(1,4,4)      % logarithmically spaced
b =
        10        100       1000      10000
```

Pendefinisian ulang dapat dilakukan dengan menggunakan pengalamatan yang sesuai:

```
 >> a(3) = 6
a =
   1.0000    3.2500    6.0000    7.7500   10.0000
```

Bentuk-bentuk khusus yang lain diantaranya: **eye, rand, magic**. Ada pula manipulasi matriks untuk mengubah susunan matriks untuk rotasi (**rot**), merubah letak dari kiri ke kanan (**fliplr**), merubah letak dari atas ke bawah (**flipud**), dsb.

6. Operasi Elementer

Di atas telah disinggung sedikit tentang operasi elementer (elemen per elemen), yaitu dalam hal operasi pangkat. Operasi elementer yang lain adalah untuk perkalian dan pembagian. Sedangkan operasi penjumlahan dan pengurangan, memang berlangsung secara elementer.

Dalam penulisannya, cukup ditambahkan perintah *dot* (.) sebelum tanda operasi diberikan.

Contoh 2.6:

```
>> r                        % pemanggilan ulang matriks r
r =
    1    2    3
    2    3    4

>> s
s =
    3    4    5
    4    5    6

>> r+s                      % operasi penjumlahan
ans =
    4    6    8
    6    8   10


>> 2*r-s
ans =
   -1    0    1
    0    1    2

>> r*s'                     % operasi perkalian biasa
ans =
   26   32
   38   47

>> r.*s                     % operasi perkalian elementer
ans =
    3    8   15
    8   15   24

>> r./s
ans =
   0.3333   0.5000   0.6000
   0.5000   0.6000   0.6667

>> r.^s
ans =
         1        16       243
        16       243      4096
```

```
>> x = [1 2 3];          % dapat digunakan untuk mengevaluasi persamaan
>> y = x.^2 + 3*x + 2    % y = x^2 + 3x + 2 pada berbagai harga x
y =
    6    12    20
```

7. Fungsi Analisis Data

Terdapat banyak sekali fungsi Matlab yang dapat digunakan untuk menganalisis data, diantaranya ditampilkan pada conth berikut ini.

Contoh 2.7:

```
>> x = [1 2 3; 4 5 6; 7 8 9; 0 11 12]
x =
    1     2     3
    4     5     6
    7     8     9
    0    11    12

>> y = [0 0 3; 1 2 3; 4 5 0; -1 2 3]
y =
    0     0     3
    1     2     3
    4     5     0
   -1     2     3

>> a = max(x)            % mencari nilai maksimum berdasarkan kolom
a =
    7    11    12

>> b = max(x,y)          % mencari nilai maksimum elemen pada setiap posisi
b =
    1     2     3
    4     5     6
    7     8     9
    0    11    12

>> c = min(x)
c =
    0     2     3

>> d = min(x,y)          % mencari nilai minimum elemen pada setiap posisi
d =
    0     0     3
    1     2     3
    4     5     0
   -1     2     3
```

```
>> e = mean(x)          % mencari nilai rata-rata berdasarkan kolom
e =
   3.0000    6.5000    7.5000

>> f = sum(x)           % menjumlahkan nilai  berdasarkan kolom
f =
   12    26    30

>> g = sort(x)          % mengurutkan nilai  berdasarkan kolom
g =
    0     2     3
    1     5     6
    4     8     9
    7    11    12

>> h = std(x)           % mencari nilai standar deviasi berdasarkan kolom
h =
   3.1623    3.8730    3.8730
```

Bagian 3
PENGATURAN ALUR PROGRAM

Pengaturan alur program memungkinkan pengguna untuk mengulangi perhitungan secara berulang-ulang ataupun memilih serta memutuskan kondisi-kondisi yang sesuai/diinginkan. Matlab menyediakan empat bentuk pengaturan alur program yang akan dibahas berikut ini.

1. *Loop for*

Loop for memungkinkan sekelompok perintah diulang sebanyak suatu jumlah yang tetap. Bentuk umum:

```
for loopvariable = loopexpression
    perintah-perintah
end
```

Loopvariable merpakan nama variabel yang diberikan, sedangkan *loopexpression* biasanya memiliki bentuk n:m atau n:i:m. Perintah-perintah di antara baris *for* dan *end* dikerjakan berulang-ulang dari nilai awal n sampai nilai akhir m, dengan *increment* (langkah) sebesar i.

Contoh 3.1: Perhitungan bilangan kuadrat dari himpunan bilangan bulat dari 1 sampai 5

```
>> for n = 1:5
       x(n) = n^2
   end

x =
    1

x =
    1    4

x =
    1    4    9

x =
    1    4    9   16

x =
    1    4    9   16   25
```

Latihan 3.1:

- Perhatikanlah hasil keluaran yang terbentuk bila tanda ditambahkan tanda (;) pada perintah di antara *for-end*
- Bandingkan bila penulisan perintah perhitungan x(n) = n^2 ditulis x = n^2.

Contoh 3.2: Operasi perkalian 2 buah vektor

```
>> x = [1 2 3];
>> y = [4 5 6];
>> sum = 0;
>> for i = 1:3
sum = sum + x(i)*y(i)
end

sum =
     4
sum =
    14

sum =
    32
```

2. *Loop while*

Bentuk umum:

```
while while_expression
        perintah-perintah
end
```

While_expression merupakan bentuk hubungan e1•e2 dimana e1 dan e2 merupakan ekspresi aritmatika biasa dan • merupakan operator relasi yang didefinisikan sebagai berikut:

> lebih besar	>= lebih besar atau sama dengan
< lebih kecil	<= lebih kecil atau sama dengan
== sama	-= tidak sama

Perintah-perintah di antara baris *while* dan *end* dikerjakan berulang kali selama hubungan e1•e2 dalam ekspresi terpenuhi.

Contoh 3.3:

```
 >> n = 1;
>> x = 0;
>> while x < 20
x(n) = n^2;
     n = n+1;
     end
>> x
x =
    1    4    9   16   25
```

3. *If-Statement*

Bentuk umum:

```
if if_ekspresi
        perintah-perintah
end
```

If_ekspresi juga mengikuti bentuk hubungan e1•e2. Perintah-perintah di antara baris *if* dan *end* dikerjakan jika semua elemen di dalam ekspresi benar.

Contoh 3.4:
Sebuah toko yang menjual buah-buahan menetapkan akan memberikan potongan harga sebesar 20% bila pelanggannya membeli apel lebih dari 10.

```
clc
apel  = input('Apel yang dibeli = ');
bayar = apel * 1000;

if apel > 5
   bayar = (1-20/100)*bayar;
end
disp(['Jumlah yang harus dibayar = Rp ', num2str(bayar)])
```

Running Program:

```
Apel yang dibeli = 5
Jumlah yang harus dibayar = Rp 5000

Apel yang dibeli = 10
Jumlah yang harus dibayar = Rp 8000
```

if-else-end
Pada kasus dengan dua pilihan, konstruksi **if-else-end** adalah:

```
if if_ekspresi
        perintah dikerjakan jika benar
else
        perintah dikerjakan jika salah
end
```

Contoh 3.5:
Penentuan kelulusan seorang siswa berdasarkan dua buah ujian yang diikutinya. Ditetapkan bahwa siswa yang lulus harus memiliki nilai rata-rata minimal 60.

```
clc
nama = input('Nama Siswa    = ', 's');
N1   = input('Nilai Ujian 1 = ');
N2   = input('Nilai Ujian 2 = ');
NR = (N1+N2)/2;

if NR > 60
   ket = 'lulus';
else
   ket = 'gagal';
end
```

```
disp(' ')
disp(['Nama           = ', nama])
disp(['Nilai rata-rata = ', num2str(NR)])
disp(['Hasil akhir     = ', ket])
```

Running Program

```
Nama Siswa    = A
Nilai Ujian 1 = 60
Nilai Ujian 2 = 70

Nama          = A
Nilai rata-rata = 65
Hasil akhir     = lulus
```

Jika terdapat 3 atau lebih pilihan, konstruksi **if-else-end** mengambil bentuk:

```
if if_ekspresi1
        perintah dikerjakan jika if_ekspresi1 benar
elseif  if_ekspresi2
        perintah dikerjakan jika if_ekspresi2 benar
elseif  if_ekspresi3
        perintah dikerjakan jika if_ekspresi3 benar
elseif  if_ekspresi4
        perintah dikerjakan jika if_ekspresi4 benar
elseif ……
        .
        .
else
        perintah dikerjakan jika tidak ada if_ekspresi yang benar
end
```

4. *Switch-case-otherwise*

Bentuk umum:

```
switch ekspresi
   case ekspresi1
        perintah-perintah
   case          ekspresi2
        perintah-perintah
   case ….
        .
        .
   otherwise
        perintah-perintah
   end
```

Contoh 3.6:

```
clc
disp('1. Metoda Substitusi Berurut')
disp('2. Metoda Newton-Raphson')
disp('3. Metoda Tali Busur')

n = input('Metoda yang dipilih = ');
switch n
case (1), disp('Metoda Substitusi Berurut')
case (2), disp('Metoda Newton-Raphson')
case (3), disp('Metoda Tali Busur')
otherwise
   disp('Metoda tidak termasuk dalam daftar')
end
```

Bagian 4
BERPIKIR SECARA MATRIKS

Banyak hal bisa diselesaikan secara lebih sederhana bila penyelesaian yang dibuat didasarkan kepada bentuk matriks.

Contoh 4.1: Pendekatan array untuk Contoh 3.1.

```
>> n = 1:5;
>> x= n.^2
x =
    1    4    9   16   25
```

Meskipun kedua cara memberikan hasil yang serupa, namun cara kedua jauh lebih cepat dan memerlukan pengetikan yang lebih sedikit.

Contoh 4.2: Harga kapasitas panas campuran gas pada suatu temperatur dihitung dengan cara menjumlahkan hasil perkalian fraksi (mol/massa) komponen, y_i, dengan kapsitas panas komponen, Cp_i, yang merupakan polinom $Cp_i = A_i + B_iT + C_iT^2 + D_iT^3$.
Penyelesaiannya dapat dibandingkan antara dua cara berikut:

(i) cara *loop*

```
Cp = [1.0 0.02 0.00323 0.000003233; 3.2 0.013 0.00466 0.000004345];
y = [0.4 0.6];
T = 300;
Cpc = 0;
for i = 1:length(y)
   Cpi = 0;
   for j = 1:length(Cp)
      Cpi = Cpi + Cp(i,j)*T^(j-1);
   end
   Cpc = Cpc + y(i)*Cpi;
end
Cpc

>> Cp1

Cpc =

  480.2854
```

Catatan: Perhatikan contoh penggunaan bentuk *loop* di dalam *loop*.

(ii) cara matriks

```
Cp = [1.0 0.02 0.00323 0.000003233; 3.2 0.013 0.00466 0.000004345];
y = [0.4 0.6];
T = 300;

P = 0:3;            % mendefinisikan pangkat P = [0 1 2 3]
TT = T.*ones(1,4);  % mendefinisikan TT = [300 300 300 300]
TT = TT.^P;         % menghitung TT = [300^0 300^1 300^2 300^3]

Cpc = y*(Cp*TT')

>> Cp2

Cpc =

  480.2854
```

Contoh 4.3: Tinjau kembali Contoh 1.6. Fungsi tersebut dapat digunakan untuk menghitung beberapa persegi panjang sekaligus dimana data panjang dan lebar ditulis dalam bentuk matriks. Namun, programnya harus diubah sedikit, yaitu menambahkan *dot* pada operasi perkalian:

```
function A = luas (p,l)
% menghitung luas persegi panjang

A = p.*l;
```

Pada *command window* :

```
>> luas([4 2 7], [3 5 6])
ans =
   12    10    42
```

Bagian 5 GRAFIK

Perintah **plot** akan menghasilkan grafik dua dimensi x-y. Dibutuhkan tabel data x dan y untuk menggunakan perintah ini.
Bentuk umum: **plot(x,y)**

Contoh 4.1:

```
>> x = [ 1 2 3];
>> y = [2 4 9];
>> plot(x,y)
```

maka akan muncul sebuah grafik pada layar baru (khusus untuk grafik) yang bernama **Figure No.1**.

Untuk menambahkan keterangan pada grafik dapat menambahkan perintah-perintah berikut:

title('teks')	untuk menampilkan judul pada grafik
xlabel('teks')	untuk memberi nama pada sumbu-x grafik
ylabel ('teks')	untuk memberi nama pada sumbu-y grafik
text(2,4,'Titik 2')	untuk menampilkan teks 'Titik 2' pada lokasi x=2 dan y=4
gtext('Titik 3')	untuk menampilkan teks 'Titik 3' dengan cara meng-*click* kursor pada sembarang lokasi yang diinginkan

Dua buah grafik dapat pula di-*plot* pada layar yang sama. Matlab akan mengatur warna dari kedua grafik tersebut.

Contoh 4.2:

```
>> x = [1 2 3];
>> y = [2 4 9];
>> z=[3 7 12];
>> plot(x,y, x,z)
```

Untuk membuat grafik dalam skala logaritma atau semilogaritma, perintah plot diganti dengan **loglog** atau **semilog** dengan cara yang sama.

Bila terdapat lebih dari 1 grafik, misalkan 2 grafik, maka pada layar grafik hanya muncul grafik yang kedua, demikian seterusnya. Untuk mengatasinya, dapat ditambahkan perintah **figure(n)** dimana n menunjukkan nomor grafik. Akibatnya akan muncul sebanyak n buah layar grafik yang baru.

Dapat pula beberapa grafik ditampilkan dalam sebuah layar grafik saja dengan menggunakan perintah:

subplot(m,n,k) atau **subplot(mnk)**

dimana m menunjukkan baris, n menunjukkan kolom, dan k menunjukkan grafik yang ke berapa. Misalnya: subplot(1,3,1) artinya terdapat sebanyak 3 grafik dalam 1 baris dimana grafik yang dimaksud pada perintah ini diletakkan pada kolom 1.

Sebagai *default*, Matlab memilih *style* garis lurus serta warna biru. Pada perintah plot dapat ditambahkan tambahan argumen untuk memilih warna dan *style* untuk grafik yang akan dibuat.

Contoh: plot(x, y, 'r+') akan menghasilkan grafik dengan warna merah (*red*) dan *style* garis yang merupakan gabungan tanda +.

Diantara contoh warna, penandaan, dan *style* garis yang disediakan Matlab adalah:

Simbol	**Warna**	**Simbol**	**Penandaan**	**Simbol**	***Style* Garis**
b	Biru	.	Titik	-	Garis lurus
g	Hijau	o	Lingkaran	:	Garis titik-titik
r	Merah	x	Tanda x	-.	Garis terpotong dan titik
m	Magenta	*	Bintang	--	Garis terpotong-potong
y	Kuning	s	Bujur sangkar		
k	Hitam	d	*Diamond*		

Bagian 6
SISTEM PERSAMAAN LINIER

Penyelesaian masalah neraca massa seringkali melibatkan banyak persamaan linier sehingga membentuk suatu Sistem Persamaan Linier (SPL). SPL tersebut dapat disusun membentuk matriks, dimana umumnya merupakan matriks bujur sangkar. Untuk proses pemisahan yang berlangsung secara multitahap seperti distilasi, absorpsi, ekstraksi, dan lainnya, persamaan neraca massanya umumnya membentuk matriks tridiagonal.

Bentuk umum persamaan linier dapat dituliskan sbb. :

$$
\begin{aligned}
a_{11}\, x_1 + a_{12}\, x_2 + a_{13}\, x_3 + \ldots + a_{1n}\, x_n &= b_1 \\
a_{21}\, x_1 + a_{22}\, x_2 + a_{23}\, x_3 + \ldots + a_{2n}\, x_n &= b_2 \\
\ldots. & \\
a_{n1}\, x_1 + a_{n2}\, x_2 + a_{n3}\, x_3 + \ldots + a_{nn}\, x_n &= b_n
\end{aligned}
$$

dengan n adalah banyaknya persamaan yang menunjukkan orde matriks.

SPL di atas dapat diubah ke dalam suatu bentuk umum A x = b berdasarkan operasi perkalian matriks sbb. :

$$
\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \ldots. & & \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix}
\qquad
\begin{bmatrix} x_1 \\ x_2 \\ \ldots. \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ \ldots. \\ b_n \end{bmatrix}
$$

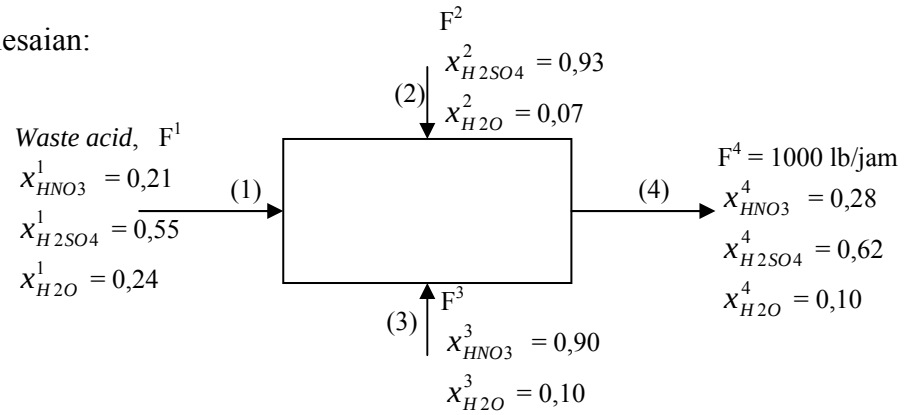
$$
A \qquad x \qquad = \qquad b
$$

Ada beberapa tahap yang harus dilalui agar dapat menyelesaikan masalah SPL secara matriks:

(i) membuat *block diagram* yang melibatkan semua alur masuk dan semua alur keluar
(ii) menurunkan persamaan neraca massa
(iii) mengubah SPL menjadi bentuk matriks A x = b.

Contoh 6.1:
Waste acid dari proses nitrasi dengan komposisi 21% HNO_3, 55% H_2SO_4, dan 24% air dipekatkan dengan menambahkan larutan H_2SO_4 93% dan larutan HNO_3 90%. Hasil pencampuran diharapkan sebanyak 1000 lb/jam dengan komposisi 28% HNO_3 dan 62% H_2SO_4. Hitunglah laju alir pada semua alur masuk.

Penyelesaian:



Persamaan neraca komponen

HNO_3 : $0{,}21\ F^1 + 0{,}90\ F^3 = 0{,}28\ F^4 = 280$ (1)

H_2SO_4 : $0{,}55\ F^1 + 0{,}93\ F^2 = 620$ (2)

H_2O : $0{,}24\ F^1 + 0{,}07\ F^2 + 0{,}10\ F^3 = 100$ (3)

Dalam bentuk matriks:

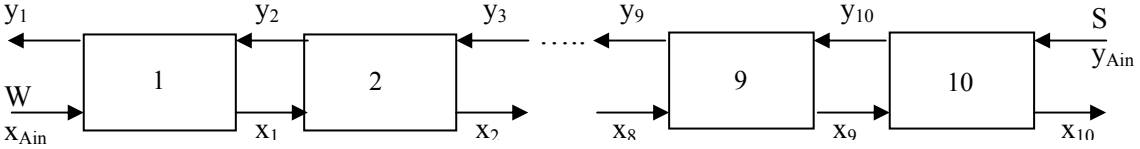
$$\begin{bmatrix} 0{,}21 & 0 & 0{,}9 \\ 0{,}55 & 0{,}93 & 0 \\ 0{,}24 & 0{,}07 & 0{,}1 \end{bmatrix} \begin{bmatrix} F^1 \\ F^2 \\ F^3 \end{bmatrix} = \begin{bmatrix} 280 \\ 620 \\ 100 \end{bmatrix}$$

Penyelesaian masalah matriks dapat dilakukan dengan menggunakan Metoda Eliminasi Gauss, baik tanpa *pivoting* maupun dengan *pivoting*. Di dalam Matlab, penyelesaiannya sedemikian sederhana:

```
>> A = [0.21  0   0.9; 0.55  0.93  0;  0.24  0.07  0.1];
>> b = [280  620  100]';   % vektor kolom
>> x=A\b

x =

  126.7894
  591.6837
  281.5269
```

Maka diperoleh laju alir umpan *waste acid* (F^1) adalah 126.7894 lb/jam, laju alir asam sulfat pekat (F^2) 591.6837 lb/jam, dan laju alir asam nitrat pekat (F^3) 281.5269 lb/jam.

Contoh 6.2: Ekstraksi Cair-Cair Multitahap



Proses ekstraksi berlawanan arah 10 tahap dilakukan untuk mengekstrak *solute* A dari campuran umpan W dengan menggunakan *solvent* S murni. Pada masing-masing tahap diasumsikan terjadi kesetimbangan dengan persamaan : $y_i = K.x_i$.

Neraca komponen A untuk tahap ke-i:

$$x_{i-1}.W + y_{i+1}.S = x_i.W + y_i.S$$

Bila diketahui data berikut:

S = 1000 kg/jam $x_{Ain} = 0{,}05$ K = 10

W = 2000 kg/jam $y_{Ain} = 0$

Maka dengan membuat peneracaan pada masing-masing tahap, diperoleh persamaan berikut yang membentuk matriks tridiagonal:

$$
\begin{aligned}
&(1)\ -6x_1 + 5x_2 = -0{,}05\\
&(2)\ x_1 - 6x_2 + 5x_3 = 0\\
&(3)\ x_2 - 6x_3 + 5x_4 = 0\\
&(4)\ x_3 - 6x_4 + 5x_5 = 0\\
&(5)\ x_4 - 6x_5 + 5x_6 = 0\\
&(6)\ x_5 - 6x_6 + 5x_7 = 0\\
&(7)\ x_6 - 6x_7 + 5x_8 = 0\\
&(8)\ x_7 - 6x_8 + 5x_9 = 0\\
&(9)\ x_8 - 6x_9 + 5x_{10} = 0\\
&(10)\ x_9 - 6x_{10} = 0
\end{aligned}
$$

Cara penyelesaian yang sama seperti Contoh 5.1 dapat digunakan.

Latihan 6.1: Aturlah cara memasukkan data dengan memanfaatkan fungsi **zeros** yang diikuti dengan pendefinisian ulang elemen dengan pengalamatan yang benar. Gunakan juga bentuk **loop-for** untuk pendefinisian ulang elemen matriks A pada posisi tridiagonal.

Bagian 7 PERSAMAAN TAK LINIER

Masalah persamaan tak linier umumnya ditujukan untuk mencari akar persamaan. Penyelesaian masalah persamaan tak linier bersifat iteratif, dilakukan berulang-ulang sehingga konvergensi tercapai.

Pada saat awal pembuatan program harus didefinisikan terlebih dahulu toleransi perhitungan yang diperkenankan serta bentuk kriteria konvergensi yang digunakan.

Salah satu dari 3 (tiga) kriteria konvergensi berikut dapat digunakan untuk mengevaluasi proses iterasi:

(i) $|x_i - x_{i-1}| \leq x_{tol}$

(ii) $\left| \frac{x_i - x_{i-1}}{x_i} \right| \leq x_{tol}$

(iii) $\left| \frac{x_i - x_{i-1}}{x_i} \right| \leq x_{tol}$

1. Persamaan Tak Linier Variabel Tunggal

Bentuk umum persamaan tak linier variabel tunggal adalah:

$$f(x) = 0$$

Ada beberapa metoda numerik yang dapat digunakan untuk menyelesaikan masalah yang melibatkan persamaan tak linier, diantaranya:

a. Metoda Substitusi Berurut

Secara ringkas metoda ini diselesaikan melalui langkah-langkah berikut:

(i) perubahan persamaan tak linier $f(x) = 0$ menjadi $x = g(x)$, dimana $g(x)$ adalah persamaan tak linier yang mengandung variabel x yang berbeda dari $f(x)$

(ii) menetapkan x_{tol}

(iii) menetapkan sebuah tebakan awal x_0

(iv) melakukan proses iterasi 1 untuk menentukan x_1:
$x_1 = f(x_0)$

(v) melakukan evaluasi dengan memilih salah satu kriteria konvergensi (kk)

(vi) apabila hasil evaluasi menunjukkan nilai kk $\leq x_{tol}$, maka perhitungan dapat dihentikan dan akar yang dicari adalah x_1;
tetapi bila tidak, maka perhitungan harus diulangi untuk iterasi selanjutnya.

Contoh 7.1: Program penentuan akar persamaan dengan menggunakan metoda substitusi berurut untuk persamaan $f(x) = x^4 - e^x + 1 = 0$ adalah sebagai berikut:

```
% Program Substitusi Berurut
```

```
% Menghitung akar persamaan : f(x) = x^4 - e^x + 1 = 0
% --->  x = (e^x - 1)^0.25

clc
xtol = 5e-5;
itr = 1;
x0 = 1;

x1 = (exp(x0) - 1)^0.25;

   while abs(2*(x1-x0)/(x1+x0)) > xtol
   itr = itr+1;
      x0 = x1;
      x1 = (exp(x0) - 1)^0.25;
   end

disp(['Akar persamaan adalah = ', num2str(x1)])
disp(['Banyaknya iterasi yang dilakukan = ', num2str(itr)])
fprintf('\n')
```

Catatan:

- program ini tidak bersifat mutlak, artinya masih dapat diubah sepanjang masih sesuai dengan dasar perhitungan untuk metoda substitusi berurut
- program ini dapat digunakan untuk metoda penyelesaian persamaan tak linier yang lain dengan melakukan perubahan sesuai dengan dasar perhitungan metoda ybs.

Latihan 7.1:

- ubahlah program di atas, agar perhitungan pada iterasi pertama tidak berada di luar *loop-while*
- ubahlah program di atas, agar hasil pada setiap iterasi dapat tampil pada layar monitor.

b. Metoda Newton-Raphson

Bentuk umum dari persamaan Newton-Raphson adalah sebagai berikut:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Jadi, metoda Newton-Raphson membutuhkan turunan fungsi dalam penyelesaian masalahnya serta sebuah tebakan awal.

c. Metoda Tali Busur (*Secant*)

Metoda Tali Busur merupakan pengembangan dari metoda Newton-Raphson, dimana persamaan turunan fungsi diganti dengan pendekatan beda maju sehingga metoda ini merupakan alternatif bagi turunan yang sukar.

Bentuk umum dari persamaan tali busur adalah sebagai berikut:

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Dibutuhkan dua buah tebakan awal untuk metoda ini, yaitu x_0 dan x_1.

Secara umum penyelesaian masalah persamaan tak linier dengan menggunakan Metoda Newton-Raphson dan Tali Busur sama dengan Metoda Substitusi Berurut.

Latihan 7.2: Ubahlah program pada Contoh 7.1, agar dapat dipergunakan untuk menentukan akar persamaan dengan menggunakan metoda Newton-Raphson dan Tali Busur.

d. Fungsi *Built-in* Matlab

Matlab mempunyai fungsi khusus untuk menyelesaikan masalah pencarian akar persamaan tak linier ini atau pencarian nol dengan perintah **fzero**. Caranya adalah dengan menuliskan *function* pada sebuah *M-file* yang berisikan persamaan tersebut.

Contoh 7.2: Program penentuan akar persamaan $f(x) = x^4 - e^x + 1 = 0$ dengan menggunakan fungsi *built-in* Matlab

```
function y = akar(x)

y = x^4 - e^x + 1;
```

Simpanlah *file* tersebut dengan nama akar.m, selanjutnya ketiklah pada *command window*:

```
>> x=fzero('akar', 0)

x =

   -1.3916
```

Setelah perintah *fzero*, buatlah di dalam kurung nama *file* dalam bentuk string yang diikuti dengan tebakan awal yang diberikan, dimana di antaranya dipisahkan dengan tanda koma. Bila akar persamaan lebih dari satu, maka hasil yang ditampilkan hanyalah akar yang paling mendekati dengan tebakan.

2. Polinomial

Polinomial mempunyai bentuk umum sebagai berikut:

$$f(x) = a_0.x^N + a_1.x^{N-1} + a_2.x^{N-2} + \ldots. + a_{N-2}.x^2 + a_{N-1}.x^1 + a_N.x^0$$

a. Menentukan akar persamaan

Untuk menentukan akar persamaan dari sebuah polinom, dapat digunakan fungsi **roots**.

Contoh 7.3: Perhatikan persamaan berikut:

$$f(x) = x^2 + 3x + 2$$
$$= (x+2)(x+1)$$

sehingga akar persamaannya adalah: $x_1 = -2$ dan $x_2 = -1$

Dalam Matlab dapat diselesaikan:

```
>> a=[1 3 2];    % koefisien polinom dimulai dari x^N sampai x^0
>> roots(a)
ans =
            -2
     -1
```

b. Membentuk polinom

Sebaliknya, Matlab juga mempunyai fungsi untuk membentuk polinom dari akar-akarnya.

Contoh 7.4: Untuk akar persamaan yang diperoleh pada contoh di atas, dapat ditentukan persamaannya:

```
>> b = [-2 -1];
>> poly(b)
ans =
    1     3     2
```

c. Operasi polinom

Polinom dapat mengalami berbagai operasi aritmatika.

Contoh 7.5: Misalkan diketahui dua buah persamaan :

$f(x) = 3x^3 + 2x^2 + 1$

$g(x) = 4x^2 + 2x + 3$

Operasi penjumlahan terhadap dua polinom adalah dengan cara menjumlahkan masing-masing koefisiennya, demikian pula dengan pengurangan.

```
>> f=[3 2 0 1];
>> g=[4 2 3];
>> f+g
??? Error using ==> +
Matrix dimensions must agree.
```

Untuk menghindari kesalahan, maka matriks yang terlibat dalam operasi penjumlahan/ pengurangan harus mempunyai ukuran yang sama.

```
>> g=[0 4 2 3];
>> f+g
ans =
    3     6     2     4

>> f-g
ans =
    3    -2    -2    -2
```

artinya, polinom hasil penjumlahan adalah: $3x^3 + 6x^2 + 2x + 4$

dan polinom hasil pengurangan adalah: $3x^3 - 2x^2 - 2x - 2$.

Operasi perkalian dan pembagian melibatkan perhitungan yang lebih rumit. Fungsi yang disediakan Matlab berturut-turut adalah **conv** (*convulation*) dan **deconv**. Tidak seperti penjumlahan atau pengurangan, penulisan vektor koefisien polinom tidaklah harus mempunyai ukuran yang sama.

Contoh 7.6: Untuk dua polinom yang sama seperti di atas, maka operasi perkalian dan pembagian dapat ditulis sebagai berikut:

```
>> f=[3 2 0 1];
>> g=[4 2 3];
>> conv(f,g)
ans =
   12    14    13    10     2     3

>> [k,s]=deconv(f,g)
k =
   0.7500    0.1250

s =
         0         0   -2.5000    0.6250
```

Sehingga jawabannya adalah :

0,75x + 0.125 dengan sisanya
-2.5x + 0.625

Jika k dikalikan dengan g(x), kemudian hasil perkalian tersebut dijumlahkan dengan s, maka pastilah akan sama dengan f(x)

```
>> kali=conv(k,g)
kali =
      3.0000    2.0000    2.5000    0.3750

>> kali + s
ans =
    3     2     0     1
```

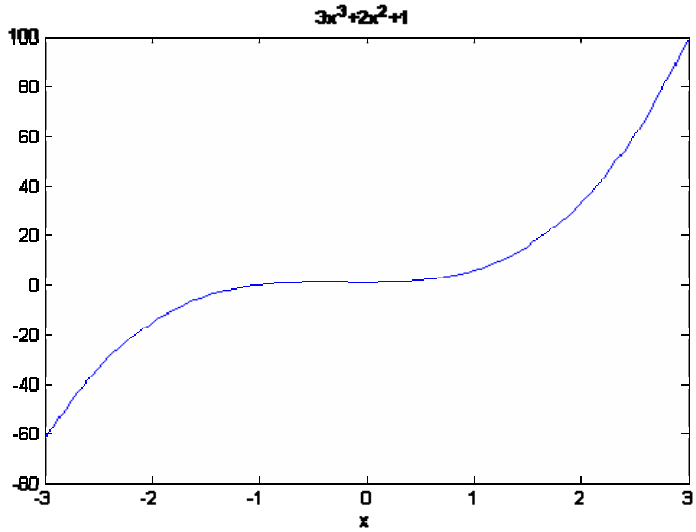
d. Evaluasi Polinom

Fungsi **polyval** digunakan untuk mengevaluasi polinom.

Contoh 7.7:

```
>> f=[3 2 0 1];
>> nilai = polyval(f,3)
nilai =

   100
>> nilai = polyval(f,[3 2])

nilai =
```

```
   100    33
>> x=linspace(-3,3);
>> nilai = polyval(f,x);
>> plot(x,nilai), title('3x^3+2x^2+1'), xlabel('x')
```



e. Turunan

Fungsi **polyder** merupakan fungsi yang disediakan Matlab untuk mencari turunan (*derivat*) dari suatu polinom.

Contoh 7.8:

```
>> f=[3 2 0 1];
>> der=polyder(f)

der =

     9     4     0
```

Bagian 8
PENCOCOKAN KURVA

1. Interpolasi

Pendekatan yang dilakukan pada interpolasi adalah mencocokkan sebuah atau sederetan kurva secara langsung melalui masing-masing titik data.

a. Interpolasi 1 Variabel

Interpolasi Linier

Merupakan bentuk yang paling sederhana untuk menaksir nilai di antara nilai-nilai yang diketahui dengan baik. Interpolasi linier menghubungkan 2 titik data [(x_1, $f(x_1)$) dan x_2, $f(x_2)$)] dengan garis lurus, lalu dengan penghampiran menentukan nilai fungsi f(x) dari suatu titik (x) yang terletak diantaranya.

Persamaan umum interpolasi linier:

$$f(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Contoh 8.1:

Diketahui data:

x	ln(x)
1	0
4	1,3863
6	1,7917

Taksirlah harga logaritma natur dari 2 atau ln(2).

Penyelesaian dalam bahasa Matlab:

```
% data x
x = [1 4 6];

% data ln x
lnX = [0 1.3863 1.7917];

cari = 2;
ln2 = lnX(1) + (lnX(2) - lnX(1)) / (x(2)-x(1)) * (cari – x(1))
```

Running Program:

```
ln2 =

    0.4621
```

Interpolasi Kuadrat

Merupakan polinom berderajat dua dengan bentuk umum:

$$f(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$$

Untuk menentukan konstanta b_0, b_1, dan b_2 dibutuhkan 3 titik data yaitu [(x_0, $f(x_0)$), (x_1, $f(x_1)$), dan x_2, $f(x_2)$)].

Substitusi ke dalam bentuk umum:

$x = x_0 \rightarrow f(x_0) = b_0$

$x = x_1 \rightarrow f(x_1) = b_0 + b_1(x_1 - x_0)$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$x = x_2 \rightarrow f(x_2) = b_0 + b_1(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1)$

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

Latihan 8.1: Dengan contoh yang sama, cobalah buat program untuk interpolasi kuadrat.

Built-in function

Penyelesaian dengan menggunakan *built-in function* adalah dengan perintah **interp1**.

Bentuk umum: **YI = interp1(X,Y,XI)**

X dan Y merupakan vektor yang berisikan data-data diskrit dimana Y = f(X). Interp1 melakukan interpolasi untuk mendapatkan nilai YI pada titik data XI.
Interpolasi dapat dilakukan dengan beberapa metoda diantaranya linier, kubik, dan spline. Bila metoda yang akan digunakan tidak dispesifikasi, maka Matlab akan menggunakan metoda linier sebagai *default*.
Perintah yang digunakan menjadi: **YI = interp1(X,Y,XI,'method')**

Contoh 8.2:

```
clc

% data x
x = [1 4 6];

% data ln x
lnX = [0 1.3863 1.7917];

hasil  = interp1(x, lnX, 2)
hasil1 = interp1(x, lnX, 2, 'linear')
hasil2 = interp1(x, lnX, 2, 'cubic')
hasil3 = interp1(x, lnX, 2, 'spline')
hasil4 = interp1(x, lnX, 2, 'nearest')
hasil5  = interp1(x, lnX, [2, 5])
```

Running Program

```
hasil =
   0.4621

hasil1 =
   0.4621

hasil2 =
   0.5729

hasil3 =
   0.5659

hasil4 =
    0

hasil5 =
   0.4621    1.5890
```

b. Interpolasi 2 Variabel

Interpolasi 2 variabel juga bertujuan untuk menaksir nilai di antara nilai-nilai yang diketahui dengan baik, tetapi dilakukan pada data yang mempunyai 2 variabel.

Contoh 8.3: Diketahui sebuah kumpulan data sebagai berikut

x_2 \ x_1	1	2	3
1	10	20	30
2	40	46	50
3	50	60	100

Tentukanlah nilai data pada $x_1 = 1{,}5$ dan $x_2 = 2{,}3$.
Penyelesaian berikut dilakukan dengan menggunakan *built-in function* **interp2**.

Bentuk umum: **ZI = interp2(X,Y,Z,XI,YI)**

X, Y, dan Z merupakan vektor yang berisikan data-data diskrit dimana Z = f(X,Y).
Interp2 melakukan interpolasi untuk mendapatkan nilai ZI pada titik data XI dan YI.
Seperti interp1, interp2 dapat dilakukan dengan beberapa metoda dengan metoda linier sebagai *default*.

```
clc

x1 = [1 2 3];
x2 = [1 2 3];
```

```
data = [10 20 30
        40 46 50
        50 60 100];

hasil = interp2(x1, x2, data, 1.5, 2.3)
```

Running Program

```
hasil =

   46.6000
```

Latihan 7.2: Data berikut diambil dari *steam table* untuk *superheated steam*

P(bar)	kJ/kg	50°C	75°C	100°C	150°C
250	H	230,7	334	437,8	647,7
	U	205,7	308,7	412,1	620,8
300	H	235	338,1	441,6	650,9
	U	205	307,7	410,8	618,7
500	H	251,9	354,2	456,8	664,1
	U	202,4	304	405,8	611

- Buatlah program untuk menentukan entalpi pada 225 bar dan 75°C
- Buatlah program untuk menentukan energi dalam pada 345 bar dan 125°C.
- Buatlah program tersebut masing-masing dengan mengikuti urutan pekerjaan secara manual (dengan tangan) serta dengan menggunakan *built-in function*.

2. Regresi

Pendekatan yang dilakukan pada regresi adalah menurunkan suatu kurva tunggal (dengan persamaan tertentu) yang membentuk suatu kecenderungan umum dari data dimana kurva tersebut tidak mengikuti pola titik-titik data tersebut tetapi diambil sebagai suatu kelompok yang mewakili.
Kurva yang terbentuk dapat mengikuti persamaan linier (garis lurus) atau polinom berderajat tertentu, juga persamaan eksponensial atau logaritma.

Bentuk umum persamaan linier: $y = a_0 + a_1.x$

dimana: a_1 = *slope* (kemiringan) dan a_0 = intersep (perpotongan garis dengan sumbu y)

Kedua konstanta dihitung dengan persamaan berikut:

$$a_1 = \frac{n\sum x_i.y_i - \sum x_i.\sum y_i}{n\sum x_i^{\,2} - (\sum x_i)^2} \quad \text{dan} \quad a_0 = \frac{\sum y_i}{n} - a_1\frac{\sum x_i}{n}$$

Contoh 8.4: Buatlah model persamaan dari data-data berikut

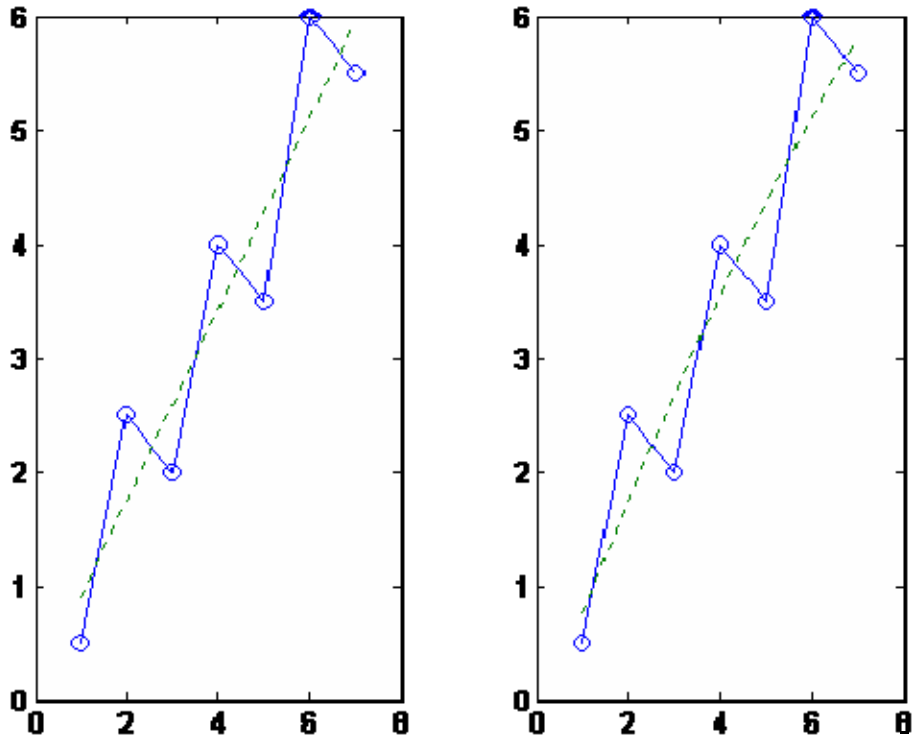
X	1	2	3	4	5	6	7
Y	0,5	2,5	2	4	3,5	6	5,5

Penyelesaian dilakukan dengan memanfaatkan fungsi **polyfit** dengan bentuk umum:
P = polyfit (X,Y,N) untuk mencari koefisien polinom P(X) berderajat N yang cocok terhadap pasangan data X(I) dan Y)I) dengan mencari *least square*. Bila N=1, akan dihasilkan pendekatan garis lurus; bila N=2 akan dihasilkan pendekatan kuadratis.

```
clc
x = [1:7];
y = [0.5 2.5 2 4 3.5 6 5.5];

p = polyfit(x,y,1)
r = polyfit(x,y,2)

yi = linspace(1, 7);
s = polyval(p, yi);
subplot(1,2,1);plot(x, y, '-o', yi, s, ':')

xi = linspace(1,7); % LINSPACE Linearly spaced vector
z = polyval(r, xi);
subplot(1,2,2);plot(x, y, '-o',xi, z, ':')
```

Running Program

```
p =
   0.8393    0.0714

r =
  -0.0298    1.0774   -0.2857
```


0
1
2
3
4
5
6
0
2
4
6
8
0
1
2
3
4
5
6
0
2
4
6
8

Bagian 9 PERSAMAAN DIFERENSIAL

1. Persamaan Diferensial Biasa Orde 1 Tunggal

Bentuk umum :

$$\frac{dy}{dx} = f(x,y) \quad \text{dimana } y(x_0) = y_0$$

a. Metoda Euler Eksplisit

Bentuk umum :

$$y_{i+1} = y_i + \Delta x \, . \, f(x_i, y_i)$$

Contoh 9.1:

Dengan menggunakan metoda euler eksplisit, tentukanlah nilai y pada x =1 jika dy/dx = x^2y, dimana y = 1 pada x = 0.

Penyelesaian:

Dari bentuk umum, maka: $y_{i+1} = y_i + \Delta x \, x_i^2 \, y_i$

Bila dipilih $\Delta x = 0{,}1$ maka:

```
% PDB - Eksplisit
clc
x0 = 0;     % Nilai awal
y0 = 1;
xa = 1;     % x akhir
dx = 0.1;

for i = 1:10
   y = y0 + dx * x0^2 * y0
   x0 = x0 + dx;
   y0 = y;
end
```

***Running Program*:**

```
y =
     1
y =
   1.0010
y =
   1.0050
```

```
y =
   1.0140
y =
   1.0303
y =
   1.0560
y =
   1.0940
y =
   1.1477
y =
   1.2211
y =
   1.3200
```

Contoh 9.2:
Reaksi berikut dilangsungkan pada suatu reaktor *semi-batch* $A_{(l)} \rightarrow P_{(l)}$ dimana $r = kC_A^2$. Pada saat awal reaktor diisi dengan cairan inert dengan volume V_0. Pada saat t = 0 senyawa A dengan konsentrasi C_{A0} diumpankan ke reaktor dengan laju Q_0. Dari neraca mol komponen A pada keadaan *unsteady* diperoleh:

$dn_A/dt = Q_0 . C_{A0} - k.n_A^2/V_R$

dimana $C_A = n_A/V_R$

Karena cairan ditambahkan ke reaktor, maka V_R akan bertambah terhadap waktu. Neraca massa di reaktor:

b. Metoda Runge-Kutta

Bentuk umum:

$$\mathbf{y_{i+1} = y_i + \Delta x/6 (k_1 + 2k_2 + 2k_3 + k_4)}$$

dimana:
$k_1 = f(x_i, y_i)$
$k_2 = f(x_i + ½ \Delta x, y_i + ½ k_1 \Delta x)$
$k_3 = f(x_i + ½ \Delta x, y_i + ½ k_2 \Delta x)$
$k_4 = f(x_i + \Delta x, y_i + k_3 \Delta x)$

c. Metoda Euler Implisit

Bentuk umum:

$$\mathbf{y_{i+1} = y_i + \Delta x\ f(x_{i+1}, y_{i+1})}$$

d. Fungsi *Built-in* Matlab

Contoh 9.3: Program untuk persamaan differensial $dy/dx = x^2 y$
Program: disimpan dalam *file* diferensial.m

```
function fx = diferensial(x,y)
fx = x^2*y;
```

***Running Program*:**

Pada *command window*:

```
[x,y] = ode45('diferensial', [0:0.1:1], 1)
```

x =	y =
0	1.0000
0.1000	1.0003
0.2000	1.0027
0.3000	1.0090
0.4000	1.0216
0.5000	1.0425
0.6000	1.0747
0.7000	1.1211
0.8000	1.1861
0.9000	1.2751
1.0000	1.3956

Hasil yang diperoleh dengan fungsi *built-in* ode45 ini sama dengan hasil perhitungan secara analitik.

2. Sistem Persamaan Diferensial Biasa Orde 1

Sistem persamaan diferensial biasa orde 1 melibatkan lebih dari satu PDB dengan bentuk umum sebagai berikut:

$$\frac{dy_1}{dx} = f_1(x,y)$$
$$\frac{dy_2}{dx} = f_2(x,y)$$
$$\vdots$$
$$\frac{dy_n}{dx} = f_n(x,y)$$

dimana $y = (y_1, y_2, \ldots, y_n)$ dan untuk x yang tertentu, maka y_i diketahui.

Metoda yang digunakan juga sama seperti penyelesaian PDB orde 1 yang tunggal.

a. Metoda Euler Eksplisit

Karena masing-masing persamaan dy_i/dx bergantung secara umum terhadap semua nilai y_i, maka masing-masing $f_i(x,y)$ harus dihitung terlebih dahulu. Maka algoritma untuk metoda ini adalah:

$$y_{1,j+1} = y_{1,j} + \Delta x \,.\, f_1(x_j, y_j)$$
$$y_{2,j+1} = y_{2,j} + \Delta x \,.\, f_2(x_j, y_j)$$
$$\vdots$$
$$y_{n,j+1} = y_{n,j} + \Delta x \,.\, f_n(x_j, y_j)$$

dimana $y_j = (y_{1,j}, y_{2,j}, \ldots, y_{n,j})$. Sebagai contoh, $y_{i,j}$ merupakan nilai y_i pada nilai x yang ke-j (yaitu, jika kondisi awal ditentukan pada x = 0, maka nilai x yang ke-j adalah j. Δx.

b. Metoda Runge-Kutta

Bentuk umum:

$$\mathbf{y_{i,j+1} = y_{i,j} + \Delta x/6\,(k_{1,i,j} + 2k_{2,i,j} + 2k_{3,i,j} + k_{4,i,j})}$$

dimana: $k_{1,i,j} = f(x, y_{1,j}, y_{2,j}, \ldots, y_{n,j},)$

$k_{2,i,j} = f(x + \frac{1}{2}\Delta x, y_{1,j} + \frac{1}{2} k_{1,i,j}\,\Delta x, \ldots., y_{n,j} + \frac{1}{2} k_{1,n,j}\,\Delta x)$

$k_{3,i,j} = f(x + \frac{1}{2}\Delta x, y_{1,j} + \frac{1}{2} k_{2,i,j}\,\Delta x \ldots., y_{n,j} + \frac{1}{2} k_{2,n,j}\,\Delta x)$

$k_{4,i,j} = f(x + \Delta x, y_{1,j} + k_{3,i,j}\,\Delta x \ldots., y_{n,j} + k_{3,n,j}\,\Delta x)$

Contoh 9.4: Reaksi seri A → B → C dijalankan dalam sebuah reactor *batch*. Kecepatan reaksi A dan B adalah:

$dC_A/dt = -k_1 C_A$

$dC_B/dt = k_1 C_A - k_2 C_B$

$dC_C/dt = -k_2 C_B$

dimana $k_1 = 0{,}1$ dan $k_2 = 0{,}5$.

Jika mula-mula (t=0) $C_A = 0{,}5$; $C_B = 0$, $C_C = 0$, maka hitunglah konsentrasi semua komponen pada saat t = 6. Gunakanlah $\Delta t = 2$.

3. Persamaan Diferensial Parsial

Bentuk umum:

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{t}} = \boldsymbol{\alpha} \frac{\partial^2 \mathbf{Y}}{\partial \mathbf{x}^2}$$

dimana $Y(0,x) = Y_0$, $Y(t,0) = Y_0$, $Y(t,L) = Y_L$.

Penyelesaian yang paling sederhana adalah dengan menggunakan Metoda Eksplisit dengan pendekatan beda maju:

$$\left.\frac{\partial Y}{\partial x}\right|_{Yi,j} = \frac{Y_{i,j+1} - Y_{i,j}}{\Delta x}$$

$$\left.\frac{\partial^2 Y}{\partial x^2}\right|_{Yi,j} = \frac{Y_{i+1,j} - 2Y_{i,j} + Y_{i-1,j}}{\Delta x^2}$$

Kemudian dengan mensubstitusi dan menyelesaikan $Y_{i,j+1}$ menghasilkan:

$$Y_{i,j+1} = Y_{i,j} + \frac{\alpha.\Delta t}{\Delta x^2}(\,Y_{i+1,j} - 2.\,Y_{i,j} + Y_{i-1,j})$$

Contoh 9.5: Sebuah benda dengan panjang dan lebar tak terhingga memiliki ketebalan 5 cm. Mula-mula benda bersuhu 30 ^{0}C. Tepat mulai saat t = 0, kedua suhu sisi benda dirubah dan dipertahankan tetap. Pada x = 0, suhu benda dibuat 70 ^{0}C dan pada x = 5 dibuat bersuhu 40 ^{0}C. Distribusi suhu sebagai fungsi posisi dan waktu mengikuti persamaan : $\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T}{\partial t}$. Jika α = 2 cm^2/menit, tentukanlah suhu pada titik berjarak 4 cm pada saat 2 menit.

Bagian 10
INTEGRAL

1. Metoda Trapesium

Integrasi fungsi dari x=a hingga x=b dan penyusunan ulang menghasilkan:

$$\int_a^b f(x)dx = \left[\frac{f(a)+f(b)}{2}\right](b-a)$$

Bila batas a~b dibagi menjadi tiga daerah maka :

$$\int_{x0}^{x3} f(x)dx = \int_{x0}^{x1} f(x)dx + \int_{x1}^{x2} f(x)dx + \int_{x2}^{x3} f(x)dx$$

terhadap masing-masing daerah dapat diterapkan metoda trapesium sehingga:

$$\int_{x0}^{x3} f(x)dx = \left[\frac{f(x0)+f(x1)}{2}\right](x1-x0) + \left[\frac{f(x1)+f(x2)}{2}\right](x2-x1) + \left[\frac{f(x2)+f(x3)}{2}\right](x3-x2)$$

atau dalam bentuk umum :

$$\int_{x0}^{xn} f(x)dx = \sum_{i=1}^{n} \frac{f(x_i)+f(x_{i-1})}{2}(x_i - x_{i-1})$$

Bila pembagian daerah dilakukan dengan jarak (Δx) yang sama, maka bentuk umumnya menjadi:

$$\int_{x0}^{xn} f(x)dx = \frac{\Delta x}{2}\left[f(x0) + 2\sum_{i=i}^{n-1} f(xi) + f(xn)\right]$$

Contoh 10.1: Dengan menggunakan data pada tabel berikut, tentukanlah data fugasitas N_2 pada 25^{o}C dan 800 atm. Untuk komponen murni, fugasitas f dihitung dengan persamaan:

$$\ln\left(\frac{f}{P}\right) = \int_0^P \frac{z-1}{P} dP$$

dimana z adalah faktor kompressibilitas dengan data sebagai berikut:

P(atm)	0^{o}C	25^{o}C	50^{o}C
0	1,000	1,000	1,000
10	0,996	0,998	1,000
50	0,985	0,996	1,004
100	0,984	1,004	1,018
200	1,036	1,057	1,072
300	1,134	1,146	1,154
400	1,256	1,254	1,253
600	1,524	1,495	1,471
800	1,798	1,723	1,697

Karena z merupakan fungsi P, maka tidak dapat dikeluarkan dari integral.
Bila semua data digunakan, maka dari bentuk umum diperoleh persamaan:

$$\ln\left(\frac{f}{800}\right) = \int_0^{800} \frac{z-1}{P} dP$$

$$= \sum_{i=1}^{8} \left(\frac{z_i - 1}{P_i} + \frac{z_{i-1} - 1}{P_{i-1}} \right) \frac{P_i - P_{i-1}}{2}$$

Khusus untuk kasus di atas, dilakukan manipulasi data P awal untuk menghindari terjadinya operasi 0/0.

```
clc
z = [1 0.998 0.996 1.004 1.057 1.146 1.254 1.495 1.723];
P = [1 10    50      100   200   300   400   600   800];
integral=0;

for i = 2:9
   jumlah = ((z(i)-1)/P(i) + (z(i-1)-1)/P(i-1)) * (P(i)-P(i-1))/2;
   integral = integral + jumlah;
end
integral
fugasitas = P(9) * exp(integral)
```

Running Program:

```
integral =

   0.4223


fugasitas =

  1.2204e+003
```

Penyelesaian dengan *built-in function* dapat dilakukan dengan perintah **trapz**.

```
clc
z  = [1 1.057 1.254 1.495 1.723];
P = [1 200  400   600  800];
y = (z-1)./P;
tek = 0:200:800;
area = trapz(tek,y)
fugasitas = P(5)*exp(area)
```

Running Program:

```
area =
   0.4394

fugasitas =
  1.2414e+003
```

Contoh 10.2: $\int_0^{0.4} x.\exp(-x)dx$

```
clc
x= 0:0.1:0.4;
y=x.*exp(-x);
area = trapz(x,y)

x1= 0:0.05:0.4;
y1=x1.*exp(-x1);
area1 = trapz(x1,y1)
```

Running Program:

```
area =
   0.0611

area1 =
   0.0614
```

2. Metoda Simpson

Luas daerah di bawah f(x) dari x = xa hingga x = xc dengan menggunakan metoda simpson:

$$\int_{xa}^{xc} f(x)dx = (xc - xa)\left[\frac{f(xa) + 4f(b) + f(xc)}{6}\right]$$

atau

$$\int_{xa}^{xc} f(x)dx = \frac{\Delta x}{3}\left[f(xa) + 4f(b) + f(xc)\right]$$

dimana Δx = jarak antara titik yang dievaluasi yaitu (xc-xa)/2.

Bila batas xa~xc dibagi menjadi tiga daerah dimana masing-masing daerah terdiri dari tiga titik, maka :

$$\int_{x0}^{x6} f(x)dx = \int_{x0}^{x2} f(x)dx + \int_{x2}^{x4} f(x)dx + \int_{x4}^{x6} f(x)dx$$

$$= \frac{\Delta x}{3}\left[f(x0) + 4f(1) + f(x2)\right] + \frac{\Delta x}{3}\left[f(x2) + 4f(3) + f(x4)\right] + \frac{\Delta x}{3}\left[f(x4) + 4f(5) + f(x6)\right]$$

$$= \frac{\Delta x}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3 0) + 2f(x_4) + 4f(x_5) + f(x_6)\right]$$

atau dalam bentuk umum:

$$\int_{x0}^{xn} f(x)dx = \frac{\Delta x}{3}\left[f(x_0) + 4\sum_{i=1,3,\ldots}^{n-1} f(x_i) + 2\sum_{i=2,4,\ldots}^{n-2} f(x_i) + f(x_n)\right]$$

Contoh 10.3:

```
F = inline('x.*exp(-x)');
area2 = quad(F,0,0.4)
area3 = quadl(F,0,0.4)
```

Running Program:

```
area2 =
   0.0616

area3 =
   0.0616
```

3. Integral Berganda (*Double Integral*)

Fungsi yang digunakan untuk menghitung *double integral* adalah **dblquad**, dengan bentuk umum:

variabel = dblquad('function', inmin, inmax, outmin, outmax)

dblquad digunakan untuk mengevaluasi *double integral* dari **function(inner,outer)** dengan *inner* adalah variabel integral dalam yang nilainya bervariasi dari *inmin* hingga *inmax*; sedangkan *outer* adalah variabel integral luar yang nilainya bervariasi dari *outmin* hingga *outmax*.

Contoh 10.4: $\int_0^{\pi}\int_0^{\pi}(4\sin(x)-3x\cos(y))dxdy$

Terlebih dahulu harus dituliskan sebuah fungsi untuk persamaan di atas yang disimpan dengan nama dint.m:

```
function w = dint(x,y)
          w  =  4*sin(x) – 3*x*cos(y);
```

Kemudian ketikkan pada *command window*:

```
>> hasil = dblquad('dint', 0, pi, 0, pi)
hasil =
         25.1330
```

Atau bila hendak menggunakan fungsi **quadl** dapat ditulis sebagai berikut:

```
>> hasil = dblquad('dint', 0, pi, 0, pi,'quadl')
hasil =
         25.1361
```

DAFTAR PUSTAKA

Azree Idris, *Matlab for Engineering Students*, Prentice Hall – Pearson education Malaysia Sdn. Bhd., Malaysia, 2000

Etter, Delores M., David C. Kuncicky, dan Doug Hull, Penerjemah Carley Tanya, *Pengantar Matlab 6*, PT. Indeks Kelompok Gramedia, Jakarta, 2003

Hanselman, Duane dan Bruce Littlefield, *Matlab: Bahasa Komputasi Teknis*, Penerbit Andi, Yogyakarta, 2000

Lindfield, G. dan John Penny, *Numerical Methods Using Matlab*, Ellis Horwood, New York, 1995

Raman, R, *Chemical Process Computation*, Elsevier, New York, 1985

Riggs, J.B, *An Introduction to Numerical Methods for Chemical Engineers*, TexasTech. University Press, 1988

MODUL PRAKTIKUM

PROGRAM KOMPUTER

Program Diploma IV
Teknologi Kimia Industri

Departemen Teknik Kimia
Fakultas Teknik
U S U
Medan
2006