Pemodelan dan Simulasi Penghitungan Matematika Menggunakan Aplikasi Matlab

Dwi Retnoningsih
Program Studi Teknik Informatika, Universitas Sahid Surakarta
Jl. Adi Sucipto 154, Jajar, Surakarta, 57144, Telp. (0271) 743493, 743494
Email: dw1retno@yahoo.co.id

Abstract

Calculate simple math might be resolved manually. But complex calculations such as Array, Matrix, and polynomial may require considerable time to complete. So in this study the authors make the modeling and simulation to assist the process for faster computation.

The purpose of this research is to make the modeling and simulation of a simple mathematical operation, array of one-dimensional, two-dimensional arrays, matrices, and polynomial.

This research resulted in the modeling and simulation of a simple mathematical operation, array of one-dimensional, two-dimensional arrays, matrices, and polynomial using Matlab 6.0 application tools

Keywords: *modeling and simulation, Array, Matrix, polynomiala, Matlab 6.0*

Pendahuluan

Latar Belakang

Menghitung matematika yang sederhana mungkin dapat diselesaikan secara manual. Tetapi penghitungan yang komplek seperti Array, Matrik, dan polinomial mungkin membutuhkan waktu yang cukup lama untuk menyelesaikannya. Sehingga pada penelitian ini penulis membuat pemodelan dan simulasi untuk membantu proses penghitunganya agar lebih cepat.

Tujuan dari penelitian ini adalah membuat pemodelan dan simulasi beberapa operasional matematika menggunakan alat bantu aplikasi Matlab 6.0.

Permasalahan

Menghitung matematika yang sederhana mungkin dapat diselesaikan secara manual. Tetapi penghitungan yang komplek seperti Array, Matrik, dan polinomial mungkin membutuhkan waktu yang cukup lama untuk menyelesaikannya. Sehingga pada penelitian ini penulis membuat pemodelan dan simulasi untuk membantu proses penghitunganya agar lebih cepat.

Batasan Masalah

Pada penelitian ini penulis membatasi pembahasan tentang matematika sederhana, array satu dimensi, array dua dimensi, matrik, dan Polinomial.

Tujuan Penulisan

Tujuan dari penelitian ini adalah membuat pemodelan dan simulasi beberapa operasional matematika menggunakan alat bantu aplikasi Matlab 6.0

Landasan Teori

Pengenalan Matlab

MATLAB adalah salh satu aplikasi canggih untuk komputasi teknik. MATLAB memiliki kemampuan penghitungan, visualisasi, dan pemrograman dalam suatu lingkungan yang mudah untuk digunakan karena permasalahan dan pemecahannya dinyatakan dalam notasi matematika biasa. Kegunaan MATLAB secara umum adalah untuk matematika dan komputasi, pengembangan algoritma, permodelan, simulasi, dan pembuatan *prototype*, analisis data, eksplorasi dan visualisasi, pembuatan aplikasi, termasuk pembuatan antarmuka grafis.

MATLAB adalah sistem interaktif dengan elemen dasar basis data *array* yang dimensinya tidak perlu dinyatakan secara khusus. Hal ini memungkinkan pengguna untuk memecahkan banyak masalah perhitungan teknik, khususnya yang melibatkan matriks dan vektor. Kemampuan lain pada MATLAB diantaranya sanggup mengerjakan operasional matematika mulai dari perhitungan matematika sederhana seperti penambahan, pengurangan, perkalian dan pembagian hingga dapat melakukan perhitungan bilangan kompleks seperti akar dan pangkat, logaritma, operasi trigonometri seperti *sinus, cosinus* dan *tangen*. Seperti kalkulator yang dapat

diprogram, MATLAB dapat digunakan untuk menyimpan dan memanggil data. Pemakai dapat membuat, menjalankan dan menyimpan sederetan perintah untuk mengotomatisasi perhitungan suatu persamaan penting, melakukan pembandingan logika dan mengatur urutan pelaksanaan perintah.

Strukturisasi Aplikasi Matlab

MATLAB terdiri dari beberapa jendela pada layar editor monitor. Dari semua jendela itu jendela *Command* merupakan tempat interaksi utama MATLAB. *Prompt* yang digunakan adalah >>. Pada saat jendela *Command* aktif, kursor seharusnya tampak di sebelah kanan *prompt*. *Prompt* dan *kursor* menunjukkan bahwa MATLAB sedang menunggu untuk menjawab suatu pertanyaan matematika.

Variabel

Seperti bahasa komputer lainnya, MATLAB mempunyai aturan penamaan variabel yaitu Variabel harus terdiri dari satu kata. Nama variabel harus diawali dengan huruf, diikuti dengan sembarang bilangan, huruf atau garis bawah. Karakter-karakter tanda baca tidak diperbolehkan karena banyak diantaranya mempunyai arti tersendiri dalam MATLAB. Contoh: a_b_c_d_e, Raihan181002. MATLAB mempunyai beberapa variabel khusus yaitu :

Tabel 1. Variabel Khusus

Variabel khusus	Nilai
ans	Nama variabel untuk hasil apapun
pi	Perbandingan antara keliling lingkaran dengan garis tengahnya
eps	Bilangan terkecil sedemikian rupa sehingga bila ditambahkan pada satu, menghasilkan bilangan lebih besar dari satu
flops	Jumlah operasi floating point
inf	Tak terhingga, misalnya 1/0
NaN atau nan	Bukan suatu bilangan, misalnya 0/0
i dan j	I=j=√-1
nargin	Jumlah argumen input suatu fungsi
nargout	Jumlah argumen output suatu fungsi
realmin	Bilangan real positif terkecil yang dapat digunakan
realmax	Bilangan real positif terbesar yang dapat digunakan

Fasilitas dasar pembentukan array MATLAB diringkas dalam Tabel 2 berikut:

Tabel 2. Fasilitas Dasar Pembentukan Array

Pembentukan Array Dasar	
X=[2 2*pi sqrt(2) 2-3j]	Menciptakan vektor baris x yang memuat elemen-elemen yang diberikan
X=awal:akhir	Membuat vektor baris x dimulai dengan awal, kenaikan satu, diakhiri pada atau sebelum akhir
X=awal:kenaikan:akhir	Membuat vektor baris x diawali dengan awal, kenaikan sebesar kenaikan diakhiri pada atau sebelum akhir

Metode Penelitian

Penelitian ini menggunakan logika algoritma matematika yang diimplementasikan menggunakan pemrograman Matlab.

Hasil dan Pembahasan

Matematika Sederhana

Contoh Kasus 1: Budi pergi ke toko buku dan membeli dua buku DOS seharga Rp. 24.500 perbuku, satu buku UNIX seharga Rp. 54.800 dan empat buku Microsoft Office seharga Rp. 13.950 perbuku. Berapa jumlah barang yang dibeli Budi dan berapa total yang harus dibayar? Penyelesaian masalah ini dengan MATLAB dapat digunakan beberapa cara.

Cara pertama menggunakan pendekatan yang sama seperti menggunakan kalkulator:

Program 1:

```
>> 2*24500+54800+4*13950

ans =

      159600
```

Sebagian besar kasus MATLAB tidak mempedulikan spasi dan bahwa perkalian lebih tinggi prioritasnya dibanding penjumlahan. MATLAB menyebut jawaban sebagai ans (singkatan dari *answer*) untuk kedua penghitungan tersebut.

Cara kedua adalah dengan menyimpan informasi dalam variabel MATLAB:

Program 2:

```
>> DOS=2

DOS =

     2

>> UNIX=1

UNIX =

     1

>> OFFICE=4

OFFICE =

     4

>> HARGA=DOS*24500+UNIX*54800+OFFICE*13950

HARGA =

     159600
```

Pada contoh program 2 tersebut terdapat tiga variabel MATLAB yaitu DOS, UNIX dan OFFICE untuk menyimpan jumlah buku yang dibeli untuk masing-masing judul buku. Seteleh selesai memasukkan setiap perintah, MATLAB menampilkan hasilnya kecuali jika pada akhir perintah disertai dengan semicolon (titik koma). Tanda titik koma pada akhir perintah MATLAB akan mengerjakan perintah tersebut tetapi tidak menampilkan hasilnya. Variabel BARANG dan HARGA digunakan untuk menamai jumlah buku yang dibeli serta total harga yang harus dibayar.

Pada setiap langkahnya MATLAB mengingat informasi yang sudah ada sehingga perhitungan dapat dilanjutkan misalnya dengan menghitung harga rata-rata buku yaitu dengan rumus: **harga total dibagi dengan jumlah buku** sehingga perintah pada matlab adalah:

Program 3:

```
>> RATA_HARGA=HARGA/BARANG

RATA_HARGA =

       22800
```

Mengubah nilai variabel

Contoh pada kasus I ternyata buku UNIX yang dibeli adalah 2 eksemplar. Jika melihat Program 2 maka secara umum perintah yang harus dirubah adalah mengganti nilai variabel UNIX yang semula satu menjadi dua setelah perintah ***>> barang=dos+unix+office*** dan memanggil kembali variabel BARANG yang mewakili jumlah barang yang dibeli. Secara lengkap dapat dilihat pada Program 4 berikut:

Program 4:

```
>> DOS=2

DOS =

     2

>> UNIX=1

UNIX =

     1

>> OFFICE=4;
>> OFFICE=4

OFFICE =

     4

>> BARANG=DOS+UNIX+OFFICE

BARANG =

     7
```

Ternyata nilai variabel BARANG tidak berubah yaitu 7. Hal ini terjadi karena **MATLAB tidak menghitung jumlah barang yang dibeli berdasarkan nilai terbaru** dari UNIX. Saat MATLAB melakukan perhitungan, **MATLAB mengerjakannya dengan nilai-nilai yang diketahuinya pada saat suatu perhitungan dikerjakan.** Hal tersebut dapat diatasi dengan menghapus variabel UNIX dengan menggunakan perintah **CLEAR**. Beberapa contoh bentuk perintah clear yaitu :

>> clear unix → menghapus variabel UNIX
>> clear dos harga → menghapus variabel DOS dan HARGA
>> clear ba → menghapus semua variabel yang diawali dengan huruf ba
>> clear → menghapus semua variabel pada ruang kerja

Perintah **clear** agak berbahaya dan harus digunakan dengan hati-hati karena semua variabel yang dihapus **tidak dapat dipanggil kembali**. Perintah dan variabel yang yang digunakan akan diingat oleh MATLAB. Perintah dan variabel tersebut dapat dipanggil kapanpun.

Contoh kasus 2 tentang penggajian karyawan: Akan kita kerjakan menggunakan perintah **variabel**. Diketahui bahwa upah lembur dihitung berdasarkan jam lembur karyawan. Upah lembur perjam adalah Rp 2500. Gaji pokok dihitung berdasarkan golongan dimana golongan I = Rp 300000 dan golongan II = Rp 400000. Pada bulan November 2012 diketahui ada 55 karyawan golongan I dan 30 karyawan golongan II dan daftar karyawan yang lembur adalah
Eko 26 jam lembur, Budi 32 jam lembur, Andi 14 jam lembur, Cici 21 jam lembur, Icha 8 jam lembur, dan Siti 18 jam lembur. Bagaimanakah matlab mendefinisikannya?
Program 5 :

```
>> g1=300000
g1 =
      300000
>> g2=400000
g2 =
      400000
>> kry_g1=55
kry_g1 =
    55
>> kry_g2=30
kry_g2 =
    30
```

```
>> jl_eko=26
jl_eko =
    26
>> jl_budi=32
jl_budi =
    32
>> jl_andi=14
jl_andi =
    14
>> jl_cici=21
jl_cici =
    21
>> jl_icha=8
jl_icha =
     8
>> jl_siti=18
jl_siti =
    18
```

a. Hitunglah Jumlah keseluruhan gaji untuk pegawai golongan I .

```
>> Jum_gaji_kry_g1=kry_g1*g1
Jum_gaji_kry_g1 =
    16500000
```

b. Jumlah keseluruhan gaji untuk pegawai golongan II.

```
>>  Jum_gaji_kry_g2=kry_gl*g2

Jum_gaji_kry_g2 =

    22000000
```

c. Jumlah upah lembur untuk pegawai yang lembur.

```
>> jum_upl=u*(jl_eko+jl_budi+jl_andi+jl_cici+jl_icha+jl_siti)

jum_upl =

      297500
```

d. Rata-rata upah lembur

```
>> Rata_ul=jum_upl/6

Rata_ul =

  4.9583e+004
```

ARRAY I

Aplikasi MATLAB juga dapat digunakan untuk membuat ARRAY. Contoh kasus 3 berikut tentang Array Sederhana. Diketahui $Y = \sin(x)$; $0 \leq x \leq \pi$. Jika dihitung maka akan diperoleh hasil berikut:

Tabel 3. Hasil Perhitungan

x	0	.1π	.2π	.3π	.4π	.5π	.6π	.7π	.8π	.9π	Π
y	0	.31	.59	.81	.95	1.0	.95	.81	.59	.31	0

Seperti terlihat, x dan y adalah daftar terurut bilangan-bilangan. Nilai pertama y berkaitan dengan nilai pertama x; nilai kedua y berkaitan dengan nilai kedua x, dan seterusnya. Karena keterurutan ini, maka pada umumnya setiap elemen dalam x dan y ditandai dengan *subscript*, misalnya x^1 adalah elemen pertama x, y5 adalah elemen kelima y, dan seterusnya.

Pembuatan array pada MATLAB dilakukan dengan mudah cukup dengan mengikuti struktur tabel 3.

Program 6 :

```
x =

  Columns 1 through 9

        0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850    2.1991    2.5133

  Columns 10 through 11

    2.8274    3.1416

>> y=sin(x)

y =

  Columns 1 through 9

        0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878

  Columns 10 through 11

    0.3090    0.0000
```

Pembuatan array dalam MATLAB yang perlu dilakukan hanyalah mengetikkan kurung kotak kiri, memasukkan elemen-elemen dengan dipisahkan oleh spasi atau koma, kemudian menutup array dengan kurung kotak kanan. Jika diperhatikan bahwa menghitung nilai sinus dari x akan langsung mengikuti bentuk x.

Pengalamatan Array

Pada MATLAB elemen-elemen array diakses menggunakan *subscript* misalnya x1 adalah elemen pertama x, y5 adalah elemen kelima y, dan seterusnya. Contoh berikutnya adalah untuk mengetahui alamat Array.

Contoh:

```
>> x(3)     %elemen ketiga x

ans =

    0.6283

>> (5)      %elemen kelima y

ans =

     5
```

Pengaksesan suatu blok elemen, MATLAB menyediakan notasi kolom.

```
>> x(1:5)

ans =

         0    0.3142    0.6283    0.9425    1.2566
```

Perintah tersebut digunakan untuk mengetahui nilai elemen pertama sampai elemen kelima.

```
>> x(7:end)

ans =

    1.8850    2.1991    2.5133    2.8274    3.1416
```

Perintah tersebut untuk mengetahui nilai elemen mulai dari elemen ke 7 dan berhenti sampai elemen terakhir. Pada hal ini kata ***end*** berarti elemen terakhir dalam array x.

```
>> y(3:-1:1)

ans =

    0.5878    0.3090         0
```

elemen-elemen di atas adalah elemen ketiga, kedua, dan pertama dalam urutan ke bawah. 3:-1:1 berarti "mulai dari kolom 3, hitung turun satu-satu, dan berhenti saat 1",

Akan kita coba untuk mengembangkan dua array. Perhatikan contoh berikut :

Program 7:

```
>> a=1:5,b=1:2:9

a =

     1     2     3     4     5


b =

     1     3     5     7     9
```

Perintah tersebut digunkan untuk menciptakan 2 array.

Program 8 :

```
>> c=[b a]

c =

     1     3     5     7     9     1     2     3     4     5
```

Program 8 tersebut adalah Array c yang terdiri dari elemen-elemen b diikuti elemen-elemen a.

Program 9 :

```
>> d=[a(1:2:5) 1 0 1]

d =

     1     3     5     1     0     1
```

Program 9 tersebut untuk menciptakan Array d yang terdiri dari elemen pertama, ketiga, kelima dari a diikuti oleh tiga elemen tambahan.

Orientasi Array

Pada contoh sebelumnya array hanya memuat satu baris dan beberapa kolom maka vektor-vektor itu disebelum vektor baris. Sebenarnya dimungkinkan juga untuk membentuk suatu array berbentuk vektor kolom, dengan satu kolom dan beberapa baris. Cara langsung untuk membuat vektor kolom adalah menyebutkan elemen-elemennya satu persatu dengan dipisahkan dengan titik koma (,).

```
>> c=[1;2;3;4;5]

c =

     1
     2
     3
     4
     5
```

Contoh tersebut merupakan cara menciptakan vektor baris menggunakan format notasi kolom. Memisahkan elemen dengan spasi atau koma membuat elemen berada dalam kolom yang berbeda, sedangkan memidahkan elemen dengan titik koma membuat elemen berada dalam baris yang berbeda.

```
>> a=1:5

a =

     1     2     3     4     5
```

Cara lain adalah dengan membentuk vektor baris kemudian ditranspose menjadi vektor kolom dengan menggunakan operator *transpose* MATLAB (‘);

```
>> b=a'

b =

     1
     2
     3
     4
     5
```

jika suatu array dapat berupa vektor baris atau vektor kolom, wajarlah jika array dapat juga mempunyai banyak baris dan banyak kolom. Artinya, array dapat mengambil bentuk berupa matriks. Koma dan spasi digunakan untuk memisahkan elemen-elemen di baris tertentu; titik koma digunakan untuk memisahkan baris,

```
>> g=[1   2   3   4 ;   5   6   7   8]

g =

     1     2     3     4
     5     6     7     8
```

Pada contoh tersebut g merupakan array atau matrik dengan 2 baris dan 4 kolom. Titik koma memberitahu MATLAB untuk memulau baris baru antara 4 dan 5.

```
>> g=[1   2   3   4
      5   6   7   8
      9   10  11  12]

g =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

sebagai ganti titik koma, menekan tombol Enter saat memasukkan matriks juga dapat membuat MATLAB memulai baris baru.

Contoh kasus 4 : Diketahui bahwa C=[10 5 2 7] . Pemodelan dalam Matlab sebagai berikut:

```
>> C=[10    5    2    7]

C =

   10     5     2     7
```

Bagaimanakah cara menghitung I = -2C ?

```
>> I=-2*C

I =

   -20   -10    -4   -14
```

Untuk menghitung H = ½ C^T (^{T}= Transpose) bagaimanakah pemodelan dalam matlab?

```
>> H=1/2*C'

H =

    5.0000
    2.5000
    1.0000
    3.5000
```

Implementasi pada Matlab transpose disimbolkan dengan tanda petik satu diatas (‘).

Hitunglah Y=I+7.5C. Berikut simulasi Matlab:

```
>> Y=I+7.5*C

Y =

   55.0000   27.5000   11.0000   38.5000
```

ARRAY II

Pada bagian ini membahas tentang Array dengan elemen bilangan satu atau bilangan nol. MATLAB menyediakan fungsi-fungsi untuk menciptakan Array yang semua elemennya satu atau nol.

```
>> ones(3)

ans =

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

Model tersebut adalah untuk membuat array 3x3 dengan nilai elemen 1.

```
>> zeros(2,5)

ans =

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

Contoh selanjutnya adalah

```
>> r=[1   2   3   2; 4   5   6   10; 7   8   9   11];
>> size(r)

ans =

     3     4
```

```
>> ones(size(r))

ans =

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

jika dipanggil dengan argumen tunggal, ones(n) atau zeros(n). MATLAB membuat array n x n yang semua elemennya satu atau nol. Jika dipanggil dengan dua argumen ones(x,y) atau zeros(x,y) MATLAB membuat array yang mempunyai x baris dan y kolom yang semua elemennya satu atau nol. Cara menciptakan array berlemen satu atau nol dengan ukuran yang sama dengan array yang lain, dapat menggunakan fungsi size.

Memanipulasi Array

Karena array dan matriks merupakan hal yang mendasar dalam MATLAB, maka terdapat banyak cara untuk memanipulasinya. MATLAB menyediakan cara mudah untuk menyisipkan, mengambil dan mengatur kembali sebagian dari matriks dengan mengidentifikasi *subscript* yang berkaitan.

```
>> g=[1   2   3   4  ;   5    6    7    8]

g =

     1     2     3     4
     5     6     7     8
```

```
>> g(2,3)=0      % mengubah elemen baris 2 kolom 3 menjadi nol

g =

     1     2     3     4
     5     6     0     8
```

Model tersebut digunakan untuk mengubah elemen baris 2 kolom 3 menjadi nol

```
>> g(:,2)=3

g =

     1     3     3     4
     5     3     0     8
```

Perintah tersebut digunakan untuk membuat semua elemen di kolom 2 menjadi 3.

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

D =

     1     2     3
     4     5     6
     7     8     9
```

```
>> E=D(3:-1:1,1:3)

E =

     7     8     9
     4     5     6
     1     2     3
```

Kedua perintah tersebut adalah pemodelan dan simulasi untuk menciptakan matriks E dengan urutan baris D yang dibalik, yaitu baris 1 pada matriks D menjadi baris 3 pada matriks E dan baris 3 pada matriks D menjadi baris 1 pada matriks E.

```
>> E=D(3:-1:1,:)

E =

     7     8     9
     4     5     6
     1     2     3
```

Dari contoh tersebut kemudian penulis kembangkan lagi. Pada contoh tersebut titik dua berarti semua kolom. Jadi : adalah singkatan untuk 1:end atau 1:3 alam contoh ini karena matriks D memiliki 3 kolom.

```
>> B=[D E(:,[1 3])]

B =

     1     2     3     7     9
     4     5     6     4     6
     7     8     9     1     3
```

Perintah tersebut untuk membuat B dengan dasar matriks D ditambah semua elemen baris pada matriks E pada kolom ke 1 dan 3 ke sisi kanan matriks D.

```
>> I=D(1:2,2:3)

I =

     2     3
     5     6
```

membuat I dengan mengambil elemen baris 1 dan 2 dan kolom 2 dan 3 pada matriks D.

```
>> I=D(1:2,2:3)

I =

     2     3
     5     6
```

membuat I dengan mengurutkan kolom-kolom D.

Perhatikan beberapa perintah berikut :

```
>> I=D(:)

I =

     1
     4
     7
     2
     5
     8
     3
     6
     9
>> U=D

U =

     1     2     3
     4     5     6
     7     8     9
>> U(:,2)=[ ]

U =

     1     3
     4     6
     7     9
```

Pemodelan dan simulasi tersebut untuk mendefinisikan U dengan membuang elemen semua baris pada kolom 2. Saat mengeset sesuatu menjadi matriks kosong, sesuatu itu akan terhapus, mengakibatkan matriks berkurang menjadi apa yang tertinggal. Yang harus diingat MATLAB hanya bisa menghapus seluruh kolom atau baris saja, artinya tidak bisa dihapus sebagian baris atau kolom saja. Contohnya >>D(2,2)= [ ], artinya yang dihapus hanya elemen baris 2 kolom 2 saja.

Bagaimanakah pemodelan dan simulasi transpose suatu matriks?. Berikut contohnya :

```
>> U=U'

U =

     1     4     7
     3     6     9
```

Perintah tersebut merupakan contoh transpose suatu matriks.

```
>> U(2,:)=[ ]

U =

     1     4     7
```

Contoh tersebut untuk menghapus elemen pada baris 2 untuk semua kolom.

OPERASI MATRIKS

Matlab juga familiar untuk pemodelan dan simulasi untuk operasional matriks. Berikut contoh kasus untuk menciptakan matrik hingga manipulasinya. Jika g adalah sebuah matriks dengan bentuk sebagai berikut : g=[1 2 3 4 ; 5 6 7 8 ; 9 10 11 12] dan h adalah matriks dengan bentuk sebagai berikut, h=[1 1 1 1 ; 2 2 2 2 ; 3 3 3 3].

Bagaimanakah untuk menghitung penjumlahan matrik g+h ?

```
>> g=[1  2   3   4 ; 5   6   7   8 ; 9   10   11    12]

g =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> h=[1  1   1   1 ; 2   2   2   2 ; 3   3   3   3]

h =

     1     1     1     1
     2     2     2     2
     3     3     3     3

>> g+h

ans =

     2     3     4     5
     7     8     9    10
    12    13    14    15
```

Artinya perintah di atas akan menambahkan h pada g dengan pola elemen ke elemen artinya elemen baris 1 kolom 1 pada matriks g akan ditambahkan dengan elemen baris 1 kolom 1 matriks h dan seterusnya.

Bagaimanakah dengan perkalian? Misalnya untuk menghitung 2*g-h ?

```
>> 2*g-h

ans =

     1     3     5     7
     8    10    12    14
    15    17    19    21
```

Perintah tersebut merupakan hasil dari mengalikan g dengan 2 kemudian dikurangi h.

Matematika array menggunakan aturan prioritas yang sama dengan ekspresi skalar untuk menentukan pengurutan pengerjaan. Perkalian dan pembagian dengan pola elemen ke elemen dapat juga dilakukan, namun dengan notasi yang sedikit berbeda:
Hitung g.*h ?

```
>> g.*h

ans =

     1     2     3     4
    10    12    14    16
    27    30    33    36
```

Perkalian dilakukan dengan mengalikan elemen-elemen yang seletak dari g dan h dengan menggunakan simbol **perkalian titik .*.** Titik yang mendahului simbol perkalian biasa memberitahukan pada MATLAB untuk melakukan perkalian elemen ke elemen.

Contoh berikut adalah perintah untuk pembagian g./h ?

```
>> g./h

ans =

    1.0000    2.0000    3.0000    4.0000
    2.5000    3.0000    3.5000    4.0000
    3.0000    3.3333    3.6667    4.0000
```

Bagaimanakah jika h.\g ?

```
>> h.\g

ans =

    1.0000    2.0000    3.0000    4.0000
    2.5000    3.0000    3.5000    4.0000
    3.0000    3.3333    3.6667    4.0000
```

pembagian array denngan array dapat menggunakan garis miring kanan atau miring kiri. Pada kedua kasus, array di bawah garis miring akan membagi array di atas garis miring.

Operasional pemangkatan di modelkan dengan g.^2.

```
>> g.^2

ans =

     1     4     9    16
    25    36    49    64
    81   100   121   144
```

memangkatkan setiap elemen g.
Selanjutnya hitung 2.^g ?

```
>> 2.^g

ans =

           2           4           8          16
          32          64         128         256
         512        1024        2048        4096
```

memangkatkan dua dengan setiap elemen pada g.

Hitunglah g.^h

```
>> g.^h

ans =

           1           2           3           4
          25          36          49          64
         729        1000        1331        1728
```

memangkatkan setiap elemen g dengan setiap elemen yang bersesuaian pada h. Dalam kasus ini baris pertama tidak berubah, baris kedua dikuadratkan dan baris ketiga dipangkatkan dengan tiga.

Contoh kasus yang lain adalah jika diketahui a=[1 2 3 ; 4 5 6 ;7 8 0]; b=[366 ;804;351];

Menghitung a.b ?

```
>> a=[1  2   3 ; 4   5   6 ;7   8    0];
>> b=[366 ;804;351];
>> a*b

ans =

        3027
        7590
        8994
```

Berapakah det(a) ?

```
>> det(a)

ans =

    27
```

Hitunglah x=inv(a)*b ?

```
>> x=inv(a)*b

x =

   25.0000
   22.0000
   99.0000
```

Pada perintah inv(a) adalah fungsi yang menghitung A-1; dan operator perkalian matriks * tanpa didahului titik adalah perkalian matriks. Cara yang lebih disukai untuk menemukan solusi adalah menggunakan operator pembagian-kiri matriks:

Hitunglah x=a\b ?

```
>> x=a\b

x =

   25.0000
   22.0000
   99.0000
```

Matrik Khusus

MATLAB menyediakan sejumlah matriks khusus. Beberapa diantaranya yaitu a=[1 2 3; 4 5 6]; b=find(a>10). Bagaimanakah hasilnya ?

```
>> a=[1   2   3; 4   5   6];
>> b=find(a>10)

b =

    []
```

variabel b adalah matriks kosong. MATLAB menghasilkan matriks kosong jika suatu operasi tidak menghasilkan apapun. Dalam contoh di atas tidak ada elemen a yang lebih besar dari 10. matriks kosong mempunyai ukuran nol, tetapi nama variabelnya tetap ada di ruang kerja MATLAB.

Coba zeros(3)? Hasilnya adalah :

```
>> zeros(3)

ans =

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

Bagaimanakah dengan ones(2,4) ?

```
>> ones(2,4)

ans =

     1     1     1     1
     1     1     1     1
```

Bagaimanjika zeros(3)+pi ?

```
>> zeros(3)+pi

ans =

    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416
```

suatu contoh membuat suatu matriks 3x3 yang semua elemennya sama dengan π.

Hitung eye(3) ?

```
>> eye(3)

ans =

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

Model tersebut untuk membuat matriks identitas berukuran 3x3.

POLINOMIAL

Akar

Cara menemukan akar suatu polinomial yaitu suatu nilai yang membuat polinomial bernilai nol, adalah problem yang muncul dalam berbagai bidang ilmu. Pada MATLAB polinomial direpresentasikan sebagai vektor baris dari koefisien-koefisien polinomial tersebut dalam urutan dari derajat tertinggi ke derajat terendah. Sebagai contoh, polinomial $x^4-12x^3-0x^2+25x=116$ dituliskan sebagai p=[1 -12 0 25 116]

```
>> p=[1    -12    0    25    116]

p =

     1   -12     0    25   116
```

perhatikan bahwa suku yang berkoefisien nol juga harus dimasukkan, karena MATLAB tidak dapat mengetahui suku mana yang berkoefisien nol. Menggunakan bentuk ini maka akar polinomial dapat ditemukan dengan fungsi ***roots***

```
>> r=roots(p)

r =

  11.7473
   2.7028
  -1.2251 + 1.4672i
  -1.2251 - 1.4672i
```

Karena baik suatu polinomial maupun akarnya adalah vektor dalam MATLAB, MATLAB menggunakan konvensi bahwa polinomial haruslah vektor baris sementara akarnya adalah vektor kolom. Pemberian akar-akar

suatu polinomial maka dimungkinkan untuk menemukan polinomialnya. Hal tersebut dikerjakan dengan menggunakan fungsi ***poly*** Perintahnya adalah sebagai berikut :

```
>> pp=poly(r)

pp =

    1.0000  -12.0000   -0.0000   25.0000  116.0000

>>
```

Perkalian

Perkalian polinomial dikerjakan dengan fungsi ***conv***. Hasil perkalian dua polinomial berikut $a(x)= x^3-2x^2+3x+4$ dengan $b(x)=x^3+4x^2+9x+16$. Pada pemrograman matlab ditulisdengan perintah berikut:

```
>>
>> a=[1    -2    3    4];b=[1    4    9    16];
>>
>> c=conv(a,b)

c =

     1     2     4    14    11    84    64
```

hasilnya adalah $c(x)= x^6+6x^5+20x^4+50x^3+75x^2+84x+64$. perkalian lebih dari dua polinomial dapat dikerjakan dengan menggunakan fungsi conv berulang-ulang.

Penjumlahan

MATLAB tidak menyediakan fungsi langsung untuk penjumlahan polinomial. Penjumlahan array biasa dapat digunakan jika kedua vektor polinomial mempunyai ukuran yang sama. Untuk menjumlahkan polinomial a(x) dengan b(x) di atas:

```
>> d=a+b

d =

     2     2    12    20
```

yang menghasilkan $d(x)= 2x^3-6x^2+12x+20$.jika kedua polinomial berbeda derajatnya maka polinomial dengan derajat yang lebih rendah harus ditambah dengan koefisien-koefisien nol untuk membuatnya mempunyai derajat yang sama.

```
>> e=c+[0    0    0    d]

e =

     1     2     4    16    13    96    84
```

Koefisien-koefisien nol harus diberikan di awal polinomial, bukan pada akhir, sebab koefisien-koefisien pangkat x harus bersesuaian.

Pembagian

Pada beberapa kasus tertentu adalah perlu membagi suatu polinomial dengan polinomial lain. Pada MATLAB hal tersebut dapat dilakukan dengan menggunakan fungsi ***deconv***. Menggunakan polinomial b dan c pada contoh sebelumnya:

Hitunglah [q,r]=deconv(c,b) berapakah hasilnya ?

```
>> [q,r]=deconv(c,b)

q =

     1    -2     3     4


r =

     0     0     0     0     0     0     0
```

hasil yang ditampilkan menyatakan bahwa polinomial b yang dibagi dengan polinomial c menghasilkan polinomial q dan sisa r, yang dalam kasus ini adalah nol karena perkalian dari b dengan q sama dengan c.

Turunan

Karena turunan suatu polinomial mudah dilakukan, MATLAB menyediakan fungsi ***polyder*** untuk turunan polynomial g=[1 6 20 48 69 72 44]. Pemodelannya h=polyder(g)

```
>> g=[1     6     20     48     69     72     44];
>> h=polyder(g)

h =

     6     30     80    144    138     72
```

Simpulan

Penelitian ini menghasilkan pemodelan dan simulasi beberapa operasional matematika sederhana, array satu dimensi, array dua dimensi, matrik, dan Polinomial menggunakan alat bantu aplikasi Matlab 6.0

Daftar Pustaka

Bambang Sridadi. 2009. *Pemodelan dan Simulasi Sistem : Teori, Aplikasi dan Contoh Program Dalam Bahasa C*. Bandung : Informatika.

JJ Siang. 2004. *Jaringan syaraf Tiruan dan Pemrogramannya Menggunakan Matlab*. Yogyakarta : Andi.

Muhammad Arhami. 2007. *Pemrograman Matlab*. Yogyakarta : Andi.

Sahid. 2007. *Pengantar Komputasi Numerik dengan Matlab*. Yogyakarta : Andi.