Muhammad Muzaini
Muhammad Ikram
Nasrun Syahrir
Sirajuddin

Pemrograman MATLAB

Cara Cepat dan Mudah Memahami Bahasa Pemrograman

Editor: Ma'rup

Pemrograman MATLAB

Cara Cepat dan Mudah Memahami Bahasa Pemrograman

Muhammad Muzaini
Muhammad Ikram
Nasrun Syahrir
Sirajuddin

Editor: Ma'rup

Haura Utama

Pemrograman MATLAB "Cara Cepat dan Mudah Memahami Bahasa Pemrograman",
Penulis: Muhammad Muzaini, dkk
diterbitkan pertama kali oleh Penerbit Haura Utama, 2022

15.5 x 23 cm, 157 hlm

Editor: Ma'rup
Penata isi: Zulfa
Perancang sampul: Nita

CV. Haura Utama
Anggota IKAPI Nomor 375/JBA/2020
Nagrak, Benteng, Warudoyong, Sukabumi
+62877-8193-0045 haurautama@gmail.com

Cetakan I, Oktober 2022

ISBN: 978-623-492-096-3

penerbithaura.com

PRAKATA

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Ahamdulillah Puji dan Syukur penulis panjatkan, atas berkat dan karunia-Nya sehingga penulis dapat menyelesaikan buku ini. Salawat serta salam semoga selalu tercurah kepada kita semua.

Kondisi kemajuan dan pesatnya perkembangan ilmu pengetahuan dan teknologi, terutama teknologi informasi bagaikan dua sisi mata uang, terdapat sisi positif maupun sisi negatif yang ditimbulkan oleh kemajuan ilmu pengetahuan dan teknologi komunikasi informasi pada siswa.

MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. MATLAB adalah singkatan dari Matrix Laboratory karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks. MATLAB versi pertama dirilis pada tahun 1970 oleh Cleve Moler. Pada awalnya, MATLAB didesain untuk menyelesaikan masalah-masalah persamaan aljabar linear. Seiring berjalannya waktu, program ini terus mengalami perkembangan dari segi fungsi dan performa komputasi.

Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga memiliki keunggulan umum lainnya, seperti analisis dan eksplorasi data, pengembangan algoritma, pemodelan dan simulasi, visualisasi plot dalam bentuk 2D dan 3D, hingga pengembangan aplikasi antar muka grafis. Dalam ruang lingkup perguruan tinggi, MATLAB digunakan sebagai

alat pembelajaran pemrograman matematika, teknik, dan sains pada level pengenalan dan lanjutan, sedangkan dalam dunia industri, MATLAB dipilih sebagai alat penelitian, pengembangan, dan analisis produk industri.

MATLAB dapat dioperasikan pada sistem operasi Windows, Linux, maupun macOS. Selain itu, MATLAB juga bisa dihubungkan dengan aplikasi atau bahasa pemrograman eksternal lainnya, seperti C, Java, .NET, dan Microsoft Excel. Dalam MATLAB tersedia pula kotak kakas (toolbox) yang dapat digunakan untuk aplikasi-aplikasi khusus, seperti pengolahan sinyal, sistem kontrol, logika fuzzy, jaringan saraf tiruan, optimasi, pengolahan citra digital, bioinformatika, simulasi, dan berbagai teknologi lainnya. Buku tentang *Pemrograman MATLAB Cara Cepat dan Mudah Memahami Bahasa Pemrograman* berisi tentang teori dasar tentang Matlab dan cara mengoperasikannya.

Berdasarkan hal tersebut di atas, maka buku ini sangat penting untuk dimiliki dan menjadi referensi bagi siapa saja yang ingin belajar tentang pemograman.

Akhirnya dengan segala kerendahan hati, penulis mengharapkan kritik dan saran dari semua pihak demi penyempurnaan buku ini. Semoga buku ini dapat memberikan mamfaat bagi pengembangan ilmu pengetahuan di masa yang akan datang Semoga Alla swt, senantiasa melimphakan berkah bagi kita semua. Aaminn YRA.

Penulis

DAFTAR ISI

TENTANG MATLAB

Apa itu MATLAB?

Pertama dirilis di tahun 1970 oleh MathWorks, MATLAB adalah salah satu platform yang paling banyak digunakan untuk mengolah angka dan bahasa pemrograman. Ada banyak sekali hal yang bisa dilakukan dengan MATLAB, khususnya yang terkait dengan rumpun ilmu di bidang teknik, matematika, dan sains.

Menurut MathWorks, MATLAB adalah platform pemrograman yang menggunakan bahasa berbasis matriks sehingga umumnya digunakan untuk menganalisis data, membuat algoritma, serta menciptakan pemodelan dan aplikasi. Aplikasi ini juga sering dimanfaatkan untuk mengembangkan *deep learning*, *machine learning*, dan hal-hal terkait lainnya. Siapa saja bisa menggunakan MATLAB, mulai dari pelajar hingga profesional. Secara garis besar, begitulah kegunaan MATLAB.

Kegunaan MATLAB

Akan tetapi, menurut Dummies, berikut adalah 5 fungsi MATLAB yang sering digunakan.

1. **Menyelesaikan masalah *engineering***

   Matematika adalah bagian besar dari ilmu teknik, oleh karena itu MATLAB sangat bermanfaat untuk menyelesaikan berbagai masalah yang dihadapi para *engineer*.

Dengan MATLAB yang mampu mengolah angka dan model rumit, solusi dapat dirancang, dicoba, dan terus dikembangkan dengan lebih cepat.

2. Mengolah permasalahan aljabar linear

Aljabar linear tak hanya dibutuhkan untuk menyelesaikan soal di bangku sekolah.

Dalam dunia kerja, aljabar linear salah satunya digunakan untuk menghitung *return on investment* (ROI).

Selain itu, rumus ini juga bisa bermanfaat untuk:

- memprediksi jumlah *turnover* perusahaan
- *inventory control*
- menyusun rencana finansial
- membuat keputusan bisnis yang tepat

Karena angka yang diolah biasanya dalam jumlah besar, tentunya penggunaan MATLAB bisa sangat membantu prosesnya.

3. Analisis numerik

Analisis numerik adalah bagian dari ilmu statistika yang sering berguna untuk membuat keputusan di berbagai bidang ilmu keteknikan, seperti arsitektur, teknik sipil, dan bahkan teknik industri.

Dengan MATLAB, pengolahan datanya jadi lebih mudah.

4. Mengolah data riset

MATLAB adalah program yang dapat digunakan untuk memvalidasi hasil riset dengan berbagai metode.

Selain itu, hasil riset juga bisa divisualisasikan dengan jelas.

5. Simulasi

Di MATLAB, kita bisa membuat suatu pemodelan ataupun algoritma untuk menyelesaikan masalah.

Program ini bisa menguji keberhasilan model atau algoritma tersebut dengan menyimulasikan hasil akhirnya.

Sistem MATLAB

Melansir Cooperative Institute for Meteorological Satellite Studies, ada lima bagian utama dari sistem MATLAB.

Lima bagian sistem MATLAB tersebut adalah sebagai berikut.

1. MATLAB *language*

MATLAB menggunakan *high-level matrix/array language* yang bisa mengolah berbagai program atau fungsi yang kompleks.

2. *Working environment*

MATLAB *working environment* adalah kumpulan *tool* dan fasilitas yang tersedia untuk bekerja di platform ini.

Dengan *tool* dan fasilitas tersebut, kamu bisa mengelola variabel yang digunakan serta mengimpor dan mengekspor data.

Tidak itu saja, masih ada banyak fungsi lain yang digunakan untuk mengembangkan apa saja yang kamu butuhkan dengan MATLAB.

3. Sistem grafis

Sistem grafis MATLAB adalah bagian yang digunakan untuk memproses gambar, visualisasi data, membuat animasi, dan mempresentasikan grafis.

4. ***Mathematical function library***

Di MATLAB, tentunya salah satu bagian yang paling penting adalah fungsi matematisnya.

MATLAB sudah dilengkapi dengan kumpulan algoritma komputasional dari yang sederhana hingga sangat kompleks.

Semua ini bisa diproses dalam kecepatan yang tinggi, asal perangkat kerasnya mendukung.

5. ***Application Program Interface*** **(API)**

API di MATLAB adalah fitur yang memberi akses pada para penggunanya untuk menulis program C dan Fortran.

BAB I
PENGENALAN MATLAB

MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. MATLAB adalah singkatan dari Matrix Laboratory karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks. MATLAB versi pertama dirilis pada tahun 1970 oleh Cleve Moler. Pada awalnya, MATLAB didesain untuk menyelesaikan masalah-masalah persamaan aljabar linear. Seiring berjalannya waktu, program ini terus mengalami perkembangan dari segi fungsi dan performa komputasi.

Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga memiliki keunggulan umum lainnya, seperti analisis dan eksplorasi data, pengembangan algoritma, pemodelan dan simulasi, visualisasi plot dalam bentuk 2D dan 3D, hingga pengembangan aplikasi antar muka grafis. Dalam ruang lingkup perguruan tinggi, MATLAB digunakan sebagai alat pembelajaran pemrograman matematika, teknik, dan sains pada level pengenalan dan lanjutan, sedangkan dalam dunia industri, MATLAB dipilih sebagai alat penelitian, pengembangan, dan analisis produk industri.

MATLAB dapat dioperasikan pada sistem operasi Windows, Linux, maupun macOS. Selain itu, MATLAB juga bisa dihubungkan dengan aplikasi atau bahasa pemrograman eksternal lainnya, seperti C, Java, .NET, dan Microsoft Excel. Dalam MATLAB tersedia pula kotak kakas (toolbox) yang dapat

digunakan untuk aplikasi-aplikasi khusus, seperti pengolahan sinyal, sistem kontrol, logika fuzzy, jaringan saraf tiruan, optimasi, pengolahan citra digital, bioinformatika, simulasi, dan berbagai teknologi lainnya.

TUJUAN PEMBELAJARAN:

Setelah mengikuti mata kuliah ini, mahasiswa mampu:

1. menyebutkan jendela-jendela pada MATLAB;
2. mampu menyebutkan operator-operator MATLAB;
3. mampu menganalisis kesalahan perintah pada MATLAB;
4. mampu membuat variabel;
5. mampu membuat string karakter;
6. mampu mengenali tipe-tipe data;
7. mampu menuliskan fungsi-fungsi matematika;
8. mampu menggunakan fasilitas help;

Materi

A. Mengenal Matlab

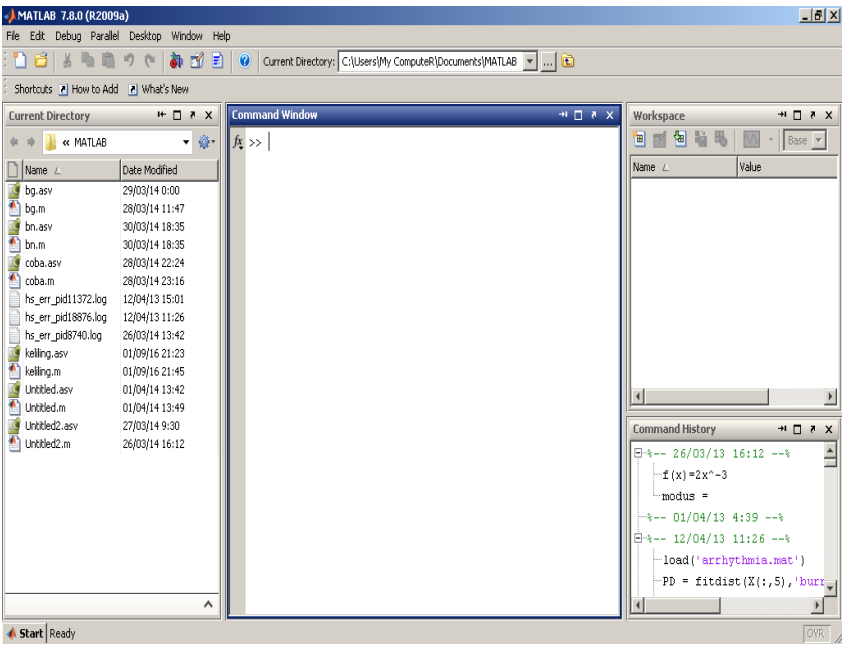
MATLAB adalah software buatan The Mathwork, Inc., yang sangat bermanfaat untuk menyelesaikan berbagai masalah komputasi numerik. Perangkat lunak ini menawarkan kemudahan dan kesederhanaan dalam menyelesaikan permasalahan yang berhubungan dengan vektor dan matriks. Memperoleh inversi matriks dan menyelesaikan persamaan linear merupakan contoh permasalahan yang dapat dipecahkan dengan cara yang sangat singkat dan mudah sekali.

Untuk menangani persoalan-persoalan yang spesifik, MATLAB menyediakan sejumlah *toolbox*. Contoh *toolbox*:

- Image processing: ditujukan secara khusus untuk melakukan pengolahan citra;
- Signal processing: ditujukan untuk menangani pengolahan isyarat;
- Neural Network: menyediakan berbagai fungsi yang terkait dengan jaringan syaraf tiruan.

B. Menjalankan Matlab

Sebelum mencoba untuk menggunakan MATLAB, perlu untuk menjalankan program tersebut terlebih dahulu. Akan dijumpai tampilan semacam yang telihat pada tampilan berikut.



Beberapa bagian penting yang terdapat pada antarmuka MATLAB adalah seperti berikut.

- *Command Window* atau jendela perintah adalah jendela yang dipakai untukmemberikan perintah secara manual.
- *Workspace* berisi daftar variabel yang diciptakan oleh pemakai dan masih adadalam memori.
- *Command History* mencantumkan perintah-perintah yang pernah diberikan olehpemakai.
- *Current Directory* menyatakan direktori kerja.

C. Menggunakan Matlab Pada Jendela Command

MATLAB memungkinkan kita bekerja pada jendela command (*command window)*. Pada *command window*, kita dapat memberikan perintah MATLAB secara interaktif. Begitu kita menekan tombol Enter, MATLAB akan mengeksekusi perintah yang telah kita berikan.

Pada *command window* tanda >> menyatakan bahwa MATLAB siap menerima perintah dari pengguna. Sebagai contoh, kita bisa mencoba menuliskan

4+5

dan selanjutnya tekanlah tombol Enter.

MATLAB akan memberikan hasil dari perintah tersebut sebagaimana terlihat padagambar berikut.

```
>> 4+5

ans =

     9
```

Hasil
ans =

9

Menyatakan bahwa hasil ekspresi 4+5 adalah 9. Tulisan ans berasal dari kata "*answer*" yang artinya jawaban.

Setiap perintah yang dapat dieksekusi oleh MATLAB disebut sebagaipernyataan (*statement*). Umumnya pernyataan berbentuk:

Variabel = ekspresi

atau

Ekspresi

Berbagai pernyataan yang lain, seperti if dan for, akan dipelajari pada bab lain.

D. Kesalahan Perintah

Apabila perintah yang diberikan tidak dikenal, MATLAB akan menampilkanpesan kesalahan. Sebagai contoh, ketikkan:

>>5+4a ↩

Begitu tombol Enter ditekan, MATLAB segera menampilkan kesalahan seperti berikut:

```
??? 5+4a
       |
Error: Unexpected MATLAB expression.
```

MATLAB menunjukkan posisi yang salah.

Untuk memperbaiki ekspresi yang salah, gunakan tombol ↑ untuk menampilkan perintah. Kemudian, lakukan pengeditan dan akhiri dengan menekan tombol Enter.Beberapa tombol yang berguna untuk melakukan pengeditan perintah:

- Del untuk menghapus karakter pada posisi
- ← Backspace untuk menghapus karakter di sebelah kiri kursor.
- ← untuk menggeser kursor ke kiri;
- → untuk menggeser kursor ke kanan;
- ↓ untuk memperoleh perintah setelah perintah sekarang;
- ↑ untuk memperoleh perintah sebelum perintah sekarang;
- Esc untuk mengosongkan perintah pada prompt sekarang.

E. Mengenal Operator

Tanda seperti + pada ekspresi 4+5 dinamakan operator. Operator adalah suatu simbol yang digunakan dalam suatu ekspresi untuk menyatakan suatu operasi tertentu. Selain +, terdapat beberapa operator yang terkait dengan operasi aritmetika.

Tabel 1.1 *Daftar operator aritmetika*

Operator	Keterangan
+	Penjumlahan atau tanda positif
-	Pengurangan atau tanda negatif
*	Perkalian
/ atau \	Pembagian
^	Perpangkatan

Sebagai contoh, jika ingin menghitung

$5 \times 4 + 3$

Kita perlu menuliskannya menjadi:

5*4+3

Catatan: Antara *operand* dan operator bisa diberikan spasi. Penulisan seperti itu lazim dilakukan dengan tujuan agar ekspresi mudah dibaca oleh orang (tidak tampak ruwet, terutama kalau ekspresi panjang). Berikut adalah contohnya.

```
>> 5 * 4 + 3

ans =

    23
```

Gambar 1.3 Contoh ekspresi matematika di *command window*

Pada contoh di atas, 5 * 4 + 3 memberikan hasil berupa 23. Perhatikan pula contoh berikut.

```
>> 7/2

ans =

    3.5000
```

Hasil di atas menyatakan bahwa hasil pembagian 7 dengan 2 adalah 3,5. Perhatikan bahwa tanda pecahan yang digunakan adalah titik, bukan koma. Itulah sebabnya jika katakanlah kita ingin menghitung kita perlu menuliskannya menjadi:

3.5 * 4

Catatan:

- MATLAB memperkenankan penulisan bilangan real dalam notasi sainssemacam:

  2.34e5

  yang berarti dengan $\sqrt[3]{125}$ 2,34 / 10^5. Adapun 2.34e-5 berarti $2,34\times10^{-5}$ atau sama dengan $2,34\times10^{5}$.

- Pada notasi seperti 2.34e-5, tanda e bisa diganti dengan E.

Selain operator /, yang disebut sebagai pembagian kanan, terdapat pula operator \, yang disebut pembagian kiri. Bila 7 / 2 menghasilkan 3,5 maka ekspresi 2 \ 7 juga menghasilkan 3,5. Perhatikan, bahwa pada pembagian kiri (dengan menggunakan operator \), yang membagi diletakkan di sebelah kiri operator \. Operator ^ berguna untuk menangani perpangkatan atau akar dari suatu bilangan. Misalnya, kita inginmenghitung 5^3. Kita bisa menuangkannya seperti terlihat pada gambar berikut.

```
>> 5 ^ 3

ans =

   125
```

Kita juga bisa menghitung seperti berikut:

$$\sqrt[3]{125}$$

Penuangannya dapat dilihat pada gambar berikut.

```
>> 125^(1/3)

ans =

     5
```

F. Mengatur Urutan Pengerjaan

Adakalanya kita perlu mengatur sendiri urutan suatu pengerjaan operasi aritmetika. Mengapa hal ini perlu dilakukan ? Jawabannya adalah karena setiap operator memiliki prioritas pengerjaan yang berbeda-beda. Menurut anda, berapa hasil ekspresi dari 3 + 4 * 2 ? Apakah hasilnya 14 ? Atau 11 ? Jawaban yang benar adalah 11. Mengapa ? Jawabannya adalah karena operator * mempunyai prioritas pengerjaan yang lebih tinggi daripada +.

Tabel 1.2 *Prioritas operator aritmetika*

Operator	Keterangan	Prioritas
+	Tanda positif Penjumlahan	1 4
-	Tanda negatif Pengurangan	1 4
*	Perkalian	3
/ atau \	Pembagian	3
^	Perpangkatan	2

Angka yang lebih kecil dalam prioritas menyatakan bahwa urutan pengerjaannya lebih tinggi. Catatan: bila dalam suatu ekspresi terdapat lebih dari sebuah operator yang memiliki prioritas yang sama, pengerjaan akan dimulai dari kiri ke kanan.

Berdasarkan adanya prioritas yang berbeda-beda, adakalanya kita perlu mengatur sendiri urutan pengerjaan suatu ekspresi aritmetika. Sebagai contoh,dikehendaki untuk menghitung

$$\frac{2+8}{1+4}$$

Kita perlu menuliskannya menjadi:

$$(2+8)/(1+4)$$

Tanda ( ) dapat dipakai untuk menentukan bagian yang perlu dihitung dulu. Contoh dapat dilihat pada gambar berikut.

```
>> ( 2 + 8 ) / ( 1 + 4 )

ans =

     2
```

G. Menggunakan Variabel

Pada Command Window, kita bisa menggunakan variabel. Variabel adalah suatunama yang dapat dipakai untuk menyimpan suatu nilai dan nilai yang ada di dalamnya bisa diubah sewaktu-waktu. Sebelum mempraktekkan penggunaan variabel, aturan tentang cara menamakan variabel perlu diketahui terlebih dulu. Aturan dalam memberikan nama terhadap variabel adalah seperti berikut.

- MATLAB membedakan huruf kecil dan huruf kapital pada penamaan variabel. Dengan demikian, bilangan dan Bilangan adalah dua variabel yang berbeda.
- Nama variabel harus diawali dengan huruf, sedangkan kelanjutannya dapat berupa huruf, angka, atau tanda garis-bawah (_).
- Panjang nama variabel dapat mencapai 31 karakter. Jika nama variabel lebih dari 31 karakter, maka karakter ke-32 dan seterusnya diabaikan.
- Nama variabel bersifat *case sensitive*. Artinya, huruf kapital dan huruf kecil dibedakan. Jadi, nama seperti Bil dan bil dianggap berbeda.

Tabel 1.3 *Nama-nama variabel yang absah dan tidak absah*

Nama variabel yang absah	Nama variabel yang tidak absah
N	_n (awalan variabel harus berupa huruf)
HARGA	Kuartal-3 (Tanda minus tidak diperkenankan)
bulan123	2n (awalan harus berupa huruf)
kuartal_1	Alamat siswa (tidak boleh ada spasi)

Catatan:

- Umumnya variabel skalar menggunakan huruf kecil dan nama larik (vektor ataupun matriks) menggunakan huruf kapital. Namun, tidak ada keharusan untuk menggunakan pedoman tersebut.
- Kata tercadang (misalnya for) tidak dapat digunakan sebagai nama variabel.
- Walaupun fungsi-fungsi bawaan (yang disediakan oleh MATLAB, seperti sqrt dan abs) dapat digunakan sebagai nama variabel, sebaiknya hindari untuk menggunakannya sebagai nama variabel.

Secara umum, penugasan/pemberian nilai ke suatu variabel dilakukan denganbentuk berikut:

Variabel=nilai

Nilai yang diberikan ke variabel dapat berupa suatu konstanta, variabel, atau bahkansuatu ekspresi. Contoh:

```
>> harga_satuan = 6000

harga_satuan =

        6000
```

Pada contoh di atas, variabel harga_satuan diberi nilai 6000.

Jika suatu waktu kita ingin melihat isi variabel tersebut, kita cukup menyebutkan namanya. Contoh:

```
>> harga_satuan

harga_satuan =

        6000
```

Seperti telah disinggung di depan, isi variabel dapat diubah sewaktu-waktu. Contoh berikut menunjukkan cara menaikkan isi variabel harga_satuan sebesar 10%.

```
>> harga_satuan = 1.1 * harga_satuan

harga_satuan =

   6.6000e+003
```

Perlu diketahui, notasi seperti 6.6000e+003

Bermakna $6,6\times10^3$

Contoh di depan sekaligus menunjukkan bahwa suatu variabel bisa diisi dengan hasil suatu ekspresi. Catatan: Ekspresi pada MATLAB bisa saja berbentuk semacam berikut:

M = m + 1

Ekspresi tersebut menyatakan bahwa hasil penjumlahan m dan 1 disimpan ke m. Dengan demikian, ekspresi tersebut sesungguhnya digunakan untuk menaikkan nilai m sebesar 1.

H. Mengenal Variabel Khusus

MATLAB menyediakan sejumlah variabel khusus, yaitu variabel yang dipakai oleh MATLAB dan memiliki makna secara khusus.

Tabel 1.4 *Daftar variabel khusus*

Variabel	Keterangan
Ans	Menampung hasil suatu ekspresi yang dijadikan sebagai sebuah pernyataan

Eps	Bilangan terkecil dalam komputer yang apabila ditambah dengan satu maka akanbernilai lebih besar daripada satu
flops	Jumlah operasi bilangan pecahan
Inf	Berasal dari kata “infinitive” yang artinya adalah tak berhingga (hasil 1/0)
NaN	Berasal dari “Not-a-Number” atau “bukan sebuah bilangan”. Misalnya hasil dari 0/0
Pi	Menyatakan bilangan π
realmin	Bilangan real positif terkecil
realmax	Bilangan real positif terbesar

Perlu diketahui, MATLAB mempunyai batasan bilangan real yang berkisar dari realmin sampai dengan realmax. Bila ada operasi yang menghasilkan nilai di luarjangkauan tersebut, akan terjadi keadaan yang disebut *overflow* atau *underflow*. Disebut *overflow*, jika nilai melebihi batas realmax dan disebut *underflow* jika nilai lebih kecil daripada realmin. Contoh berikut menunjukkan keadaan ini.

```
>> realmin

ans =

  2.2251e-308

>> realmax

ans =

  1.7977e+308

>> x = 2.1e400

x =

   Inf

>> x=2.1e-400

x =

     0
```

Hasil di atas menunjukkan bahwa :

- Keadaan *overflow* akan membuat nilai diubah menjadi tidak berhingga
- Keadaan *underflow* akan membuat nilai diubah menjadi nol

Contoh berikut menunjukkan penggunaan variabel pi untuk menghitung luaslingkaran yang jari-jarinya diketahui.

```
>> jari_jari = 4.5

jari_jari =

    4.5000

>> luas = 0.5 * pi * jari_jari^2

luas =

   31.8086
```

I. Menghilangkan Tampilan Hasil Penugasan Variabel

MATLAB menyediakan mekanisme yang membuat penugasan terhadap suatu variabel tidak membuat MATLAB secara otomatis menampilkan isi variabel tersebut. Hal ini bisa diatur dengan menambahkan titik-koma (;) di belakang pernyataan penugasan. Untuk melihat efeknya, perhatikan contoh berikut.

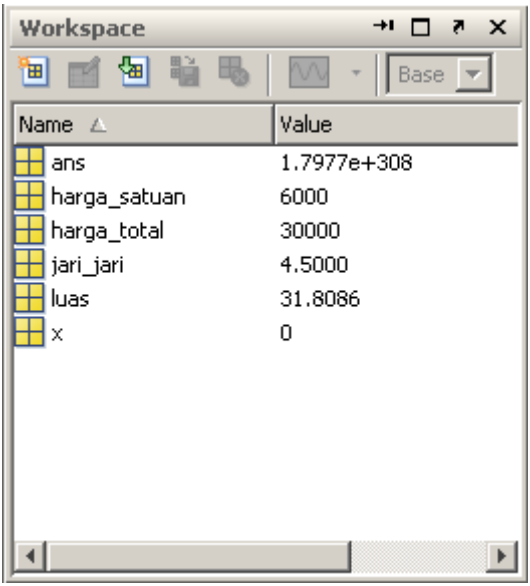
```
>> harga_satuan = 6000;
>> harga_total = 5 * harga_satuan

harga_total =

        30000
```

Tampak bahwa:

- Pada pernyataan pertama, titik koma membuat nilai harga_satuan tidak ditampilkan;
- Pada pernyataan kedua, karena titik-koma tidak diberikan, nilai harga_total secara otomatis ditampilkan.

J. Mengenal Workspace

Ketika kita bekerja dengan MATLAB, semua variabel yang kita ciptakan akan tersimpan dalam memori komputer. MATLAB menggunakan istilah *workspace* untuk menyimpan variabel-variabel tersebut. Catatan: Sebagaimana telah disinggung di depan,variabel yang diciptakan oleh pemakai akan tersaji pada jendela *workspace* . Contoh ditunjukkan di bawah ini.



Untuk mengetahui variabel apa saja yang terdapat pada *workspace* tanpa melihat pada jendela Workspace, kita bisa menggunakan perintah *who*. Contoh:

```
Your variables are:

ans           jari_jari
harga_satuan  luas
harga_total   x
```

Sebuah variabel membutuhkan memori komputer. Agar memori yang dipakai tidak membesar, variabel-variabel yang tidak digunakan bisa kita hapus. Caranya, gunakan perintah clear.

```
>> clear ans
```

Contoh:

Akan menghapus variabel bernama ans dari *workspace*.

Jika kita ingin menghapus beberapa variabel sekaligus, kita bisa menyebutkan semua variabel bersangkutan. Contoh:

```
>> clear harga_satuan harga_total jari_jari
```

menghapus variabel harga_satuan, harga_total, dan jari_jari.

Untuk menghapus semua variabel dalam *workspace*, kita bisa memberikan perintah seperti berikut:

```
>> clear
```

Setelah perintah ini kita berikan, *workspace* menjadi kosong, sebagaimana ditunjukkan pada contoh berikut:

```
>> who
```

MATLAB juga menyediakan perintah untuk menyimpan variabel-variabel dalam *workspace* ke dalam berkas di *disk*. Tentu saja isi berkas seperti itu bisa dipanggil lagi pada kesempatan lain sehingga variabel-variabel yang tersimpan dalam disk akan dimuat ke dalam *workspace*. Untuk penyimpanan variabel ke berkas, mula-mula buatlah dua variabelx dan y seperti berikut:

```
>> x = 5;
>> y = 6;
```

Setelah itu, lakukan penyimpanan dengan memberikan perintah **save**, sebagaimanadiperlihatkan pada gambar berikut.

```
>> save

Saving to: matlab.mat
```

Dengan cara seperti itu, isi *workspace* disimpan ke berkas dengan nama matlab.mat.

Sekarang kita coba untuk menghapus isi *workspace* dengan memberikan perintah

```
>> clear
```

Dengan begitu, semua variabel dihapus. Selanjutnya, berikan perintah **load** seperti terlihat pada gambar berikut sehingga isi berkas matlab.mat akan dimuat ke *workspace*.

```
>> load

Loading from: matlab.mat
```

Untuk melihat isi *workspace* sekarang, berikan perintah **who.** Hasilnya ditunjukkan padagambar berikut:

```
>> who

Your variables are:

x  y
```

Tampak bahwa variabel x dan y ada dalam *workspace*. Catatan:

- Jika dikehendaki untuk menyimpan isi *workspace* ke dalam berkas dengan nama yang diatur sendiri, gunakan perintah semacam berikut:

  ```
  >> save xy
  ```

  Pada perintah seperti itu, MATLAB akan menyimpan variabel-variabel dalam *workspace* ke berkas dengan nama xy.mat.

- Untuk memeuat isis berkas xy.mat, gunakan perintah:

  ```
  >> load xy
  ```

- Bila ingin menyimpan variabel tertentu ke dalam suatu berkas, gunakan perintahsemacam berikut:

save nama_berkas variabel1 variabel2 variabel3

- Untuk memuat variabel tertentu ke dalam *workspace*, gunakan perintah semacam berikut:

  Load nama_berkas variabel1 variabel2 variabel3

Selain who, terdapat pula perintah whos. Perintah whos menghasilkan informasi yang lebih lengkap terhadap variabel-variabel yang terdapat pada *workspace*. Contoh hasil perintah ini dapat dilihat pada Gambar

```
>> whos
  Name        Size              Bytes  Class     Attributes

  x           1x1                   8  double
  y           1x1                   8  double
```

Catatan: *workspace* juga dipakai untuk menampung semua perintah yang pernah kita tuliskan dalam jendela Command. Untuk memanggil perintah yang pernah kita gunakan, kita bisa menekan tombol panah atas (↑). Tekanlah satu atau beberapa kali sampai perintah yang kita kehendaki muncul.

K. Tipe Data

Setiap ekspresi ataupun variabel mempunyai tipe data. Pada MATLAB, tipe data dinyatakan dalam kelas. Sebuah kelas pada dasarnya adalah gabungan antara tipe data dan operasi yang dapat dikenakan terhadap nilai pada tipe tersebut. Berikut adalah nama-nama kelas yang tersedia pada MATLAB.

- Bilangan pecahan atau titik mengambang:
  - **Single:** bilangan pecahan berpresisi tunggal
  - **Double:** bilangan pecahan berpresisi ganda

- Bilangan bulat:
  - **Int8:** bilangan bulat berukuran 8 bit
  - **Int16:** bilangan bulat berukuran 16 bit
  - **Int32:** bilangan bulat berukuran 32 bit
  - **Int64:** bilangan bulat berukuran 64 bit
- Karakter:
  - **Char:** menyatakan sebuah karakter atau deretan karakter (string)
- Logika:
  - **Logical:** menyatakan nilai logika true (benar) atau false (salah).

Untuk membuat suatu variabel dengan kelas tertentu secara eksplisit, kita bisamenggunakan bentuk semacam berikut:

X = kelas (nilai)

Contoh:

```
>> intensitas = uint8(20);
```

Dengan cara seperti itu, intensitas berkelas **unit8.** Perlu diketahui, **uint8** berarti kelasbilangan bulat tak bertanda (hanya mencakup nilai positif).

Kita bisa melihat informasi mengenai variabel intensitas dengan memberikan perintah:

whos intensitas

Hasilnya seperti berikut:

```
>> whos intensitas
  Name            Size              Bytes  Class     Attributes

  intensitas      1x1                   1  uint8
```

Tampak bahwa kelas untuk variabel intensitas berupa **uint8**.

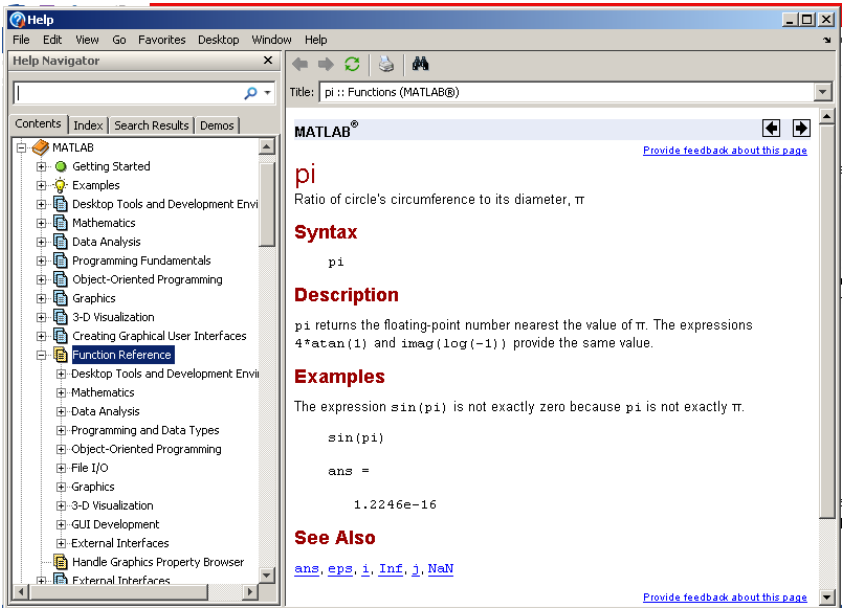
Catatan: untuk mengetahui bilangan terkecil dalam suatu kelas bilangan bulat, kita bisa menggunakan fungsi **intmin.** Misalnya, intmin('uint8') menghasilkan bilangan terkecil dalam kelas **uint8.** Sebaliknya, nilai terbesar untuk kelas bilangan bulat diperoleh dengan menggunakan fungsi **intmax**.

L. Mengenal Fasilitas Help

MATLAB menyediakan fasilitas bantuan yang sangat berguna seandainya kita ingin mengetahui topik tertentu. Untuk menampilkan seluruh topik bantuan, berikan perintah help. Untuk mendapatkan bantuan tentang suatu materi, gunakan help diikuti dengan materi yang penjelasannya ingin kita dapatkan. Contoh:

```
>> help pi

 PI     3.1415926535897....
    PI = 4*atan(1) = imag(log(-1)) = 3.1415926535897....

    Reference page in Help browser
       doc pi
```

Catatan: selain menggunakan help, kita juga bisa menggunakan doc untuk memperoleh bantuan. Kelebihan doc, informasi yang disajikan terformat lebih baik. Contoh, doc pi menghasilkan tampilan seperti berikut.



M. Mengenal Bilangan Kompleks

Sejauh ini, kita telah mengenal operasi bilangan real pada MATLAB. MATLABtidak hanya bisa dipakai untuk menangani bilangan real tetapi juga bisa dipakai untuk memanipulasi bilangan kompleks. Sebagaimana diketahui, bilangan kompleks adalah bilangan yang memiliki bentuk: a + bj dengan j adalah$\sqrt{-1}$. Pada MATLAB, dapat diwakili oleh variabel i atau j. Jadi, kita bisa memilih sendiri salah satu dari kedua tanda tersebut. Contoh penugasan bilangan kompleks ke variabel dapat dilihat di bawah ini.

```
>> x = 5 + 6j

x =

   5.0000 + 6.0000i
```

Berbagai operasi aritmetika seperti pada bilangan real juga bisa dilakukan pada bilangankompleks. Sebagai contoh:

```
>> z = x + 4 - 2j

z =

   9.0000 + 4.0000i
```

Contoh berikut menunjukkan operasi pengurangan:

```
>> z = x - (4 - 2j)

z =

   1.0000 + 8.0000i
```

N. Mengenal Fungsi Matematika

MATLAB menyediakan banyak fungsi yang berguna untuk melakukan operasi tertentu. Fungsi adalah suatu nama yang mewakili suatu operasi tertentu; misalnya, sqrt digunakan untuk menghitung akar kuadrat dan sin dipakai untuk menghitung sinus suatu radian. Dalam hal ini, nilai yang akan dilewatkan ke dalam fungsi disebut sebagai argumen. Contoh: sqrt (25) merupakan bentuk pemanggilan fungsi sqrt dengan argumen berupa angka 25. Maksudnya, fungsi tersebut digunakan untuk menghitung akar kuadratbilangan 25.

Beberapa fungsi hanya melibatkan sebuah argumen, tetapi ada juga yang melibatkan lebih dari satu argumen. Bila argumen fungsi lebih dari sebuah, antarargumen perlu diberi tanda koma. Contoh:

```
>> rem(6, 4)

ans =

     2

>>
```

Pada contoh di atas, antara 6 dan 4 harus dipisahkan dengan koma. Adapun tanda spasi hanya bersifat opsional dan biasanya diberikan agar mudah dibaca oleh orang. Fungsi rem sendiri berguna untuk mendapatkan sisa pembagian. Pada contoh, sisa pembagian 6dengan 4 adalah sebesar 2.

Sebuah fungsi menghasilkan nilai balik (*returnvalue*) dan nilai ini tentu saja dapat diberikan ke variabel. Contoh:

```
>> x = sqrt(25)

x =

     5
```

Pada contoh ini, nilai balik sqrt yaitu akar kuadrat 25 diberikan ke variabel x. Dengan demikian, x berisi 5.

Perlu juga diketahui, argumen fungsi juga bisa berupa pemanggilan fungsi.

Contoh:

```
>> x = sqrt(rem(10, 6))

x =

     2
```

Pada contoh di atas, nilai balik rem(10, 6) dijadikan sebagai argumen fungsi sqrt. Selanjutnya, nilai balik fungsi sqrt diberikan ke variabel x.

Tabel 1.5 *Daftar fungsi dasar matematika*

Fungsi	Keterangan
abs(x)	Fungsi ini menghasilkan nilai absolut suatu bilangan. Jika dikenakan pada bilangan kompleks, hasilnya berupa *magnitude*-nya. Contoh: abs(-5)→5 abs(3 + 4j)→5
angle(x)	Fungsi ini menghasilkan sudut dari suatu bilangan kompleks x.Satuan sudut adalah radian. Contoh: angle(4-4i)→-0.7854
ceil(x)	Fungsi melakukan pembulatan ke bilangan bulat terdekat yangnilainya lebih besar daripada x. Contoh: ceil(4.5)→5 ceil(-4.5)→-4
conj(x)	Fungsi ini menghasilkan *conjugate* dari suatu bilangan kompleks.CONJ(X)=REAL(X)-i*IMAG(X). Contoh: Conj(4 +8j)→4 - 8j Conj(4 – 8j)→4 + 8j
exp(x)	Fungsi ini menghasilkan nilai eksponen dari bilangan x (ex).

fix(x)	Fungsi ini menghasilkan bagian bulat dari suatu bilangan.Contoh: fix(4.5)→4 fix(-4.5)→-4
floor(x)	Fungsi ini menghasilkan bilangan yang merupakan faktor persekutuan terbesar dari bilangan x dan y. Contoh: gcd(60, 24)→12
imag(x)	Fungsi ini menghasilkan bagian imajiner dari suatu bilangan kompleks. Contoh: imag(3 + 4j)→4 imag(3 – 4j)→-4
log(x)	Fungsi ini menghasilkan logaritma alami suatu bilangan (ln x).
log10(x)	Fungsi ini menghasilkan logaritma dari suatu bilangan.Contoh: log10(100)→2 log10(1000)→3
real(x)	Fungsi ini menghasilkan bagian real suatu bilangan kompleks.Contoh: real(3+4j)→3
rem(x, y)	Fungsi ini menghasilkan sisa dari x/y.Contoh:

	Rem(3, 2)→1
round(x)	Fungsi ini menghasikan bilangan bulat yang merupakan pembulatanterdekat terhadap suatu bilangan. Contoh: round(4.5)→4 round(4.1)→4
sign(x)	Fungsi ini menghasilkan bilangan berupa:1 kalau x bernilai lebih dari nol 0 kalau x bernilai nol -1 kalau x bernilai kurang dari nolContoh: sign(5)→1sign(0)→0 sign(-5)→-1
sqrt(x)	Fungsi ini menghasilkan akar kuadrat dari bilangan x.

Tabel 1.6 *Daftar fungsi trigonometri dan hiperbolik*

Fungsi	**Keterangan**
sin(x)	Fungsi ini menghasilkan sinus dari x. Dalam hal ini, x berupa sudut dalam satuan radian.
cos(x)	Fungsi ini menghasilkan cosinus dari x. Dalam hal ini, x berupa sudut dalam satuan radian.
tan(x)	Fungsi ini menghasilkan tangent dari x. Dalam hal ini, x berupa sudut dalam satuan radian.

asin(x)	Fungsi ini menghasilkan inversi sinus dari x. x berupa nilai antara - 1 dan 1. Hasilnya berkisar antara –π/2 dan π/2 dalam satuan radian.
acos(x)	Fungsi menghasilkan inversi kosinus dari x. X berupa nilai antara -1 dan 1. Hasilnya berkisar antara –0 dan π dalam satuan radian.
atan(x)	Fungsi ini menghasilkan inversi tangent dari x. Hasilnya berkisar antara –π/2 dan π/2 dalam satuan radian.
atan2(x, y)	Fungsi ini menghasilkan inversi tangent dari y/x. Hasilnya berkisar antara –π dan π dalam satuan radian.
sinh(x)	Fungsi ini menghasilkan sinus hiperbolik dari x (yaitu $\frac{e^x - e^{-x}}{2}$)
cosh(x)	Fungsi ini menghasilkan sinus hiperbolik dari x (yaitu $\frac{e^x + e^{-x}}{2}$)
tanh(x)	Fungsi ini menghasilkan tangent hiperbolik x.
asinh(x)	Fungsi ini menghasilkan inversi sinus hiperbolik x.
acosh(x)	Fungsi ini menghasilkan inversi kosinus hiperbolik x.
atanh(x)	Fungsi ini menghasilkan inversi tangent hiperbolik x.

Perlu diketahui, hubungan antara sudut dalam derajat dan radian adalah seperti berikut:180^o= π radian
Dengan demikian:

- Sudut_dalam_derajat=sudut_dalam_radian x(180/π)

- Sudut_dalam_radian=sudut_dalam_derajat x π/180

O. Mengenal String Karakter

MATLAB tidak hanya berguna untuk menangani data numerik, melainkan juga bisa digunakan untuk memproses string karakter. String karakter (atau sering disebut dengan nama singkat string) adalah deretan karakter. Sebuah string bisa saja terdiri dari puluhan karakter, satu karakter, atau bahkan tidak mengandung karakter sama sekali. String yang tidak mengandung karakter satupun biasa disebut string kosong.

String ditulis dengan awalan dan akhiran tanda petik-tunggal. Contoh: nama= ‘Siti Nurhaliza”. Pada contoh tersebut, variabel nama diisi dengan string ‘Siti Nurhaliza’. Adapun contoh berikut menunjukkan variabel yang diisi dengan stringkosong: Temp=”.

Bila suatu string mengandung petik-tunggal, petik tunggal perlu ditulis dua kali.

Contoh:

Hari = ‘don’’t’

Dalam hal ini, variabel hari berisi string: don’t, sebagaimana diperlihatkan pada gambar.

```
>> hari = 'don''t'

hari =

don't

>>
```

P. Prioritas Operator dalam Matlab

Prioritas keseluruhan operator dalam MATLAB bisa dilihat pada Tabel 1.7. Informasi yang terkandung di dalamnya sangat bermanfaat ketika kelak kita melibatkan berbagai operator.

Tabel 1.7 *Daftar prioritas dalam MATLAB*

Prioritas	**Keterangan**
1 (tertinggi)	Kurung
2	Perpangkatan
3	Logika NOT (~)
4	Perkalian, Pembagian
5	Penjumlahan, pengurangan
6	Operator relasional
7	Logika AND (&)
8 (terendah)	Logika OR (|)

Q. Mengakhiri Matlab

Untuk mengakhiri MATLAB, kita bisa memilih Exit MATLAB pada menu File atau cukup dengan mengklik tanda exit pada program MATLAB. Tombol Ctrl+Q juga bisa dipakai untuk mengakhiri MATLAB.

Latihan:

1. Buatlah ekspresi matematika di *command window* terdiri dari minimal 4operator berbeda!
2. Buatlah penamaan variabel yang sah dan tidak sah!

BAB II
MENGGUNAKAN BERKAS SKRIP/EDITOR

TUJUAN PEMBELAJARAN:

Setelah mengikuti mata kuliah ini, mahasiswa mampu:

1. mampu menjelaskan skrip;
2. mampu menngetahui perintah komentar;
3. mampu memberikan perintah dalam satu baris;
4. mampu memberikan satu perintah pada beberapa baris;
5. mampu memasukan data lewat keyboard;
6. mampu menggunakan disp;
7. mampu memformat keluaran;
8. mampu menkonversi data;
9. mampu menyusun metodologi penyelesaian masalah.

Materi:

MENGGUNAKAN BERKAS SKRIP

A. Mengenal Berkas Skrip

Sejauh ini perintah-perintah MATLAB dijalankan dengan cara menuliskan perintah secara langsung pada Command Window. Cara ini tidak praktis kalau secara berkala menjalankan

sederetan perintah yang sama. Selain itu memberikan sejumlah perintah yang sama secara berulang akan membosankan. Untuk mengatasi hal itu, MATLAB menyediakan solusi berupa berkas skrip atau yang juga dikenal berkas-M. Disebut berkas-M karena nama belakang berkas berupa .m.

Berkas skrip adalah berkas teks yang dapat diciptakan dengan menggunakan sebarang editor teks (misalnya Notepad pada Windows). Namun, MATLAB juga menyediakan fasilitas untuk membuat berkas-M, yang akan dibahas belakangan. Isi berkas-M adalah deretan perintah MATLAB. Dengan kata lain, berkas ini menghimpun perintah-perintah yang biasanya diketikkan pada *prompt* pada jendela Command.

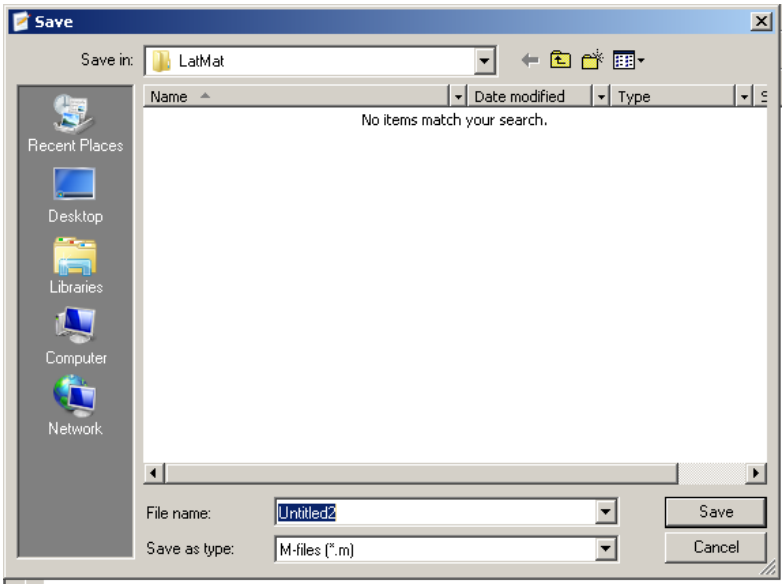
Catatan: Skrip adalah sekumpulan perintah yang diproses melalui interpreter. Namun, skrip juga terkadang disebut sebagai program.

Sebelum membuat berkas-M, siapkan direktori (folder) bernama C:\LatMat. Dapat menggunakan Windows Explorer untuk melakukannya. Selanjutnya, agar direktori tersebut menjadi direktori kerja, berikan perintah seperti berikut pada jendela Command: cd c:\LatMat. Catatan: Di direktori LatMat inilah dapat menyimpan berkas- berkas-M yang dibuat selama praktik membuat skrip. Untuk membuat berkas-M, ikutilah langkah-langkah berikut:

1. Pilih menu File. Lalu, pilihlah New.
2. Pilihlah Blank M-File**.** Langkah ini membuat jendela editor teks ditampilkan.
3. Ketikkan perintah-perintah MATLAB berikut:

```
lebar = 3;
panjang = 6;
keliling = 2 * (panjang + lebar)
```

4. Agar yang diketikkan tersebut tersimpan dalam berkas, pilihlah menu File dan kemudian pilihlah Save. Akan menjumpai tampilan semacam berikut:



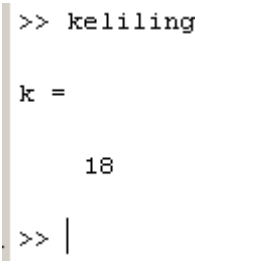
Gambar 2.1 *Kotak dialog untuk menentukan nama berkas*

5. Isikan keliling pada kotak teks yang terletak di sebelah kanan judul Filename:.
6. Klik tombol Save.
7. Tutuplah jendela editor.

   Dengan cara seperti itu, berkas dengan nama keliling.m telah tercipta.

B. Menjalankan Berkas Skrip

Agar perintah-perintah pada berkas keliling.m diproses oleh MATLAB, perlu menjalankan berkas tersebut. Caranya, ketikkan nama depan berkas tersebut pada Command Window dan tekanlah tombol Enter. Contoh:



Gambar 2.2 Contoh eksekusi berkas-M di *command window*

Hasil di atas menyatakan bahwa isi berkas-M keliling.m telah dieksekusi.

C. Menambahkan Komentar

Komentar biasa digunakan dalam berkas-M dengan tujuan untuk memberikan keterangan-keterangan bagi pembaca berkas-M. Penambahan komentar tidak mempengaruhi hasil eksekusi karena komentar memang ditujukan kepada pembaca isi berkas-M, bukan keterangan pada hasil eksekusi berkas tersebut. Komentar bisa berupa tujuan berkas-M, pembuatnya, tanggal yang menyatakan kapan berkas tersebut dibuat atau dimodifikasi ataupun penjelasan terhadap bagian tertentu dalam berkas-M. Penjelasan-penjelasan seperti itu terkadang berguna bagi pembaca berkas-M ataupun bahkan bagi pembuatnya pada suatu waktu ketika dia tidak ingat lagi terhadap perintah- perintah tertentu dalam berkas-M.

Komentar ditandai dengan awalan %. Ketika MATLAB menjumpai tanda %, karakter dimulai dari tanda tersebut hingga akhir baris akan dianggap sebagai komentar sehingga bagian tersebut tidak dijalankan oleh MATLAB. Contoh:

```
>> % berkas: keliling.m
>> lebar = 3; % Lebar persegi panjang
```

Pada contoh pertama, seluruh baris dijadikan sebagai komentar. Adapun pada contohkedua, komentar berupa:

% lebar persegi panjang

Dengan demikian, bagian lebar = 3; akan tetap dijalankan oleh MATLAB.

D. Menuliskan Beberapa Perintah dalam Satu Baris

Normalnya, sebuah baris berisi sebuah perintah. Pada praktiknya, dimungkinkan untuk menuliskan beberapa perintah dalam sebuah baris. Hal ini dapat dilakukan dengan meletakkan tanda titik-koma (;) atau koma (,) antara dua buah perintah.

E. Menuliskan Sebuah Perintah pada Beberapa Baris

Normalnya, sebuah perintah dituliskan dalam sebuah baris. Namun, bisa saja sebuah perintah ditulis dalam beberapa baris asalkan ada pemberitahuan. Pemberitahuan bahwa sebuah perintah mempunyai kelanjutan pada baris berikutnya dilakukan dengan meletakkan tanda titik tiga kali (...) di akhir baris yang kelanjutannya masih ada.

F. Menangani Pemasukan Data dari Keyboard

Berkas-M dapat dipakai untuk menangani pengeksekusian sejumlah perintah. Pada penanganan seperti itu, dimungkinkan untuk memproses data yang berasal dari *keyboard* ketika berkas-M dieksekusi. MATLAB menyediakan pernyataan bernama **input**. Contoh penggunaannya seperti berikut:

```
>> nama = input('Nama Anda:','s')
```

Gambar 2.3 Contoh

Setelah dijalankan, diperoleh tampilan:

Gambar 2.4 Contoh tampilan

Komputer menunggu *user* memasukkan namanya serta menunggu tombol Enter ditekan oleh *user*. Argumen 's' menyatakan bahwa yang dimasukkan dari *keyboard* adalah string. Setelah data dimasukkan oleh *user* melalui *keyboard* dan pemakai menekan tombol Enter, data tersebut akan diletakkan ke variabel nama. Jika argumen kedua tidak disertakan, MATLAB akan mengevaluasi string yang dimasukkan dari *keyboard* dan hasil evaluasi menjadi nilai balik fungsi input.

```
>> x=input('Masukkan suatu ekspresi: ')
Masukkan suatu ekspresi: 1 + 1

x =

     2

>>
```

Gambar 2.5 Contoh penggunaan input

Contoh berkas-M yang melibatkan input: Berkas-M: nama.m

```
>> % Berkas: nama.m
>> nama = input('Nama Anda: ','s'); usia = input('Usia Anda: ','s');
Nama Anda:
Usia Anda:
```

Gambar 2.6 Tampilan eksekusi contoh penggunaan input di *command window*

Selain input, MATLAB menyediakan perintah pause yang juga terkait dengan *keyboard*. Perintah ini berguna untuk menunggu

user menekan sebarang tombol pada *keyboard*. Seperti berikut:

```
>> fprintf('Tekan sebarang tombol'), pause
Tekan sebarang tombol
```

Gambar 2.7 Contoh fprintf

G. Menampilkan dengan Disp

MATLAB menyediakan perintah disp yang berguna untuk menampilkan suatu teks atau isi suatu variabel. Contoh penggunaan perintah ini dapat dilihat pada gambar berikut.

```
>> disp('Tes...tes..123')
Tes...tes..123
>> info = 'Selamat belajar MATLAB';
>> disp(info)
Selamat belajar MATLAB
>>
```

Gambar 2.8 Contoh disp

Contoh berikut menunjukkan penggunaan disp untuk menggabungkan informasi string dan isi sebuah variabel.

```
>> jum = 8;
>> disp(['Jumlah ' num2str(jum) ' buah'])
Jumlah 8 buah
>>
```

Gambar 2.9 Contoh disp

Penggabungan antara string ‘Jumlah’, isi variabel jum, dan string ‘buah’ dilakukan dengan meletakkan ketiga item tersebut dalam tanda [ ]. Perlu juga diketahui, notasi [ ] adalah notasi larik

(*array*) dan elemen dalam larik harus setipe. Itulah sebabnya, jum yang bertipe bilangan perlu dikonversikan ke string melalui num2str.

H. Memformat Keluaran

Penggabungan sejumlah item yang akan menjadi keluaran ke layar memang dapat dilakukan dengan **disp** dengan melibatkan fungsi-fungsi konversi. Adakah cara yang lebih praktis ? Ada, dengan menggunakan fprintf, dengan format seperti berikut:

```
>> fprintf(string_pemformat, nilai1, nilai2, ...)
```

Gambar 2.10 Contoh

Argumen pertama fprintf berupa string yang menentukan format keluaran, sedangkan argumen kedua dan seterusnya berupa nilai-nilai yang akan diformat. Contoh:

```
>> fprintf('%5.2f', 6)
 6.00>>
```

Gambar 2.11 Contoh

Contoh pemformatan bilangan bulat ke dalam bilangan berbasis oktal dan heksadesimalseperti diperlihatkan berikut ini:

```
>> fprintf('%X', 27)
1B>> fprintf('%x', 27)
1b>> fprintf('%o', 27)
33>>
```

Gambar 2.12 Contoh

Contoh di atas menunjukkan bahwa bilangan 27 identik dengan 1B heksadesimal atau 33 oktal. Contoh berikut menunjukkan

fprintf yang melibatkan dua buah nilai yang diformat:

```
>> buah = 'Jeruk';
>> jum = 10;
>> fprintf('Yang dibeli %d %s', jum, buah)
Yang dibeli 10 Jeruk>>
```

Gambar 2.13 Contoh

Pada contoh di atas, string 'Yang dibeli %d %s' digunakan untuk menampilkan string 'Yang dibeli' diikuti dengan suatu nilai bulat (%d), sebuah spasi, dan suatu string (%s).

Tabel 2.1 Contoh pemformatan keluaran bilangan real (~~b~~ berarti sebuah spasi)

Perintah	Hasil
fprintf('%f', pi)	3.141593
fprintf('%3.0f', pi)	~~bb~~3
fprintf('%2.0f', pi)	~~b~~3
fprintf('%1.0f', pi)	3
fprintf('%0.0f', pi)	3
fprintf('%.0f', pi)	3
fprintf('%.1f', pi)	3.1
fprintf('%.2f', pi)	3.14
fprintf('%.3f', pi)	3.142
fprintf('%5.2f', pi)	b3.14

fprintf('%.10f', pi)	3.1415926536
fprintf('%e', pi) fprintf('%.2e', pi) fprintf('%.3e', pi) fprintf('%.4e', pi) fprintf('%.10e', pi)	3.141593e+000 3.14e+000 3.142e+000 3.1416e+000 3.1415926536e+000
fprintf('%E', pi) fprintf('%.2E', pi) fprintf('%.3E', pi) fprintf('%.4E, pi) fprintf('%.10E', pi)	3.141593E+000 3.14E+000 3.142E+000 3.1416E+000 3.1415926536E+000
fprintf('%g', pi) fprintf('%.2g', pi) fprintf('%.3g', pi) fprintf('%.4g', pi) fprintf('%.10g', pi)	3.14159 3.1 3.14 3.142 3.141592654
fprintf('%G', pi) fprintf('%.2G', pi) fprintf('%.3G', pi)	3.14159 3.1 3.14

fprintf('%.4G', pi)	3.142
fprintf('%.10G', pi)	3.141592654

Tabel 2.2 Kode pemformat keluaran

Kode pemformat	Keterangan
f	Format bilangan real dengan notasi biasa
e	Format bilangan real dengan notasi sains
E	Format bilangan real dengan notasi sains. Huruf E digunakan sebagai ganti huruf e
G atau g	Serupa dengan f. Hanya saja untuk bilangan bulat, digit pecahan tidak ditampilkan
o	Memformat bilangan bulat ke dalam bentuk bilangan berbasis octal
x	Memformat bilangan bulat ke dalam bentuk bilangan berbasis heksadesimal.Notasi a, b, c, d, e, dan f (huruf kecil) digunakan untuk menyatakan bilangan 10, 11, 12, 13, 14, dan 15

X	Memformat bilangan bulat ke dalam bentuk bilangan berbasis heksadesimal. Notasi A, B, C, D, E, dan F (huruf kapital) digunakan untuk menyatakan bilangan 10, 11, 12, 13, 14, dan 15
d	Memformat bilangan bulat

I. Konversi Data

MATLAB menyediakan sejumlah fungsi yang berguna untuk mengonversi dari suatu bentuk data ke bentuk data yang lain.

Tabel 2.3 Sejumlah fungsi konversi data

Fungsi	**Keterangan**
dec2hex	Fungsi ini mengonversikan desimal ke heksadesimal. Contoh: dec2hex(10)→ 'A'

hex2dec	Fungsi ini mengonversikan heksadesimal ke desimal. Contoh: hex2dec('A')→ 10 Hex2dec('a')→ 10
oct2dec	Fungsi ini mengonversikan oktal ke desimal. Contoh: oct2dec(10)→ 8
dec2bin	Fungsi ini mengonversikan desimal ke biner.Contoh: dec2bin(9)→ '1001'
bin2dec	Fungsi ini mengonversikan biner ke desimal. Contoh: bin2dec('1001')→ 9
dec2base	Fungsi ini mengonversikan desimal ke suatu sistem bilangan. Argumen kedua menentukansistem bilangan. Contoh: Dec2base(8, 2)→ '1000'

base2dec	Fungsi ini mengonversikan suatu bilangan dalam suatu sistem bilangan ke desimal. Argumen keduamenentukan sistem bilangan. Contoh: base2dec(‘1000’, 2)→ 8
num2str	Fungsi ini mengonversikan bilangan ke string.Contoh: num2str(23)→ ‘23’
str2num	Fungsi ini mengonversikan string ke bilangan.Contoh: str2num(‘23’)→ 23
Abs	Fungsi ini mengonversikan nilai karakter ke ASCII. Contoh: abs(‘A’)→ 65 abs(‘B’)→ 66 abs(‘AB’)→ larik dengan anggota berupa 65 atau66
Setstr	Fungsi ini mengonversikan ASCII ke string. Contoh: setstr(65)→ ‘A’ setstr(66)→ ‘B’ setstr([65 66 67])→ ‘ABC’

J. Mengenal Metodologi Penyelesaian Masalah

Untuk menyelesaikan suatu masalah diperlukan suatu metodologi. Pada dasarnya, metodologi untuk menyelesaikan suatu masalah mencakup 5 langkah seperti berikut.

1. Menyatakan masalah dengan jelas.
2. Menjabarkan hal yang menjadi masukan dan keluaran.
3. Menentukan algoritma untuk menyelesaikan masalah.
4. Menuliskan kode MATLAB untuk menyelesaikan masalah.
5. Menguji dan memperbaiki kode MATLAB sehingga diperoleh hasil yang benar untuk berbagai keadaan masukan.

Pernyataan masalah harus dilakukan dengan jelas dan tepat. Tujuannya adalah agar tidak menimbulkan kesalahpahaman. Contoh pernyataan masalah yaitu: "Menghitung luas lingkaran yang jari-jarinya diketahui". Setelah suatu masalah ditentukan, perlu dicari data yang menjadi masukan dan informasi yang menjadi keluaran.

Masukan bisa lebih dari satu dan begitu juga keluaran. Pada masalah perhitungan luas lingkaran, masukan berupa jari-jari lingkaran dan keluaran berupa luas lingkaran. Setelah masukan dan keluaran ditentukan, suatu algoritma penyelesaian perlu dibuat. Algoritma adalah urutan yang sistematik yang ditujukan agar komputer dapat menyelesaikan suatu masalah. Pada permasalahan menghitung luas lingkaran, algoritma untuk menyelesaikan bisa ditulis seperti berikut.

1. Masukkan (jari-jari)
2. Hitung: luas
3. Tampilkan (luas)

Pada masalah yang kompleks, algoritma dapat diperoleh dengan cara mencarinya pada literatur, menggabungkan beberapa algoritma yang sudah ada dalam literatur, atau dalam keadaan terpaksa harus membuat algoritma itu sendiri. Setelah algoritma ditentukan, algoritma perlu dituangkan dalam kode MATLAB. Berkas-M: lingkaran.m.

```
%Berkas: lingkaran.m
%
%Menghitung luas lingkaran

jari_jari = str2num(input('Jari-jari:', 's'));
luas = 0.5 * pi * jari_jari^2;

fprintf('Luas lingkaran: %f', luas)
```

str2num dimaksudkan untuk mengonversi hasil input yang berupa string ke dalam bilangan. Selanjutnya, hasil konversi ini diberikan ke variabel jari_jari. Setelah kode di atas dituliskan, kode tersebut perlu diuji. Bila ada kesalahan, kode perlu dimodifikasi, dan kemudian diuji lagi. Hal ini dilakukan berulang sampai diperoleh hasil seperti yang diharapkan. Contoh pengeksekusian berkas-M lingkaran:

```
>> lingkaran
Jari-jari:5
Luas lingkaran: 39.269908>>
```

Latihan:

1. Buatlah program sederhana di berkas –M untuk keliling bangun datar.
2. Tambahkan penulisan komentar pada program yang dibuat di no.1.

BAB III
PENGAMBILAN KEPUTUSAN DAN PENGULANGAN

TUJUAN PEMBELAJARAN:

Setelah mengikuti mata kuliah ini, mahasiswa mampu:

1. melakukan pengambilan keputusan dan pengulangan;
2. mampu menerapkan perintah if;
3. mampu menerapkan perintah if bersarang;
4. mampu menerapkan perintah if.. else if;
5. mampu menerapkan perintah switch;
6. mampu menerapkan perintah while;
7. mampumenerapkan perintah for.

Materi:

PENGAMBILAN KEPUTUSAN DAN PENGULANGAN

Dasar yang digunakan untuk pengambilan keputusan adalah operator relasional dan operator logika. Kedua operator ini, juga menjadi komponen penting pada perintah pengulangan. Perlu diketahui, baik operator relasional maupun operator logika hanya menghasilkan dua kemungkinan nilai:

1) 0 (atau disebut salah / **false**)
2) 1 (atau disebut benar / **true**)

Operator relasional adalah semua operator yang berfungsi untuk melakukanpembandingan**.**

Tabel 3.1 Tabel Operator Relasional

Operator	Keterangan	Contoh pemakaian
<	Kurang dari	a < 1
<=	Kurang dari atau sama dengan	a <= 1
>	Lebih dari	a > 1
>=	Lebih dari atau sama dengan	a >=1
==	Sama dengan	a==1
~=	Tidak sama dengan	a~=1

```
>> 2 > 1

ans =

      1

>> 20 < 5

ans =

      0

>> x >= 5

ans =

      1

>> x == 5;
>> x ~= 6

ans =

      1
```

Gambar 3.1 Contoh Eksekusi di *Command Window*

Perhatikan hasil untuk masing-masing pembandingan. Nilai 0 menyatakan salah dannilai 1 menyatakan benar. Ekspresi seperti 2>1 biasa disebut sebagai ekspresi relasional.

Operator logika berguna untuk menggabungkan dua buah ekspresi relasional atau untuk membalik nilai logika dari suatu ekspresi relasional. Berikut ini daftar operator logika dan bentuk pemakaiannya.

Tabel 3.2 Operator Logika

Operator	Keterangan	Contoh pemakaian
&	Operator dan	x & y
|	Operator atau	x | y
~	Operator tidak	~x

Tabel 3.3 Daftar kemungkinan pada operasi dengan & dan |

```
>> x='A';
>> x >= 'a' & x <= 'z'

ans =

     0

>> x = 'c';
>> x >= 'a' & x <= 'z'

ans =

     1

>>
```

Ekspresi X >= ‘a’ & x <= ‘z’, dapat dipakai menentukan apakah isi variabel x berupahuruf kecil atau bukan. Pada contoh, ketika x diisi dengan ‘A’, hasil ekspresi berupa nol (salah), sedangkan ketika x diisi dengan ‘c’ maka ekspresi menghasilkan nilai 1 (benar).

```
>> x='B';
>> (x >= 'a' & x <= 'z') | (x >='A' & x <= 'z')

ans =

     1

>> x='+';
>> (x >= 'a' & x <= 'z') | (x >= 'A' & x <= 'z')

ans =

     0

>>
```

Ekspresi (x >= ‘a’ & x <= ‘z’) | (x >= ‘A’ & x <= ‘z’), digunakan untuk memastikan apakah variabel x berisi huruf (huruf kecil atau huruf kapital). Hasilnya berupa 0 kalaux tidak berisi huruf dan 1 kalau x berisi huruf.

Operator ~ berfungsi untuk membalik nilai logika. Dengan bentuk pemakaian berupa ~x Ekspresi tersebut menghasilkan:

- Benar kalau x bernilai salah, atau
- Salah kalau x bernilai benar.

```
>> a = 0;
>> ~(a == 5)

ans =

     1

>> ~(a == 0)

ans =

     0

>>
```

Ekspresi ~(a == 5), menghasilkan nilai 1 (benar) disebabkan a == 5 sendiri menghasilkan salah. Dengan melibatkan ~, maka diperoleh keadaan sebaliknya (yaitu benar). Sedangkan, ~(a == 0) menghasilkan nilai 0 (salah) disebabkan ekspresi a == 0 menghasilkan nilai benar. Khusus untuk mendukung operasi "atau eksklusif", MATLAB menyediakan fungsi bernama xor, dengan bentuk pemakaian xor(x, y).

Tabel 3.4 Xor

x	y	Xor(x, y)
Salah	Salah	Salah
Salah	Benar	Benar

Benar	Salah	Benar
Benar	Benar	Salah

A. Pernyataan If

Untuk menangani pengambilan keputusan, MATLAB menyediakan struktur **if** dengan bentuk:

If *ekspresi*

pernyataan_pernyataan

End

Dan

if *ekspresi*

pernyataan_pernyataan_1

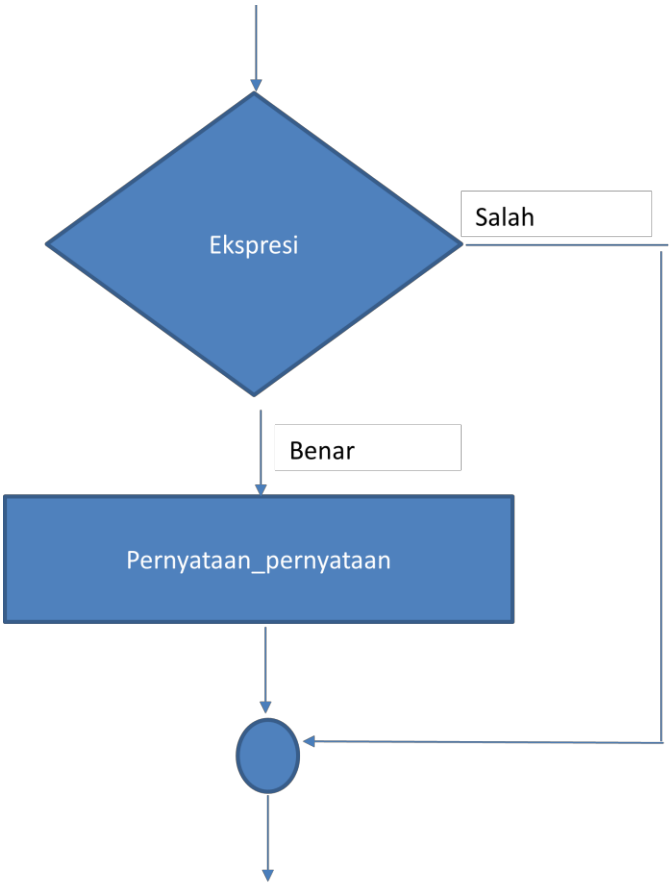
else

pernyataan_pernyataan_2

end

Pada bentuk pertama, bagian *pernyataan_pernyataan* (bisa lebih dari sebuah pernyataan) hanya akan dijalankan kalau *ekspresi* bernilai benar.

Diagram alir pernyataan if sederhana (tanpa else):



Penjelasan pada bentuk kedua sebagai berikut.

1. Bagian

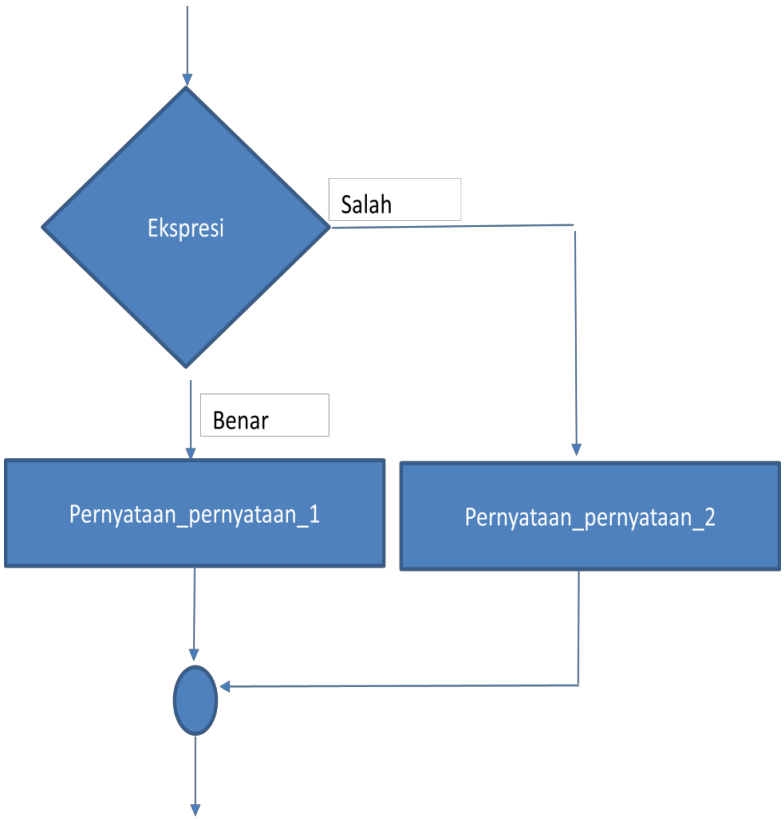
   pernyataan_pernyataan_1

   hanya akan dijalankan kalau *ekspresi* bernilai benar.

2. Bagian

   pernyataan_pernyataan_2

   hanya akan dijalankan kalau *ekspresi* bernilai salah.



Pernyataan *if* dapat tidak harus dipraktikkan dalam bentuk berkas-M. Bisa dipraktikkan pada jendela *command* secara langsung. Contoh:

```
>> x = 5;
>> if x >= 0
        disp('Positif')
   end
Positif
>>
```

Tulisan "Positif"muncul karena nilai x memang lebih besar daripada nol.

```
>> x = -5;
>> if x >= 0
      disp('Positif')
   end
>>
```

Apa hasilnya ? Tidak ada hasil atau dengan kata lain, Disp('Positif') Tidak dijalankan mengingat x >= 0 menghasilkan nilai salah disebabkan x bernilai -5.

Berkas-M: tesif.m

```
% Berkas: tesif.m
x = input('Masukkan sebuah bilangan: ');
if x >= 0
    disp('Positif')
else
    disp('Negatif')
end
```

Pada berkas-M di atas, nilai x diisi melalui *keyboard* ketika berkas-M dijalankan. Kemunculan tulisan *Positif* atau *Negatif* ditentukan oleh ekspresi X >= 0.

```
>> tesif
Masukkan sebuah bilangan: 5
Positif
>> tesif
Masukkan sebuah bilangan: -5
Negatif
>>
```

B. Pernyataan if Bersarang

Kedua bentuk if sebelumnya hanya bisa menangani dua kemungkinan. Bagaimana halnya kalau ada tiga kemungkinan atau lebih ? Salah satu solusinya adalah dengan meletakkan pernyataan if di dalam pernyataan if, atau dikenal dengan sebutan if bersarang (*nested if*).

Contoh berikut memberikan gambaran tentang penyelesaian persamaan

$ax^2 + bx + c = 0$

Persamaan tersebut mempunyai solusi berupa tiga kemungkinan, yaitu:

1. x_1 dan x_2 berupa bilangan real yang berbeda jika berupa bilangan positif (lebih besar daripada 0).
2. x_1 dan x_2 berupa bilangan real yang sama jika berupa bilangan nol.
3. x_1 dan x_2 berupa bilangan kompleks jika berupa bilangan negatif.

Secara umum, nilai x_1 dan x_2 dihitung dengan menggunakan rumus:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

```
%Berkas: akar.m

disp('Menghitung akar persamaan ax^2 + bx + c = 0')
a = input('koefisien dari x^2 = ');
b = input('koefisien dari x = ');
c = input('konstanta = ');

d = b ^ 2 - 4 * a * c;

if d > 0
   disp('Dua akar real yang berbeda')

   x1 = (-b + sqrt(d)) / (2 * a);
   x2 = (-b - sqrt(d)) / (2 * a);

   fprintf('x1 = %.0f\n', x1); %untuk memformat keluaran yaitu x1
   fprintf('x2 = %.0f\n', x2); %untuk memformat keluaran yaitu x2
else
   if d == 0
      disp('Dua buah akar yang sama')

      x = (-b ) / (2 * a);
```

```
        fprintf('x1 = x2 = %.0f\n', x);
    else
        disp('Dua buah akar kompleks')

        x1 = (-b + sqrt(d)) / (2 * a);
        x2 = (-b - sqrt(d)) / (2 * a);

        fprintf('x1 = ')
        disp(x1)

        fprintf('x2 = ')
        disp(x2)
    end
end
```

```
>> akar
Menghitung akar persamaan ax^2 + bx + c = 0
a = 1
b = 4
c = 1
Akar real dan berbeda
x1 = -0.267949
x2 = -3.732051
>> akar
Menghitung akar persamaan ax^2 + bx + c = 0
a = 1
b = -10
c = 25
Akar kembar
x1 = x2 = 5.000000
>> akar
Menghitung akar persamaan ax^2 + bx + c = 0
a = 1
b = 1
c = 1
Akar kompleks
x1 =   -0.5000 + 0.8660i

x2 =   -0.5000 - 0.8660i
```

C. Pernyataan if..elseif

Khusus untuk menangani persoalan yang mempunyai tiga kemungkinan atau lebih, MATLAB menyediakan struktur berbentuk seperti berikut:

If *ekspresi_1*

pernyataan_pernyataan_1

elseif *ekspresi_2*

pernyataan_pernyataan_2

...

else *pernyataan_pernyataan_n*

end

Tanda ... menyatakan bahwa bagian **elseif** bisa lebih dari sebuah. Pada bentuk di atas,bagian

pernyataan_pernyataan_1

Hanya akan dijalankan kalau *ekspresi_1* bernilai benar. Bagian

pernyataan_pernyataan_2

Hanya akan dijalankan kalau *ekspresi_2* bernilai benar dan *ekspresi_1* bernilai salah.

Bagian

pernyataan_pernyataan_n

Hanya dijalankan kalau tidak ada ekspresi pada **if** maupun **elseif** yang bernilai benar.

Tabel 3.5 *Kriteria penentuan skor*

Kriteria	Skor
Nilai ≥ 90	A
70 ≤ Nilai < 90	B
60 ≤ Nilai < 70	C
50 ≤ Nilai < 60	D
Nilai ≤ 50	E

Contoh berkas-M menentukan skor A, B, C, D, atau E

```
%Berkas: skor.m
nilai = input('Nilai (0 s/d 100): ');

if nilai >= 90
    hasil = 'A';
elseif nilai >= 70
    hasil = 'B';
elseif nilai >= 60
    hasil = 'C';
elseif nilai >= 50
    hasil = 'D';
else
    hasil = 'E';
end
fprintf('Skor: %c\n', hasil)
```

```
>> skor
Nilai (0 s/d 100): 90
Skor: A
>> skor
Nilai (0 s/d 100): 75
Skor: B
>> skor
Nilai (0 s/d 100): 67
Skor: C
>> skor
Nilai (0 s/d 100): 58
Skor: D
>> skor
Nilai (0 s/d 100): 25
Skor: E
>>
```

D. Pernyataan Switch

Pernyataan switch sebenarnya merupakan bentuk lain dari perintah **if** yang berguna untuk melakukan pengambilan keputusan yang melibatkan banyak alternatif.

Switch *ekspresi atau nama variabel*

case ekspr_case_1

pernyataan_pernyataan_1

case *{ekspr_case21, ekspr_case22,*

ekspr_case23,...}

...

otherwise

pernyataan_pernyataan_n

end

Contoh:

```
% Berkas: tesswitch.m
a = input('Masukkan arah mata angin (Inggris/Indonesia): ', 's');
switch lower (a)
   case {'utara', 'north'}
      disp('Utara / North')
   case {'selatan', 'south'}
      disp('Selatan / South')
   case {'barat', 'west'}
      disp('Barat / West')
   case {'timur', 'east'}
      disp('Timur / East')
   case 'kulon'
      disp('Jangan pakai bahasa Jawa')
   otherwise
      disp('Masukkan satu kata aja ya!')
end
```

Contoh eksekusi switch di *command window*.

```
>> tesswitch
Masukkan arah mata angin (Inggris/Indonesia): kulon
Jangan pakai bahasa Jawa
>> tesswitch
Masukkan arah mata angin (Inggris/Indonesia): barat daya
Masukkan satu kata aja ya!
>> tesswitch
Masukkan arah mata angin (Inggris/Indonesia): selatan
Selatan / South
>>
```

E. Pernyataan While

Pernyataan while merupakan perintah yang berguna untuk menangani suatu pengulangan.

While *ekspresi*

pernyataan_pernyataan

end

Contoh:

```
%Berkas: x10.m

pencacah = 1;
while (pencacah < 11)
    disp('MATLAB')
    pencacah = pencacah + 1;
end
```

Hal yang terpenting yang perlu diperhatikan, pada pengulangan seperti itu, ada variabel yang dijadikan sebagai pencacah untuk menghitung tulisan MATLAB yang sudah ditampilkan. Perintah pencacah = pencacah + 1;

menaikkan isi variabel pencacah setiap kali tulisan MATLAB telah ditampilkan. Contoh berikut menunjukkan cara menampilkan bilangan 1 sampai dengan 10 dengan menggunakan while

```
% Berkas: satusd10.m

pencacah = 1;
while (pencacah < 11)
    fprintf('%d\n', pencacah)

    pencacah = pencacah + 1;
end
```

Hal terpenting dalam menggunakan while, harus ada bagian dalam while yang membuat suatu ketika ekspresi while bernilai salah sehingga pengulangan segera berakhir. Jika keadaan seperti itu tidak dipenuhi, akan terjadi pengulangan selamanya. Jika menjumpai pengulangan yang tampaknya tidak pernah berakhir, tekanlah tombol Ctrl+C untuk menghentikan pengeksekusian while.

Contoh persoalan komputasi yang menggunakan while. Dikehendaki untuk memperoleh bilangan Faktor Persekutuan Terbesar (*Greatest Common Divisor*), yakni bilangan positif terbesar yang menjadi pembagi baik m maupun n. Dengan algoritma sebagai berikut:

1. Masukkan(m, n)
2. r sisa pembagian m dengan n
3. ULANG SELAMA

   r ≠ 0

   m ← nn ← r

   r ← Sisa pembagian m dengan nAKHIR-ULANG
4. tampilkan(n)

Pada MATLAB, persoalan faktor persekutuan terbesar sebenarnya dapat dipecahkan dengan menggunakan fungsi gcd dengan bentuk pemanggilan gcd(m, n).

```
%Berkas: fpb.m

disp('Memperoleh Faktor Persekutuan Terbesar bilangan m dan n');
m = input('m = ');
n = input('n = ');

r = rem(m, n); % Memperoleh sisa pembagian m dengan n
while r ~= 0
    m = n;
    n = r;
    r = rem(m, n);
end

fprintf('Hasilnya yaitu: %d', n)
```

Hasil eksekusi program fpb

```
>> fpb
Memperoleh Faktor Persekutuan Terbesar bilangan m dan n
m = 60
n = 24
Hasilnya yaitu: 12>>
```

F. Pernyataan For

MATLAB menyediakan pernyataan **for** juga untuk kepentingan penanganan pengulangan, terutama untuk menangani pencacahan. Format pernyataan **for**:

for variable = ekspr pernyataan_pernyataanend

Contoh berikut menunjukkan penggunaan for untuk menampilkan bilangan dari 1 sampai dengan 10

```
%Berkas: contohfor1.m
for i = 1:10
    fprintf('%d\n', i)
end
```

Ekspresi 1:10 pada for i = 1:10 menyatakan bahwa pada setiap iterasi, i akan bernilaidari 1 hingga 10.

Hasil eksekusinya:

```
>> contohfor1
1
2
3
4
5
6
7
8
9
10
>>
```

Selain notasi x:y, juga bisa menggunakan notasi a:b:c pada ekspresi for.

```
%Berkas: contohfor2.m
for i = 20:-2:0
    fprintf('%d\n', i)
end
```

Ekspresi 20:-2:0 pada for menyatakan bahwa nilai i bergerak dari 20 hingga 0 dengan penurunan sebesar -2 untuk setiap iterasi.

```
%Berkas: contohfor3.m

for i = [5 9 1 3 6 8]
    fprintf('%d\n', i)
end
```

Pada persoalan tertentu, adakalanya diperlukan untuk menggunakan for di dalam for.

```
% Berkas: tabelkali.m

%Membuat judul kolom
fprintf('     ');
for i = 1:8
     fprintf('%3d', i)
end
fprintf('\n'); % Pindah baris

%Menampilkan tabel perkalian
for i = 1:8
     fprintf('%3d', i) %Judul baris

     for j = 1:8
          fprintf('%3d', i*j)
     end

     fprintf('\n'); %Pindah baris
end
```

Pada contoh di atas, for j = 1:8 berada dalam for i = 1:8. Artinya, untuk setiap nilai i, forj = 1:8 akan dieksekusi kembali.

G. Pernyataan Break

Pernyataan break berguna untuk mengakhiri eksekusi suatu pernyataan for ataupun while. Bila diletakkan dalam suatu for(while) yang berada dalam for(while), break hanya memberikan efek mengakhiri for(while) terdalam.

Contoh untuk memperlihatkan efek break:

```
% Berkas: contohbr.m

for i = 1:10
    if i == 5
        break
    end

    fprintf('%d\n', i)
end
```

H. Pernyataan *Continue*

Seperti halnya break, pernyataan continue digunakan bersama for atau while. Secara umum, continue digunakan untuk mengatur eksekusi ke iterasi berikutnya. Pada pernyataan for, continue membuat semua pernyataan di bawahnya akan diabaikan dan variabel pencacah for dinaikkan (atau diturunkan) ke nilai berikutnya. Lalu, eksekusi dilanjutkan ke bagian awal pernyataan dalam tubuh for sepanjang batas akhir pada variabel for belum tercapai.

```
% Berkas: count.m
for i = 1:10
    if i == 5
        continue
    end
    fprintf('%d\n', i)
end
```

Pada while, kontrol eksekusi dilanjutkan ke pengujian ekspresi while.

```
%Berkas: count2.m

i = 1;
while i < 11
    if i == 5
        i = i + 2;
        continue
    end

    fprintf('%d\n', i)
    i = i +1;
end
```

Pada contoh di atas, ketika i bernilai 5 maka nilai i dinaikkan sebesar 2 (menjadi 7) dan kemudian perintah continue membuat bagian i < 11 pada while diuji kembali. Dengan demikian, nilai 5 dan 6 tidak pernah ditampilkan, sebagaimana diperlihatkan pada hasilberikut.

BAB IV
MENGGUNAKAN LARIK

TUJUAN PEMBELAJARAN:

Setelah mengikuti mata kuliah ini, mahasiswa mampu:

1. mampu menjelaskan konsep larik;
2. mampu memberikan notasi untuk mendapatkan jangkauan;
3. mampu melakukan operasi transpose;
4. mampu membentuk matriks;
5. mampu menggunakan opersai scalar terhadap larik;
6. mampu melakukan operasi matematika antarlarik;
7. mampu mengakses larik;
8. mampu memperoleh ukuran larik;
9. mampu melakukan operasi dengan matriks;
10. mampu mengenal sejumlah fungsi larik;
11. mampu memformat larik;
12. mampu membentuk larik secara dinamis.

Materi:

Larik

Sebuah larik (array) dapat menampung sejumlah data yang sejenis. Oleh karena itu, larik sangat berguna untuk menyatakan vektor ataupun matriks.

Catatan:

- Vektor adalah larik dengan 1 dimensi. Vektor kolom adalah vektor dengan satu kolom dan vektor baris adalah vektor dengan satu baris.
- Matriks adalah larik yang berdimensi dua.
- Hanya sebagai kebiasaan, nama larik biasa menggunakan huruf awal berupa kapital. Jadi, hal ini bukanlah suatu keharusan.

Contoh:

$$\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

Vektor tersebut dapat dinyatakan dengan larik dengan cara yang sangat sederhana, yaitusebagai berikut:

V = [5; 6; 7; 8; 9]

Tanda [ ] digunakan untuk menyatakan larik dan tanda titik-koma (;) digunakan untuk memisahkan antarelemen. Dengan cara seperti itu, v berisi vektor kolom (vektor yang mengandung hanya sebuah kolom dan memiliki sejumlah baris).

Sesungguhnya, vektor adalah jenis matriks yang hanya memiliki sebuah kolom atau sebuah baris. Jika sebuah matriks hanya mengandung sebuah kolom dan sebuah baris, bilangan tersebut akan dinamakan sebagai skalar.

Seperti halnya variabel biasa (non-larik), isi larik dapat ditampilkan dengan cukupmenyebutkan namanya.

Contoh:

```
>> V = [ 5; 6; 7; 8; 9];
>> V

V =

     5
     6
     7
     8
     9

>>
```

Bila dikehendaki untuk membuat vektor baris (vektor yang hanya mengandung sebuah baris, tetapi memiliki sejumlah kolom), antarelemen dalam larik perlu ditulis dengan pemisah spasi atau koma.

Contoh: antarelemen dipisahkan oleh spasi

```
>> V = [ 5 6 7 8 9 ];
>> V

V =

     5     6     7     8     9

>>
```

Vektor baris dengan spasi sebagai pemisah antarelemen

Contoh: Antarelemen dipisahkan oleh koma

```
>> V = [ 5, 6, 7, 8, 9 ];
>> V

V =

     5     6     7     8     9
```

Vektor baris dengan koma sebagai pemisah antarelemen. Selain dengan menyebutkan nama larik, isi larik bisa ditampilkan dengan menggunakan disp**.** Contoh:

```
>> V = [ 5, 6, 7, 8, 9 ];
>> disp(V)
     5     6     7     8     9

>>
```

Menampilkan larik dengan disp. Notasi: untuk Menyatakan Jangkauan. Pada pembentukan elemen larik, notasi titik-dua (:) dapat dipakai untuk menyatakan jangkauan. Contoh: V = [1:5] Identik dengan V = [1, 2, 3, 4, 5]. Selain notasi berbentuk x:y, MATLAB juga menyediakan notasi: a: b: c

Pada bentuk ini,

- a menyatakan nilai awal;
- b menyatakan kenaikan untuk elemen berikutnya;
- c menyatakan batas nilai tertinggi dalam jangkauan.

Notasi: untuk Menyatakan Jangkauan

```
>> V = [ 1 : 2 : 7 ]
V =
      1      3      5      7
>> V = [ 1 : 2 : 8 ]
V =
      1      3      5      7
```

Contoh berikut memperlihatkan pembentukan larik yang menggunakan 10:-2:0

```
>> V = 10 : -2 : 0
V =
    10     8     6     4     2     0
>> V = [ 10 : -2 : 0 ]
V =
    10     8     6     4     2     0
>>
```

Pembentukan vektor dengan notasi 10:-2:0. Perlu diketahui, V = 10:-2:0 identik dengan V = [10:-2:0].

A. Operasi Transpos

MATLAB menyediakan operator yang disebut transpos dan dinotasikan dengan petik tunggal (‘). Jika diterapkan pada vektor, operator transpos akan menghasilkan vektor baris

terhadap suatu vektor kolom atau menghasilkan vektor kolom terhadap vektor baris. Bentuk pemakaiannya: V'. Contoh berikut

```
>> V = [1; 2; 3; 4]

V =

     1
     2
     3
     4

>> W = V'

W =

     1     2     3     4

>>
```

digunakan untuk mendapatkan vektor baris dari suatu vektor kolom.

Adapun contoh berikut digunakan untuk mendapatkan vektor kolom dari suatu vektor baris.

```
>> V = [ 1 2 3 4 ]

V =

     1     2     3     4

>> W = V'

W =

     1
     2
     3
     4
```

Selain operator transpos berupa tanda petik, MATLAB juga menyediakan operator yang disebut titik-transpos, yang

dinotasikan berupa tanda titik dan petik tunggal (.'). Kedua operator transpos ini sebenarnya tidak memberikan efek yang berbeda pada bilangan real. Perbedaan hanya akan terlihat jika dikenakan pada bilangan kompleks.

```
>> V = [ 1 + j, 2 - 2j, 3 + 3j ]

V =

   1.0000 + 1.0000i   2.0000 - 2.0000i   3.0000 + 3.0000i

>> V'

ans =

   1.0000 - 1.0000i
   2.0000 + 2.0000i
   3.0000 - 3.0000i

>> V.'

ans =

   1.0000 + 1.0000i
   2.0000 - 2.0000i
   3.0000 + 3.0000i
```

Dapatkah melihat perbedaannya ?

Tampak bahwa operator transpos berupa petik tunggal membuat tanda positif/negatif pada bagian imajiner dibalik. Itulah sebabnya, operator tersebut dikatakan sebagai operator transpos konjugat pada bilangan kompleks.

- Operasi konjugat adalah operasi yang membalik tanda positif/negatif pada bagian imajiner.
- Operator transpos dapat dikenakan pada matriks.

B. Membentuk Matriks

Matriks mempunyai notasi seperti berikut:

$$\begin{bmatrix} a_1 & a_2 & \dots & a_{1n} \\ a_1 & a_2 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_m \end{bmatrix}$$

Ciri khas matriks yaitu mengandung lebih dari sebuah kolom dan baris. Adapun jumlah baris dan kolom tidak harus sama.

Tanda koma atau titik-koma untuk memisahkan elemen. Tanda koma untuk memisahkan elemen dalam baris yang sama, sedangkan tanda titik-koma untuk berpindah ke baris berikutnya. Contoh:

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

M =

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

Alternatif lain, setiap baris dipisahkan dengan Enter. Contoh:

```
>> A = [1, 2, 3
4, 5, 6
7, 8, 9]

A =

     1     2     3
     4     5     6
     7     8     9

>>
```

C. Operasi Skalar terhadap Larik

MATLAB menyediakan cara yang sangat mudah untuk mengenakan operasi nilai skalar terhadap suatu larik. Sebagai contoh, untuk menjumlahkan setiap elemen larik dengan nilai 2, dapat dituliskan sebagai berikut:

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

M =

     1     2     3
     4     5     6
     7     8     9

>> M = 2 + M

M =

     3     4     5
     6     7     8
     9    10    11

>>
```

Secara prinsip, dapat digunakan operator seperti +, -, dan / untuk melakukan operasi skalar terhadap larik. Khusus untuk melakukan pemangkatan setiap elemen dengan suatu nilai, perlu menggunakan .^.

```
>> X = [ 1 2 3 4 ]

X =

     1     2     3     4

>> Y = X.^2

Y =

     1     4     9    16

>>
```

D. Operasi Matematika Antarlarik

Bila dua buah matriks memiliki dimensi yang sama (jumlah baris sama dan jumlah kolom sama), operasi matematika antarlarik bisa dikenakan. Sebagai contoh, terdapat dua buah matriks seperti berikut:

$$A = \begin{bmatrix} a_1 & a_1 & \cdots & a_{1n} \\ a_2 & a_2 & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_m \end{bmatrix} \quad B = \begin{bmatrix} b_1 & b_1 & \cdots & b_{1n} \\ b_2 & b_2 & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_m \end{bmatrix}$$

Kalau dikehendaki untuk membentuk matriks C yang diperoleh dengan melakukanoperasi seperti berikut:

$$\begin{bmatrix} a_1 + b_1 & a_1 + b_1 & \dots & a_{1n} + b_{1n} \\ a_2 + b_2 & a_2 + b_2 & \dots & a_{2n} + b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_m + b_m \end{bmatrix}$$

Dapat dituliskan perintah:C = A + B

Contoh:

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

A =

     1     2
     3     4

>> B = [2 1; 6 5]

B =

     2     1
     6     5

>> C = A + B

C =

     3     3
     9     9
```

Jika dikehendaki untuk melakukan operasi pengurangan seperti berikut:

$$\begin{bmatrix} a_1 - b_1 & a_1 - b_1 & \cdots & a_{1n} - b_{1n} \\ a_2 - b_2 & a_2 - b_2 & \cdots & a_{2n} - b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & \cdots & a_m - b_m \end{bmatrix}$$

Dapat digunakan perintah:

C = A - B

Dengan menggunakan larik A dan B di depan, diperoleh:

```
>> C = A - B

C =

     -1      1
     -3     -1

>>
```

Untuk melakukan operasi perkalian antarelemen matriks seperti berikut

$$\begin{bmatrix} a_1 b_1 & a_1 b_1 & \cdots & a_{1n} b_{1n} \\ a_2 b_2 & a_2 b_2 & \cdots & a_{2n} b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} b_{m1} & a_{m2} b_{m2} & \cdots & a_m b_m \end{bmatrix}$$

Dapat dituliskan perintah:

C = A . * B

Perhatikan bahwa, di depan * ada tanda titik.

Contoh:

```
>> C = A.*B

C =

     2     2
    18    20

>>
```

Operasi seperti

A*B

memiliki makna yang berbeda.

Untuk melakukan operasi pembagian antarelemen matriks seperti berikut

$$\begin{bmatrix} a_1 : b_1 & a_1 : b_1 & \cdots & a_{1n} : b_{1n} \\ a_2 : b_2 & a_2 : b_2 & \cdots & a_{2n} : b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} : b_{m1} & a_{m2} : b_{m2} & \cdots & a_m : b_m \end{bmatrix}$$

Dapat dituliskan perintah:

C = A./B

atau C = B.\A

```
>> C = B.\A

C =

    0.5000    2.0000
    0.5000    0.8000

>>
```

```
>> C = A./B

C =

    0.5000    2.0000
    0.5000    0.8000

>>
```

Untuk melakukan operasi pemangkatan antarelemen matriks seperti berikut

$$\begin{bmatrix} a_1 \wedge b_1 & a_1 \wedge b_1 & \cdots & a_{1n} \wedge b_{1n} \\ a_2 \wedge b_2 & a_2 \wedge b_2 & \cdots & a_{2n} \wedge b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} \wedge b_{m1} & a_{m2} \wedge b_{m2} & \cdots & a_m \wedge b_m \end{bmatrix}$$

Dapat dituliskan perintah: C = A.^BContoh:

```
>> C = A.^B

C =

     1     2
   729  1024
```

Mengakses Larik

Untuk keperluan mengakses (mengambil atau mengubah) elemen larik, dapat digunakannotasi:

1) A(i) untuk vektor
2) A(i, j) untuk matriks

dengan A adalah nama larik dan i dan j menyatakan indeks. Dalam hal ini, baik i dan jdimulai dari 1.

Jika V adalah vektor, maka

- V(1) menyatakan elemen pertama dalam vektor
- V(2) menyatakan elemen kedua dalam vektor, dan seterusnya.

```
>> V = [ 5 6 7 8 ]

V =

     5     6     7     8

>> V(1)

ans =

     5

>> V(2)

ans =

     6

>> V(4)

ans =

     8

>>
```

Perlu diketahui, sekiranya indeks melebihi jumlah elemen, pesan kesalahan akan ditampilkan sebagaimana terlihat pada contoh berikut:

```
>> V(5)
??? Attempted to access V(5); index out of bounds because
numel(V)=4.
```

Contoh berikut menunjukkan pembentukan matriks A dan cara menampilkan elemenpada baris 2 kolom 1.

```
>> A = [6, 7; 8, 9]

A =

     6     7
     8     9

>> disp(A(2,1))
     8

>>
```

Penampilan isi sebuah elemen larik melalui disp

Adapun contoh berikut digunakan untuk mengubah elemen pada baris 2 kolom 1dengan nilai 4

```
>> A = [6, 7; 8, 9]

A =

     6     7
     8     9

>> A(2,1) = 4;
>> disp(A)
     6     7
     4     9

>>
```

Contoh pembesaran ukuran larik:

```
>> A = [6, 7; 8, 9]

A =

     6     7
     8     9

>> A(2,4) = 4

A =

     6     7     0     0
     8     9     0     4
```

Tampak bahwa larik A tidak lagi berukuran 2x2 melainkan menjadi 2x4.

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

A =

      1      2      3
      4      5      6
      7      8      9

>> B = A(2:3, 1:2)

B =

      4      5
      7      8
```

Dengan cara serupa, perintah

C = A(1:1, 1:3)

Menghasilkan

[1 2 3]

Perintah seperti

C = A(1:1, 1:3)

Pada contoh tersebut bisa disederhanakan menjadi:

C = A(1, 1:3)

Atau bahkan

C = A(1, :)

Notasi : yang tidak disertai angka apapun menyatakan dari indeks yang pertama hingga yang terakhir.

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

A =

     1     2     3
     4     5     6
     7     8     9

>> C = A(:,1)

C =

     1
     4
     7
```

Pada contoh tersebut, mengingat tanda : diletakkan pada indeks baris maka berarti dari baris pertama hingga baris terakhir. Dengan kata lain, A(:,1) berarti mengambil hanya kolom pertama pada larik A (untuk semua baris).

MATLAB juga memungkinkan penukaran kolom atau baris. Contoh:

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

A =

     1     2     3
     4     5     6
     7     8     9

>> B = A(3:-1:1, :)

B =

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

Notasi 3:-1:1 pada indeks baris menyatakan bahwa baris yang diperoleh adalah baris 3,2,1. Perlu diketahui, notasi 3:-1:1 berarti:

- Dimulai dari 3,
- Hingga 1,
- Dengan penurunan sebesar 1 (-1 berarti penurunan sebesar

1).

Notasi : pada indeks kolom pada contoh di depan menyatakan bahwa semua kolom disertakan dengan ukuran kolom tidak berubah (1 sampai dengan 3).

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

A =

     1     2     3
     4     5     6
     7     8     9

>> B = [ 12, 13; 14, 15; 16, 17]

B =

    12    13
    14    15
    16    17

>> C = [ A B(:,2)]

C =

     1     2     3    13
     4     5     6    15
     7     8     9    17
```

Contoh berikut menunjukkan cara mengubah elemen dalam matriks menjadi sebuah vektor

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

A =

     1     2     3
     4     5     6
     7     8     9

>> B = A(:)

B =

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

Memperkecil ukuran larik, contoh:

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

A =

     1     2     3
     4     5     6
     7     8     9

>> A(:,3) = []

A =

     1     2
     4     5
     7     8
```

Notasi A(:, 3) berarti semua kolom 3, dengan mengisi [] ke notasi seperti itu akan diperoleh efek berupa penghapusan terhadap kolom tersebut.

Catatan:

[] biasa disebut sebagai matriks kosong, yaitu matriks yang tidak memiliki satupunelemen. Bentuk penugasan sebagai berikut:

M = []

Juga diperkenankan, yang menyatakan bahwa M adalah matriks kosong. Contoh: Cara membentuk matriks didasarkan pada pengulangan isi suatu vektor.

```
>> V = [ 5 6 7 ];
>> X = V([1 1 1 1], :)

X =

     5     6     7
     5     6     7
     5     6     7
     5     6     7

>>
```

Tabel [1 1 1 1] menyatakan ada empat baris, dengan masing-masing baris diambilkan dari baris 1 pada V dan untuk semua kolom (dinyatakan dengan :).

Contoh: cara mengganti suatu baris dalam suatu matriks dengan isi suatu vektor.

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

A =

     1     2     3
     4     5     6
     7     8     9

>> V = [ 6, 3, 0 ]

V =

     6     3     0

>> A(2, :) = V

A =

     1     2     3
     6     3     0
     7     8     9
```

Pada contoh tersebut, A(2, :) berarti baris 2 untuk semua kolom. Itulah sebabnyaA(2, :) = V

Berarti isi baris 2 pada matriks A diganti dengan isi vektor baris V.

Contoh: mengubah sejumlah elemen dalam suatu matriks dengan suatu nilai yang sama

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

A =

     1     2     3
     4     5     6
     7     8     9

>> A(1:2, 2:3) = 5

A =

     1     5     5
     4     5     5
     7     8     9

>>
```

E. Memperoleh Ukuran Larik

Ukuran suatu larik ditentukan oleh jumlah baris dan kolom. Misalnya, matriks

$$\begin{bmatrix} 1 & & \\ 5 & & \\ 9 & & \end{bmatrix} \quad \begin{bmatrix} 2 & 3 & 4 \\ 6 & 7 & 8 \\ 0 & 1 & 2 \end{bmatrix}$$

$$\begin{vmatrix} 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 0 \end{vmatrix}$$

Memiliki ukuran 5x4 Sedangkan vektor kolom

$$\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

Memiliki ukuran 5x1.

Ukuran suatu larik dapat diketahui dengan menggunakan fungsi **size.**

```
>> A = [1, 2; 3, 4; 5, 6];
>> size(A)

ans =

      3      2

>>
```

Hasil **size** berupa vektor baris yang berisi 2 elemen. Elemen pertama berupa jumlahbaris dan elemen kedua berupa jumlah kolom.

```
>> V = [ 1 2 3 4 5];
>> [baris, kolom] = size(V)

baris =

     1


kolom =

     5

>>
```

Pada contoh sebelumnya, variabel baris diisi dengan elemen pertama yang dihasilkan **size**, yaitu menyatakan jumlah baris dan variabel kolom berisi jumlah kolom.

Fungsi **size** juga memperkenankan untuk mendapatkan jumlah baris atau jumlah kolomsaja pada suatu larik.

```
>> V = [ 1 2 3 4 5];
>> baris = size(V, 1)

baris =

      1

>> kolom = size(V, 2)

kolom =

      5

>>
```

Dengan menambahkan argumen kedua berupa angka 1, size akan menghasilkan jumlah baris, sedangkan nilai 2 pada argumen kedua memberikan hasil berupa jumlah kolom.

Terkait dengan larik, terdapat fungsi length, yang kegunaannya adalah menghasilkan jumlah elemen dalam suatu vektor.

```
>> V = [ 1 2 3 4 5];
>> length(V)

ans =

     5
```

Jika fungsi **length** dikenakan pada matriks berukuran m x n, hasilnya berupa nilaiterbesar di antara m dan n.

```
>> A = [1, 2; 3, 4; 5, 6];
>> size(A)

ans =

     3     2

>> length(A)

ans =

     3
```

Misalnya, dihadapkan pada persoalan untuk menampilkan tabel seperti berikut:

1	**1**
2	4
3	9
4	16
...	
14	196
15	255

Apa yang terpikirkan untuk menyelesaikannya ? Barangkali anda akan menuliskan perintah seperti berikut:

```
>> for bil = 1:15
fprintf('%4d %4d\n', bil, bil * bil)
end
```

Namun, persoalan semacam itu tidak harus dipecahkan dengan menggunakan **for** atau **while**. Alternatif yang lain adalah dengan menggunakan larik.

```
% Berkas: dglarik.m

M1 = [1:15]';
M2 = M1 .^2;
M = [M1 M2];
disp(M)
```

```
>> dglarik
     1     1
     2     4
     3     9
     4    16
     5    25
     6    36
     7    49
     8    64
     9    81
    10   100
    11   121
    12   144
    13   169
    14   196
    15   225
```

Penjelasan terhadap kode pada berkas-M dglarik.m:

- Pernyataan

  M1 = [1:15]';

  Digunakan untuk membentuk vektor kolom yang berisi angka 1 sampai dengan 15. operator ' digunakan untuk melakukan transpos dari vektor baris [1:15] menjadi vektor kolom.

- Pernyataan

  M2 = M1 .^2;

  digunakan untuk membentuk vektor kolom yang berukuran sama dengan vektor M1 dan nilainya adalah kuadrat dari masing-masing elemen dalam M1.

- Pernyataan

  M = [M1 M2];

  digunakan untuk membentuk matriks M yang merupakan gabungan antara vektor baris M1 dan M2.

Pada matriks (dan juga vektor), juga dapat menggunakan operator relasional, seperti berikut:

```
>> A = [ 1, 2, 3; 4, 1, 5]

A =

     1     2     3
     4     1     5

>> A > 1

ans =

     0     1     1
     1     0     1
```

A > 1

Berarti membandingkan masing-masing elemen dalam larik apakah bernilai lebih besar daripada 1. Hasilnya berupa larik dengan ukuran yang sama dengan larik A dengan masing-masing elemen bernilai 0 yang berarti salah atau 1 yang berarti benar.

Matriks Nol dan Matriks Satu

Untuk membuat matriks yang semua elemennya bernilai 0, kita bisa menggunakan **zeros**, sedangkan kalau semua elemennya bernilai 1, maka gunakanlah **ones**. Baik pada **zeros** maupun **ones**, argumen pertama menyatakan jumlah baris dan argumen kedua menyatakan jumlah kolom.

```
>> zeros(3, 2)

ans =

     0     0
     0     0
     0     0

>> ones(4, 3)

ans =

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

Matriks Identitas

Matriks identitas adalah matriks bujur sangkar yang salah satu diagonalnya bernilai 1 dan yang lain bernilai 0. Untuk menciptakan sebuah matriks identitas berukuran n x n, kita bisa menggunakan pernyataan:

eye(n)

```
>> eye(5)

ans =

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

F. Menghitung Determinan

Determinan sebuah matriks berupa sebuah bilangan. Sebagai contoh, terdapat matriks berukuran 2 x 2 dengan elemen-elemen seperti berikut:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Determinan matriks tersebut berupa: det $= a_{11}a_{22} - a_{12}a_{21}$

Untuk matriks berukuran lebih dari 2x2, determinannya tidak sesederhana yangberukuran 2x2. Namun, dengan menggunakan MATLAB, ukuran berapapun tidak menjadi masalah. Determinan bisa diperoleh dengan mudah dengan menggunakan **det**. Contoh menghitung determinan:

```
>> A=[7 6; 8 1]

A =

     7     6
     8     1

>> det(A)

ans =

   -41
```

Catatan:

- Determinan yang bernilai nol mengisyaratkan bahwa matriks bersifat *singular*, yaitu matriks yang tidak mempunyai matriks inversi.

G. Kegunaan Determinan

Determinan berguna untuk memperoleh solusi pada semacam persoalan berikut:

$x + 3y + z = 4$

$2x - 6y - 3z = 10$

$4x - 9y + 3z = 4$

Salah satu solusi untuk memecahkan persoalan berbentuk umum seperti berikut

$$a_1x + b_1y + c_1z = d_1 a_2x + b_2y + c_2z = d_2 a_3x + b_3y + c_3z = d_3$$

Yaitu dilakukan dengan menggunakan aturan Cramer. Solusinya seperti berikut:

$$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\Delta}, \; y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\Delta}, \quad z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\Delta}$$

Dalam hal ini, adalah determinan koefisien-koefisien pada x, y, dan z:

$$\Delta = \begin{vmatrix} a_1 & b & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

Dan $|\,|$ menyatakan determinan matriks. Dengan demikian, nilai x, y, dan z yang memenuhi ketiga persamaan tersebut bisa dihitung dengan cara seperti berikut.

Berikut adalah cara menyelesaikannya.

```
>> A=[1 3 1; 2 -6 -3; 4 -9 3];
>> det_pembagi = det(A)

det_pembagi =

   -93

>> x = det([4 3 1; 10 -6 -3; 4 -9 3]) / det_pembagi

x =

     4

>> y = det([1 4 1; 2 10 -3; 4 4 3]) / det_pembagi

y =

    0.6667

>> z = det([1 3 4; 2 -6 10; 4 -9 4]) / det_pembagi

z =

    -2
```

Jadi, solusi untuk persamaan linear di depan berupa x=4, y=0,6667 (2/3), dan z=-2.

H. Alternatif Penyelesaian Persamaan Linear

Persamaan linear seperti

$x + 3y + z = 4$

$2x - 6y - 3z = 10$

$4x - 9y + 3z = 4$

Dapat diselesaikan dengan cara yang sederhana melalui operator \. Pertama-tama, siapkan matriks yang berisi koefisien-koefisien x, y, dan z. Sebut saja A. Kemudian, susun vektor kolom yang berisi nilai-nilai yang terletak di sebelah kanan tanda =. Sebut saja B. Maka, solusi x, y, dan z diperoleh melalui A\B

Contoh:

```
>> A=[ 1 3 1; 2 -6 -3; 4 -9 3 ]

A =

     1     3     1
     2    -6    -3
     4    -9     3

>> B=[4; 10; 4]

B =

     4
    10
     4

>> A\B

ans =

    4.0000
    0.6667
   -2.0000
```

Hasil A\B secara berturut-turut menyatakan nilai untuk x, y, dan z.

I. Mendapatkan Rank Matriks

Rank matriks adalah suatu bilangan yang menyatakan seberapa banyak baris dalam matriks yang tak tergantung secara linear terhadap baris lain. Suatu vektor dikatakan tak tergantung secara linear apabila tidak dapat ditulis sebagai independen linear terhadap yang lain. Misalnya, terdapat tiga buah vektor seperti berikut:

$X = \begin{bmatrix}1\\2\end{bmatrix}, Y = \begin{bmatrix}2\end{bmatrix}, Z = \begin{bmatrix}8\end{bmatrix}$

Ketiga vektor di atas mempunyai hubungan secara linear seperti berikut:

$2X+3Y=Z$

Untuk memahami *rank* matriks, perhatikan contoh berikut:

$$\begin{bmatrix}1 & 2 & 3 & 4\\2 & 4 & 6 & 8\\4 & 6 & 1 & 3\\0 & 2 & -7 & 1\end{bmatrix}$$

Rank matriks di atas sebesar 3 mengingat baris yang tidak tergantung baris lain secara linear hanya ada tiga buah. Contoh pemakaian **rank** dapat dilihat pada gambar dibawah ini.

```
>> A = [1 2 3 7; 2 4 6 14; 3 4 5 6]

A =

     1     2     3     7
     2     4     6    14
     3     4     5     6

>> rank(A)

ans =

     2
```

```
>> A = [1 2 3 7; 2 4 6 14; 4 8 12 28]

A =

     1     2     3     7
     2     4     6    14
     4     8    12    28

>> rank(A)

ans =

     1
```

Contoh penggunaan rank

Catatan:

Rank matriks dapat digunakan untuk mengetahui apakah suatu persamaan linear mempunyai solusi atau tidak. Sebagai contoh, terdapat m persamaan linear dengan n variabel yang belum diketahui dan dinotasikan sebagai Ax = b

Matriks gabungan dibentuk dengan menggabungkan vektor b dan matriks A, seperti berikut: [A b]

Nah, persamaan-persamaan linear tersebut mempunyai penyelesaian hanya jika *rank* A dan *rank*[A b] bernilai sama.

J. Matriks Inversi

Matriks inversi dari matriks A biasa ditulis dengan notasi A^{-1} akan diperoleh matriks identitas (I). Jadi,

$AA^{-1}=I$

Matriks inversi dari matriks A diperoleh dengan menggunakan fungsi **inv**. Pemanggilannya seperti berikut:

```
B = inv(A);
```

Pada contoh di atas, B adalah matriks inversi dari matriks A. Contoh:

```
>> A = [1 3 1; 2 -6 -3; 4 -9 3]

A =

     1     3     1
     2    -6    -3
     4    -9     3

>> B=inv(A)

B =

    0.4839    0.1935    0.0323
    0.1935    0.0108   -0.0538
   -0.0645   -0.2258    0.1290

>> B*A

ans =

    1.0000         0    0.0000
         0    1.0000         0
         0         0    1.0000
```

Perhatikan bahwa hasil perkalian A dan B berupa matriks identitas (semua elemen dalam salah satu diagonalnya bernilai 1).

K. Matriks-Matriks Khusus

MATLAB menyediakan beberapa fungsi yang berguna untuk membentuk matriks dengan sifat khusus. Selain **ones, zeros,** dan **eye** yang telah dibahas, terdapat pula yang lain.

Tabel fungsi-fungsi penghasil matriks dengan sifat khusus

Fungsi	Keterangan
magic(n)	Menghasilkan matriks berukuran n x n dengan elemen bernilai 1 hingga n^2 dengan jumlah nilai yang sama pada kolom dan baris.
diag(V)	Membentuk matriks diagonal, dengan nilai pada vektor V diletakkan dalam diagonal
triu(X)	Menghasilkan matriks yang bagian segitiga atasnya diambil dari matriks X
tril(X)	Menghasilkan matriks yang bagian segitiga bawahnya diambil dari matriks X
hilb(n)	Menghasilkan matriks Hilbert orde n. Elemen dalam matriks Hilbert dihitung melalui H(i, j) = 1/(i+j-1)

Contoh **magic** untuk mendapatkan matriks dengan ukuran 5 x 5:

```
>> magic(5)

ans =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

Matriks dengan jumlah nilai-nilai kolom dan baris sama

Contoh penggunaan **diag**

```
>> V = [5 6 7 4];
>> diag(V)

ans =

     5     0     0     0
     0     6     0     0
     0     0     7     0
     0     0     0     4

>> diag(V, 1)

ans =

     0     5     0     0     0
     0     0     6     0     0
     0     0     0     7     0
     0     0     0     0     4
     0     0     0     0     0
```

```
>> diag(V, -1)

ans =

     0     0     0     0     0
     5     0     0     0     0
     0     6     0     0     0
     0     0     7     0     0
     0     0     0     4     0
```

Contoh penggunaan **triu** dan **tril**

```
>> X=magic(5)

X =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> triu(X)

ans =

    17    24     1     8    15
     0     5     7    14    16
     0     0    13    20    22
     0     0     0    21     3
     0     0     0     0     9
```

```
>> tril(X)

ans =

    17     0     0     0     0
    23     5     0     0     0
     4     6    13     0     0
    10    12    19    21     0
    11    18    25     2     9
```

Matriks penggunaan **hilb**

```
>> hilb(4)

ans =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

L. Manipulasi Matriks

MATLAB menyediakan beberapa fungsi yang berguna untuk melakukan pemanipulasian matriks, misalnya untuk memutar posisi elemen-elemen yang terdapat matriks dan menukarkan posisi elemen-elemen dalam matriks melalui pencerminan secara vertikal ataupun horizontal.

Tabel fungsi-fungsi untuk memanipulasi susunan elemen dalam matriks

Fungsi	Keterangan
rot90	Memutar matriks sebesar 90^o berlawanan dengan arah jarum jam
fliplr	Mencerminkan matriks secara vertikal sehingga elemen kiri akan berada di elemen kanan dan begitu juga sebaliknya
flipud	Mencerminkan matriks secara horizontal sehingga elemen atas akan berada di elemen bawah dan begitu juga sebaliknya
reshape	Mengubah dimensi matriks
repmat	Berguna untuk membuat matriks yang merupakan pengulangan elemen-elemen dalam suatu matriks

Bentuk fungsi **rot90** yang pertama berbentuk semacam berikut: rot90(M)

Fungsi akan mengembalikan matriks dengan elemen-elemen berdasarkan matriks M yang telah diputar sebesar 90^o berlawanan arah jarum jam. Contoh:

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

M =

     1     2     3
     4     5     6
     7     8     9

>> rot90(M)

ans =

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

Fungsi **rot90** mendukung penyertaan argumen kedua. Argumen kedua menentukanjumlah pemutaran. Contoh:

```
>> M = [1 2 3; 4 5 6; 7 8 9];
>> rot90(M, 2)

ans =

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

Contoh pemutaran sebesar 180 derajat

Fungsi **fliplr** melakukan pencerminan secara vertikal. Contoh:

```
>> M = [1 2 3; 4 5 6; 7 8 9];
>> fliplr(M)

ans =

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

Contoh pencerminan secara vertikal

Fungsi **flipud** melakukan pencerminan secara horizontal. Contoh:

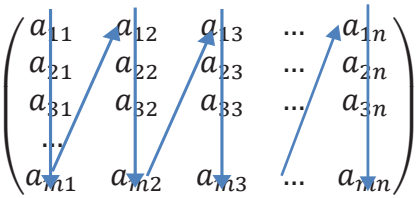
```
>> flipud(M)

ans =

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

Contoh pencerminan secara horizontal

Fungsi **reshape** berguna untuk mengubah dimensi matriks. Hal yang terpenting dalam melakukan pengubahan dimensi, total elemen matriks semula dan matriks yang dihasilkan harus sama. Bentuk pemakaian:

reshape(M, nb, nk)

Dalam hal ini, matriks yang dihasilkan berdasarkan elemen-elemen dalam matriks M akan berukuran nb x nk, dengan nb menyatakan jumlah baris dan nk menyatakan jumlah jumlah kolom. Konsep penyusunan elemen-elemen dalam matriks menggunakan pola seperti berikut.



Mekanisme penyusunan ulang matriks

Contoh penggunaan **reshape**:

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

M =

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

>> reshape(M, 2, 6)

ans =

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

>> reshape(M, 3, 4)

ans =

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

Fungsi **repmat** sangat berguna untuk membentuk matriks baru yang merupakanpengulangan dari suatu matriks. Contoh

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

M =

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

>> repmat(M,2,3)

ans =

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

M. Bilangan Random

MATLAB menyediakan fungsi bernama **rand** yang berguna untuk mendapatkan bilangan acak antara 0 dan 1. Contoh berikut menunjukkan penggunaan **rand** untuk menghasilkan sebuah bilangan acak:

```
>> rand

ans =

    0.8147
```

Contoh rand untuk menghasilkan sebuah bilangan acak

Untuk mendapatkan sebuah vektor yang berisi sejumlah bilangan acak, bisa menggunakan pemanggilan dengan bentuk rand(1, n). Dalam hal ini, akan terbentuk vektor baris dengan n kolom. Contoh:

```
>> rand(1,5)

ans =

    0.9058    0.1270    0.9134    0.6324    0.0975
```

Contoh membangkitkan sejumlah nilai acak

Untuk membuat matriks berukuran n x n, bisa menggunakan pemanggilan dengan bentuk rand(n). Namun, jika menghendaki matriks berukuran m x n, gunakan rand(m, n). Contoh:

```
>> rand(4)

ans =

    0.2785    0.1576    0.8003    0.7922
    0.5469    0.9706    0.1419    0.9595
    0.9575    0.9572    0.4218    0.6557
    0.9649    0.4854    0.9157    0.0357

>> rand(3,2)

ans =

    0.8491    0.7577
    0.9340    0.7431
    0.6787    0.3922
```

Contoh pembentukan matriks dengan elemen-elemen berupa bilangan acak

Fungsi rand sesungguhnya menghasilkan nilai acak yang bersifat seragam (uniform). Distribusin*ya merata pada semua nilai antara 0 dan 1. Untuk mendapatkan bilangan acak yang terdistribusi pada jangkauan [a, b] kita bisa menuliskan perintah seperti berikut:

```
r = a + (b-a).*rand(100, 1);
```

BAB V
BEKERJA DENGAN GRAFIK

TUJUAN PEMBELAJARAN:

Setelah mengikuti mata kuliah ini, mahasiswa mampu:

1. Membuat grafik fungsi 2 dimensi menggunakan perintah plot
2. Membuat berbagai grafik fungsi dalam satu jendela gambar
3. Membuat grafik fungsi menggunakan subplot

Materi:

A. Pembuatan Plot

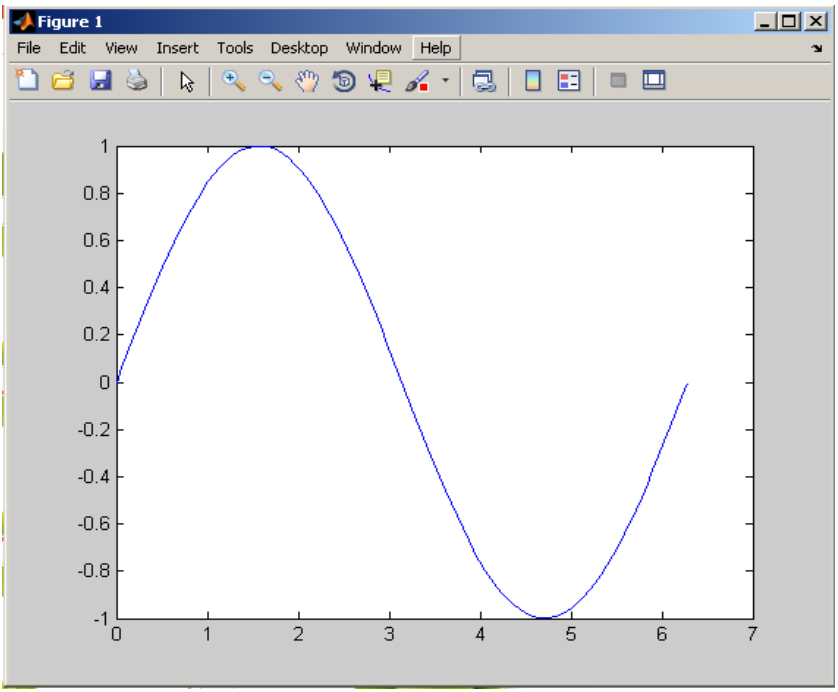
Menyajikan grafik dua dimensi yang menyatakan hubungan antara nilai dalam sumbu x dan sumbu y dapat dilaksanakan dengan mudah dengan menggunakan fungsi bernama plot.

Contoh penggunaan plot yaitu untuk menggambarkan fungsi sinus.

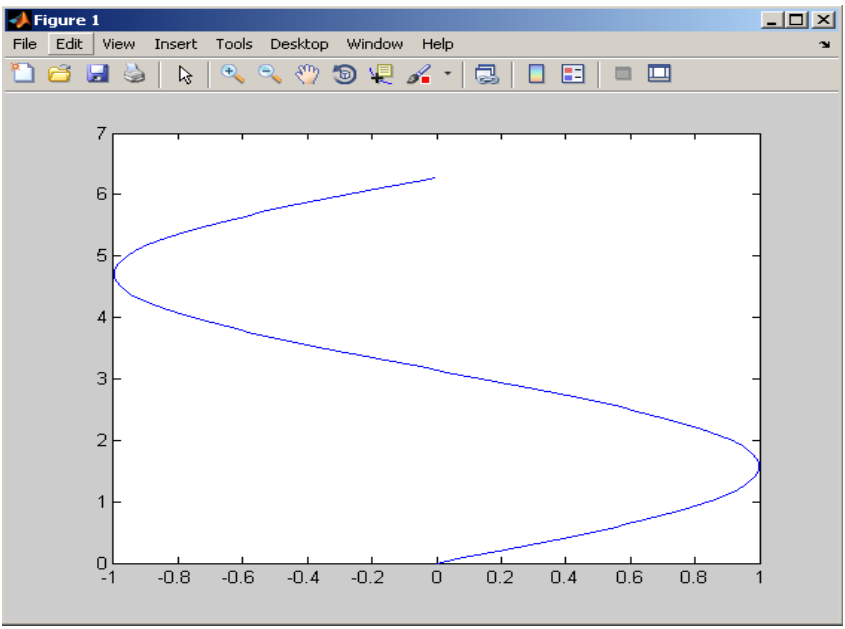
```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> plot(X,Y);
```

Pada contoh di atas, X berisi 100 nilai antara 0 sampai dengan 2*pi. Adapun Y berisi100 nilai sinus yang didasarkan nilai pada vektor X.

Contoh penggunaan plot yaitu untuk menggambarkan fungsi sinus.



Dapat juga mencoba untuk menukarkan posisi X dan Y pada plot. Contoh: plot(Y, X)



B. Menambahkan Label pada Sumbu dan Judul

Tiga buah fungsi yang berguna untuk memberikan judul untuk grafik ataupun judul pada sumbu X dan Y yaitu xlabel, ylabel, dan title.

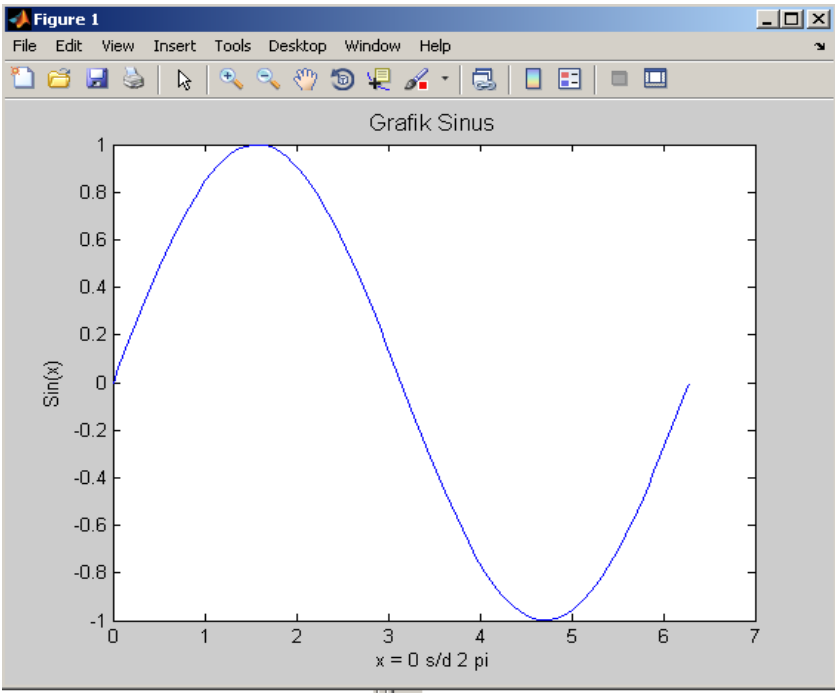
- xlabel (judul): menambahkan judul pada sumbu X
- ylabel (judul): menambahkan judul pada sumbu Y
- Title (judul): menambahkan judul grafik

Contoh:

```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> plot(X,Y);
>> xlabel('x = 0 s/d 2 pi');
>> ylabel('Sin(x)');
>> title('Grafik Sinus', 'Fontsize', 12);
```

Argumen pada fungsi title bisa hanya berisi judul pada grafik. Namun, juga bisa menyertakan tiga argumen, dengan argumen kedua menyatakan properti grafik yangakan diatur dan argumen ketiga menyatakan nilai pengaturan properti yang tersebut dalam argumen kedua.

Contoh:

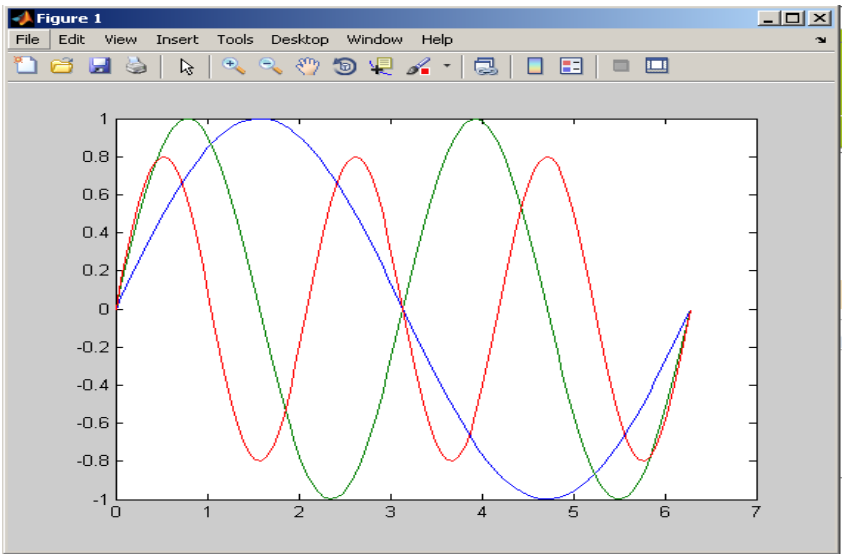


C. Menyajikan Beberapa Himpunan Data

Sebuah grafik dimungkinkan untuk menampung beberapa himpunan data. Sebagaicontoh:

```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> Y2 = sin(2 * X);
>> Y3 = 0.8 * sin (3 * X);
>> plot(X, Y, X, Y2, X, Y3);
```

Hasilnya:



Beberapa perintah plot juga bisa diletakkan pada jendela yang sama dengan memanfaatkan perintah hold on. Ketika perintah hold on dijalankan, MATLAB tidak akan menghapus gambar ketika terjadi pemanggilan plot kembali. Hal ini akan terus dilakukan sampai MATLAB menjumpai perintah hold off.

Contoh:

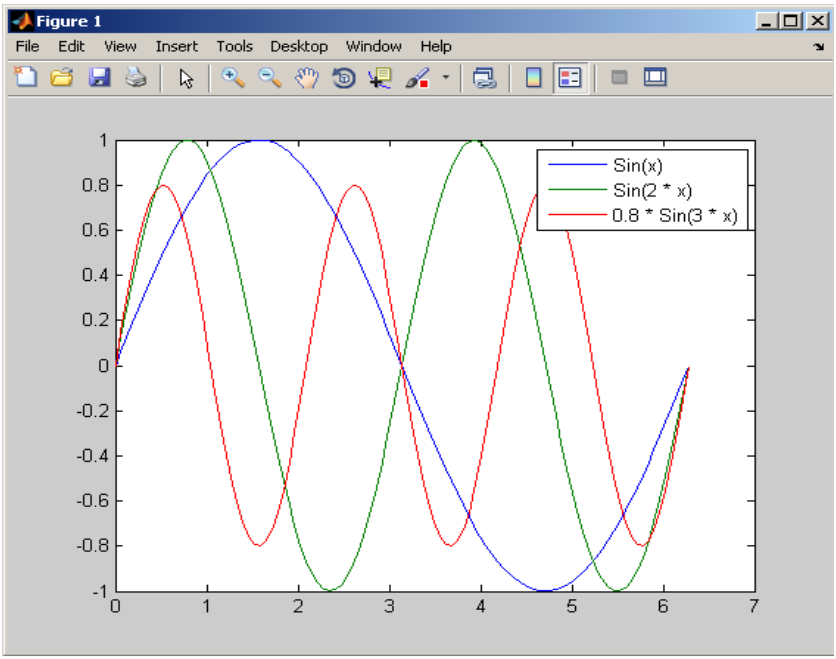
```
>> plot(X, Y)
>> hold on
>> plot(X, Y2)
>> plot(X, Y3)
>> hold off
```

Tiga jenis gambar sinus diletakkan pada satu grafik. Menambahkan Legenda

Legenda dalam grafik berfungsi sebagai penjelas. Hal itu bisa ditambahkan denganmenggunakan fungsi legend.

```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> Y2 = sin(2 * X);
>> Y3 = 0.8 * sin (3 * X);
>> plot(X, Y, X, Y2, X, Y3);
>> legend('Sin(x)', 'Sin(2 * x)', '0.8 * Sin(3 * x)');
```

Hasilnya:



Terkait dengan legend, terdapat perintah berupa legend on dan legend off.

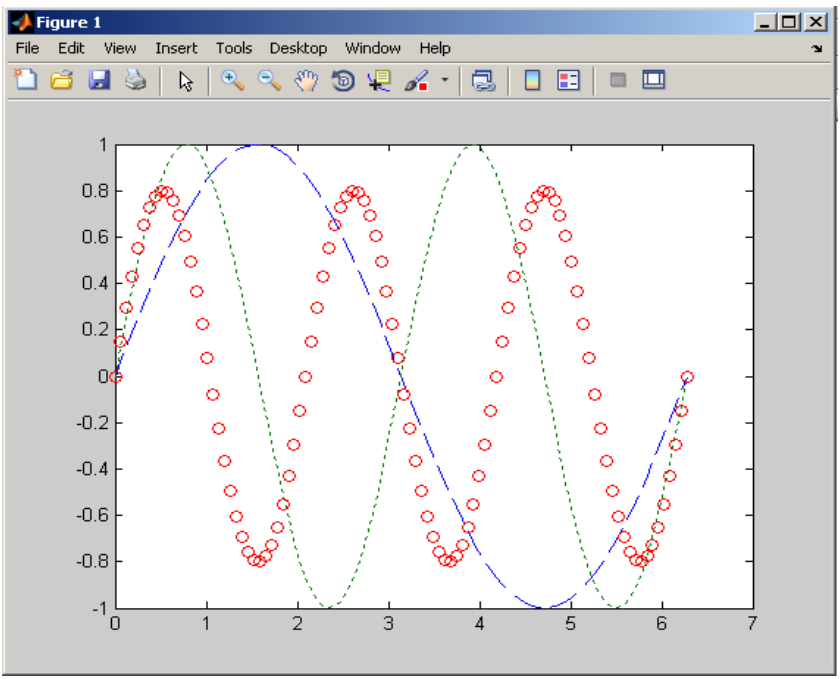
- legend off berguna untuk menyembunyikan legenda.
- legend on berguna untuk menampilkan legenda

Menentukan Jenis Garis, Jenis Penanda, dan Warna

Pada contoh sebelumnya, garis yang ditampilkan selalu utuh. Sesungguhnya, juga bisa mengatur agar garis yang digunakan terputus-putus dengan menggunakan komponen penyusun berupa titik, garis, ataupun yang lain.

```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> Y2 = sin(2 * X);
>> Y3 = 0.8 * sin(3 * X);
>> legend off %menyembunyikan legenda
>> plot(X, Y, '--', X, Y2, ':', X, Y3, 'o');
```

Setiap pasangan X dan Y terdapat argumen yang berfungsi sebagai penentu jenis garis yang digunakan untuk menyusun grafik.



Catatan:

Jenis garis pada plot:

‘-’ : garis utuh

‘--’ : garis terputus-putus menggunakan tanda –

‘:’ : garis terputus-putus menggunakan tanda titik dua

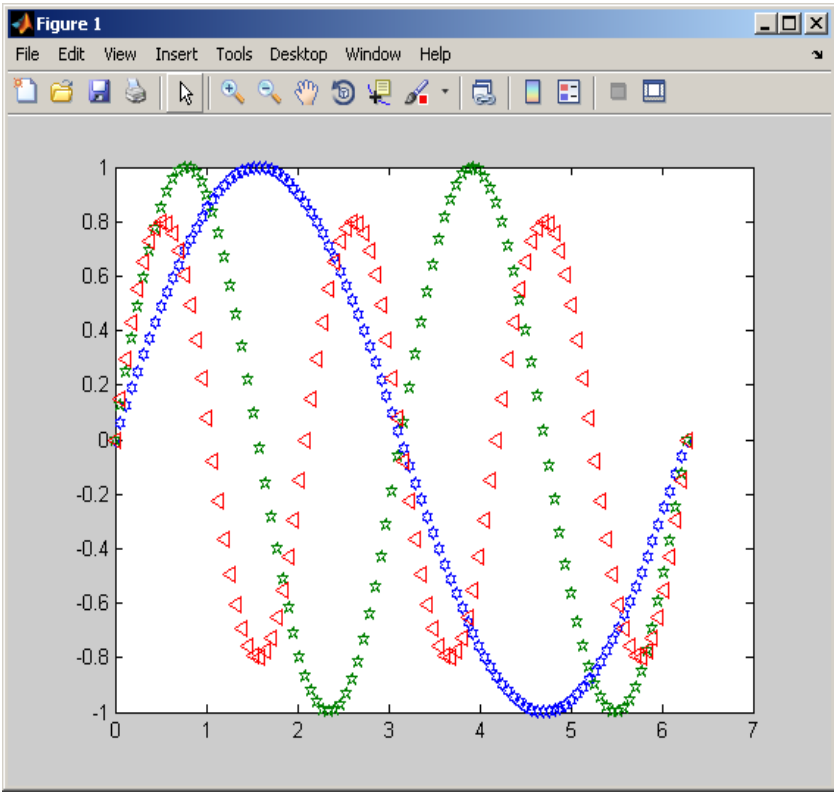
‘-.’ : garis terputus-putus menggunakan tanda – dan

Selain menggunakan garis untuk menyusun grafik, penanda seperti lingkaran, tandasilang, dan segitiga bisa disertakan.

Tabel Tanda pada grafik

Penanda Berupa	Keterangan
‘.’	Tanda titik
‘x’	Tanda x
‘*’	Tanda bintang
‘d’	Belah ketupat
‘^’	Segitiga ke atas
‘>’	Segitiga ke kanan
‘v’	Segitiga ke bawah
‘<‘	Segitiga ke kiri
‘h’	Heksagram
‘o’	Lingkaran

'+'	**Tanda +**
's'	**Bujur sangkar**
'p'	**pentagram**



Warna yang digunakan pada grafik sebenarnya secara otomatis ditentukan oleh plot. Namun, jika menghendaki untuk mengaturnya sendiri, kita bisa menambahkan kode- kode warna.

Kode Warna	Keterangan
‘y’	**Kuning**
‘c’	***Cyan* (hijau kebiru-biruan)**
‘g’	**Hijau**
‘w’	**Putih**
‘m’	**Magenta (merah kecokelat-cokelatan)**
‘r’	**Merah**
‘b’	**Biru**
‘k’	**Hitam**

Kode warna bisa dipadukan dengan jenis garis. Contoh:

Plot(X, Y, ‘y--’, X, Y2, ‘md’, X, Y3, ‘k>’);

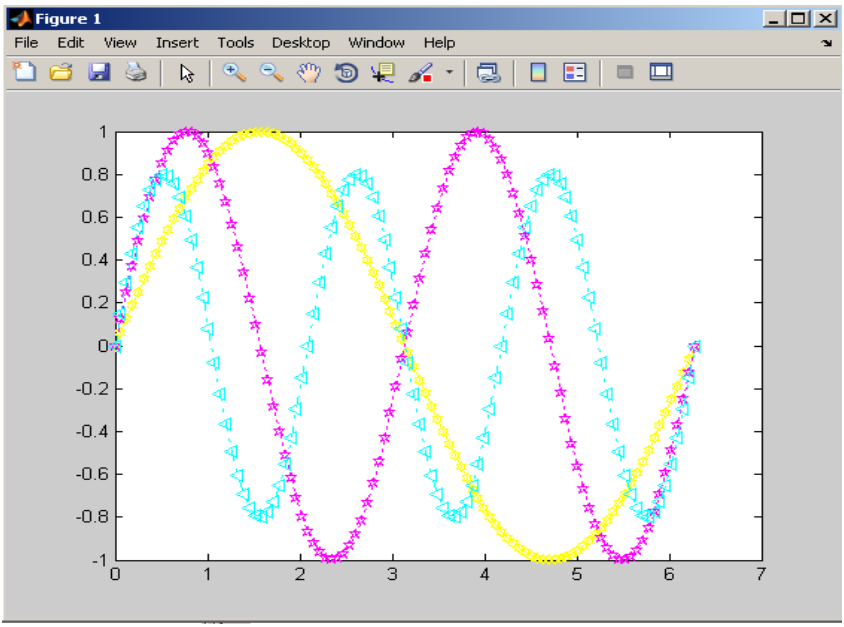
Pada contoh di atas, grafik pertama menggunakan garis terputus-putus dan berwarna kuning, grafik kedua menggunakan bentuk belah ketupat berwarna merah, dan grafik ketiga menggunakan segitiga yang menghadap ke kanan dengan warna hitam.

```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> Y2 = sin(2 * X);
>> Y3 = 0.8 * sin(3 * X);
>> plot(X, Y, 'h-y', X, Y2, 'p:m', X, Y3, '<-.c');
```

Sebagai contoh, argumen ‘h-y’ menyatakan bahwa simbol yang digunakan adalah

heksagram (h), garis utuh (-), dan warna kuning (y).

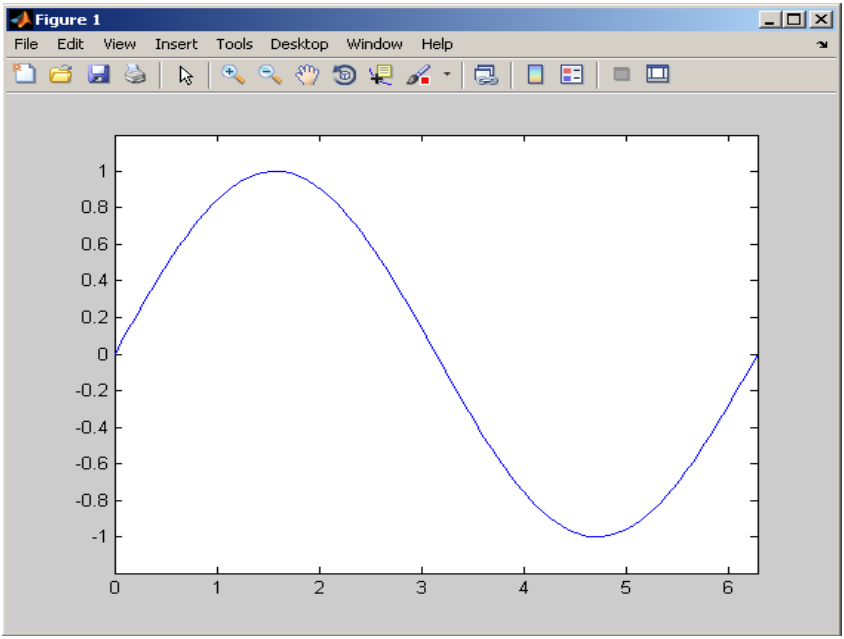
Contoh berikut menunjukkan kombinasi pemakaian jenis garis, jenis penanda, dan warna



D. Menangani Sumbu

Dalam hal ini, *xmin* dan *xmax* menyatakan nilai terkecil dan nilai terbesar dalam sumbu x, dan *ymin* dan *ymax* menyatakan nilai terkecil dan nilai terbesar dalam sumbu Y. Pada contoh di depan, mula-mula nilai terkecil dan terbesar pada X dan Y dihitung terlebih dulu. Selanjutnya, batas atas dan batas bawah sumbu Y diubah melalui: Axis([minx maxx (1.2 * miny) (1.2 * maxy)])

Dalam hal ini, 1.2 menjadi faktor yang mengatur nilai terkecil dan nilai terbesar dalamsumbu Y.



Untuk mengembalikan ke bentuk bawaan, berikan perintah:

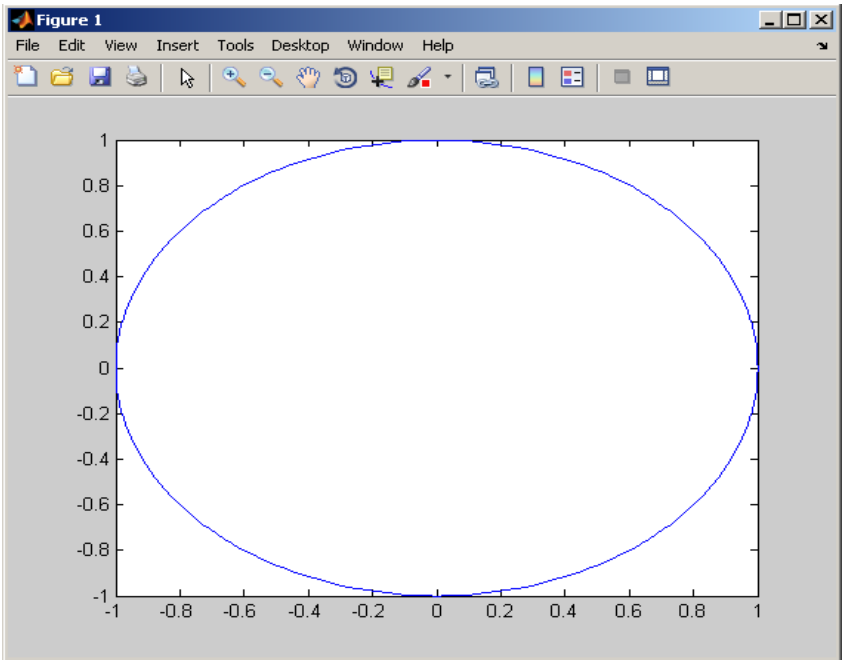
axis auto;

Perintah axis yang juga menarik untuk dibicarakan yaitu axis square. Dengan menggunakan perintah itu, panjang sumbu X dan sumbu Y dibuat sama.

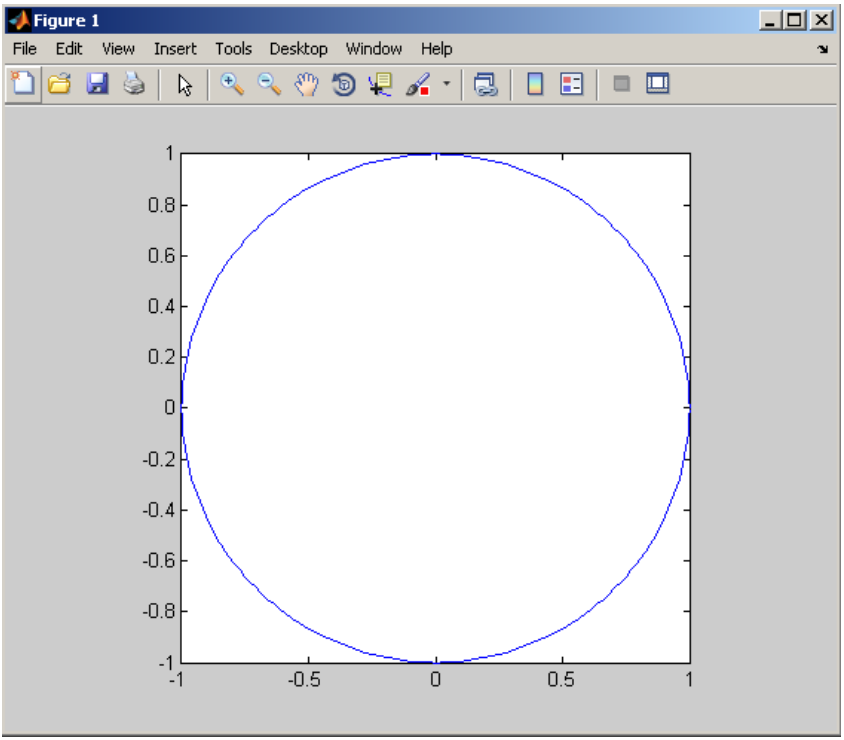
```
>> legend off %Menyembunyikan legenda
>> minx = min (X);
>> maxx = max (X);
>> miny = min (Y);
>> maxy = max (Y);
>> plot (exp(i*[0:pi/100:2*pi]));
```

```
>> plot (exp(i*[0:pi/100:2*pi]));
```

Bentuk lingkaran menggunakan mode bawaan



Bentuk lingkaran menggunakan axis square



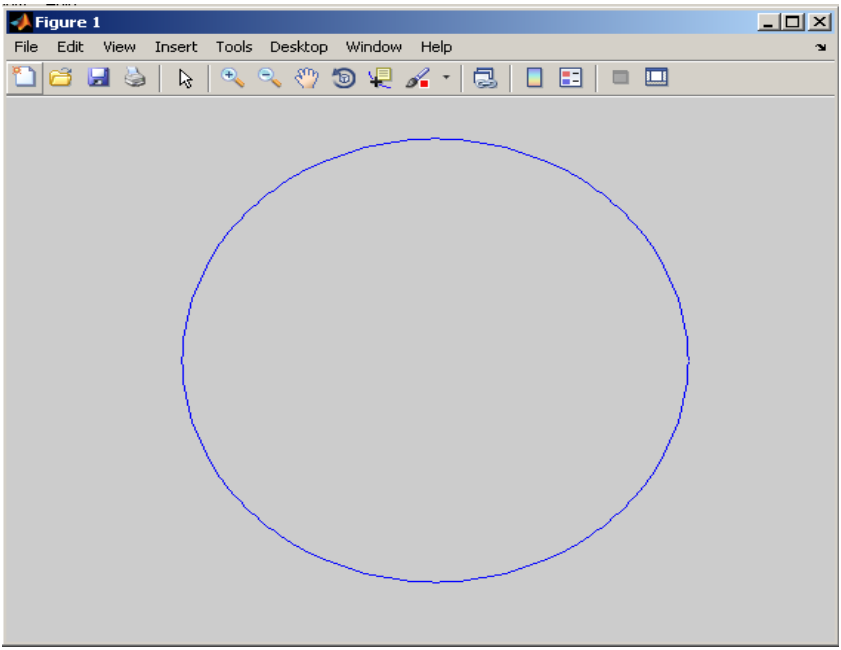
Terlihat bahwa dengan menggunakan axis square, lingkaran disajikan lebih real; bukanseperti oval.

Tabel Beberapa perintah pengaturan grafik

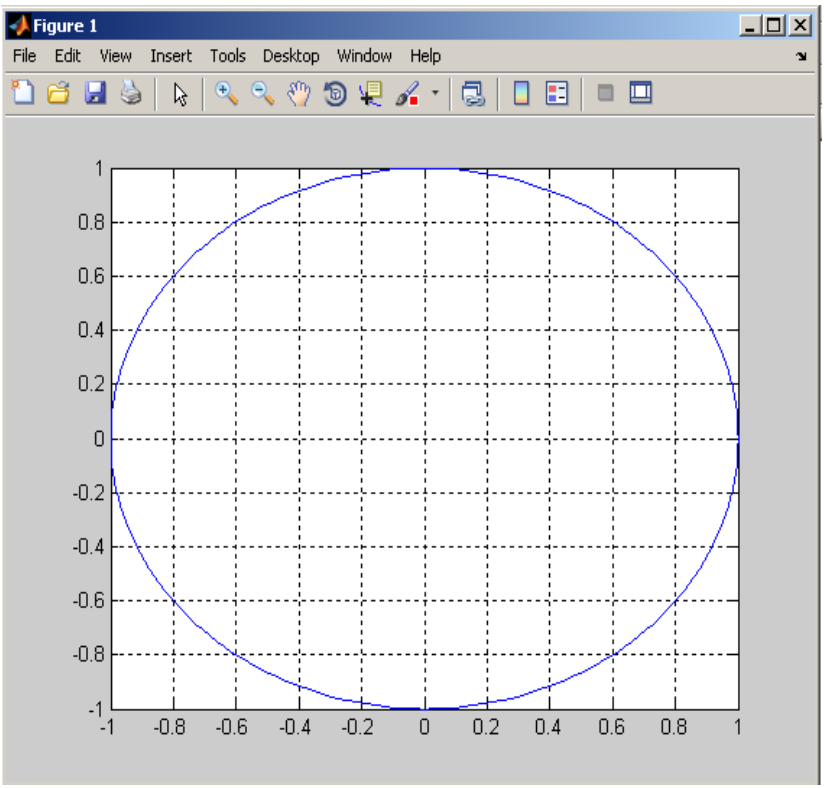
Perintah	Keterangan
axis on	**Menampilkan sumbu X dan Y**
axis off	**Menyembunyikan sumbu X dan sumbuY**

grid on	**Menampilkan kisi-kisi pada grafik**
grid off	**Menyembunyikan kisi-kisi pada grafik**

Garis sumbu disembunyikan



Gambar dengan kisi-kisi



E. Menggunakan Subplot

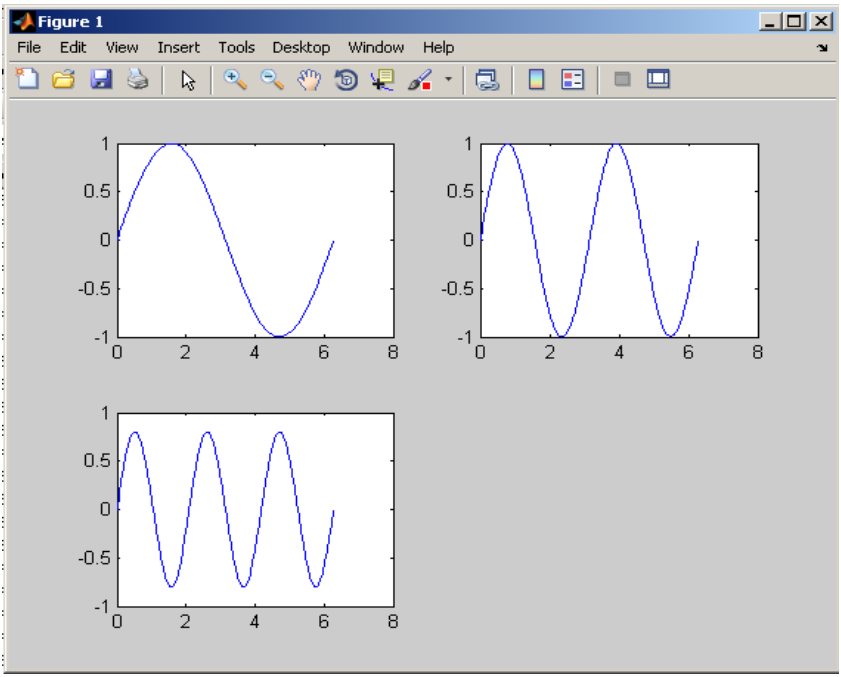
Sebuah jendela gambar bisa digunakan untuk menampung sejumlah grafik. Kuncinya adalah fungsi bernama subplot. Bentuk pemanggilannya:

subplot(m, n, p)

- Argumen pertama (m) menentukan jumlah baris gambar.
- Argumen kedua (n) menentukan jumlah kolom gambar.
- Argumen ketiga (p) menentukan posisi gambar akan diletakkan.

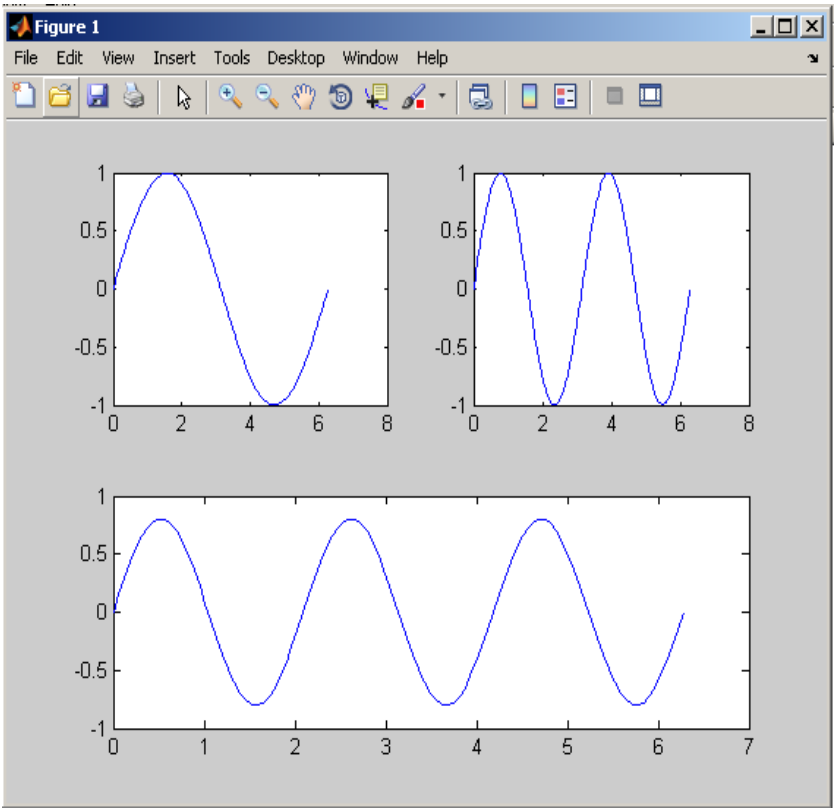
```
>> X = linspace(0, 2 * pi);
>> Y = sin(X);
>> Y2 = sin(2 * X);
>> Y3 = 0.8 * sin(3 * X);
>> subplot(2, 2, 1);
>> plot(X, Y);
>> subplot(2, 2, 2);
>> plot(X, Y2);
>> subplot(2, 2, 3);
>> plot(X, Y3);
```

Grafik dengan subplot



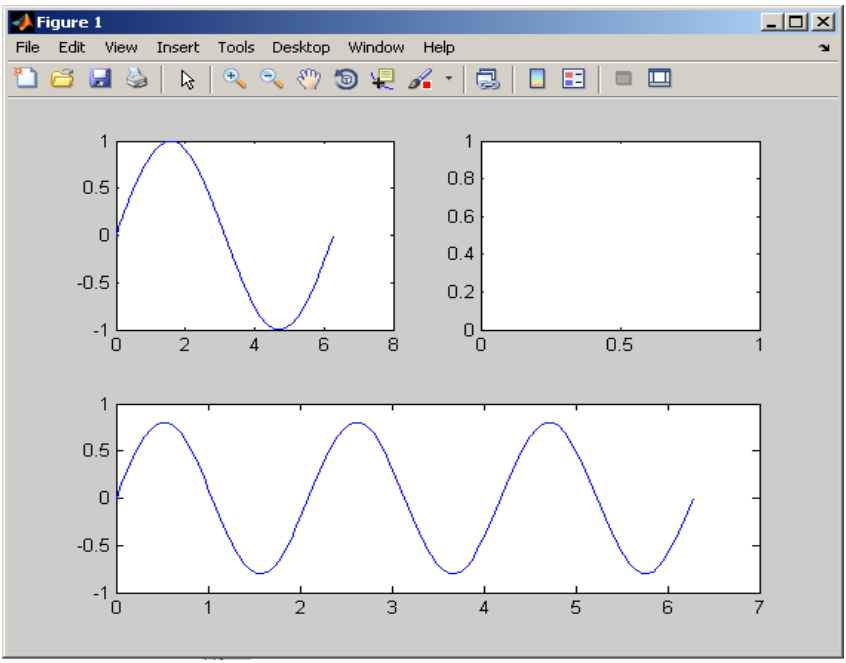
Grafik ketiga bisa diatur agar diletakkan pada gabungan posisi 3 dan 4. Perintah yangdiperlukan:

```
>> subplot(2, 2, [3 4]);
>> plot(X, Y3);
```



Kalau ingin menghapus objek gambar dalam sebuah subplot, kita bisa menggunakancla. Contoh:

```
>> subplot(2, 2, 2);
>> cla;
```



DAFTAR PUSTAKA

Cleve Moler. 2004. "the creator of MATLAB The Origins of MATLAB". http://www.mathworks.com/company/newsletters/news_notes/clevescornr/ dec04.html

Edy Sujatmiko.2007. "Pemilihan Algoritma Optimal untuk Kompresi Data Citra Iris Mata Manusia". Semarang:Universitas Diponegoro

Hermawan, Santoso.2009."Pemampatan data *lossless* dengan metode *Static-adaptive arithmetic coding.* Surakarta : Fakultas Komunikasi dan Informatika Universitas Muhammadiyah Surakarta

Iqbal, M. 2009. "Dasar Pengolahan Citra Menggunakan MATLAB". Departemen Ilmu dan Teknologi Kelautan. Bogor: Institut Pertanian

Krisnawati.2007. "Kompresi Citra RGB Dengan Metode Kuantisasi" Yogyakarta: Sekolah Tinggi Manajemen Informatika dan Komputer AMIKOM

Paul E. Black 2012. http://en.wikipedia.org/wiki/Arithmetic_coding

Petrus, Santoso. 2001. "Studi kompresi data dengan metode Aritmathic Coding". Semarang: Fakultas Teknologi Industri, Jurusan Teknik Elektro, Universitas Kristen Petra

Portable Network Graphics. 2011. http:id.wikipedia.org/wiki/Portable_Network_Graphic

Praditya, 2006. “Aplikasi Pemampatan Dan Dekonstruksi Citra Menggunakan Analisis Komponen Utama (Principal Componant Analisis)”. Semarang : Universitas Diponegoro

Richard, Goering (2004). *Matlab edges closer to electronic design automation worldEE Times*

Siregar, Debi Maulina.2011. “Analisis Dan Perancangan Algoritma Arithmetic Coding Dalam Kompresi File Audio”. Sumatra Utara : Universitas SumatraUtara

Sriwiyanto, Agus. 2012. “Implementasi Metode *Run Length Encoding* Dalam Kompresi Citra Dengan Citra Hitam Putih”. Surakarta : Universitas Muhammadiyah Surakarta

Sugiharto, Aris. 2006. *Pemrograman GUI dengan Matlab.* Semarang : Andi

Sunaryo. 2007. “Enkripsi Data Hasil Analisis Komponen Utama (PCA) Atas Citra Iris Mata Menggunakan Algoritma Md5”. Semarang : Universitas Diponegoro

Widakdo, Ari. 2012. “Implementasi Algoritma Metode *Huffman* Pada Kompresi Citra”. Surakarta : Universitas Muhammadiyah Surakarta

Widiarsono, Teguh. 2005. *Tutorial Praktis Belajar Matlab.* Jakarta

Wijaya, Adnan Rifki. 2012. “Kompresi Citra Berwarna Dengan Penerapan *Discrete Cosine Transform* (DCT)”. Surakarta : Universitas Muhammadiyah Surakarta

Yunianto, Taufik. 2012.“ Membangun Aplikasi Kompresi *Image* Menggunakan Metode DPCM **(**Differensial Pulse Code Modulation)”. Surakarta : Universitas Muhammadiyah Surakarta

TENTANG PENULIS

Muhammad Muzaini adalah doktor di bidang Pendidikan Matematika lulusan Program Pascasarjana Universitas Negeri Surabaya lulus tahun 2019. Lahir di Mulyorejo (Kab. Lutra) 08 Desember 1986. Lulus S-1 di Program Studi Pendidikan Matematika Universitas Muhammadiyah Makassar Tahun 2009 dan S-2 di Universitas Negeri Makassar Tahun 2012. Saat ini ia mengajar di Program Studi Pendidikan Matematika FKIP Universitas Muhammadiyah Makassar. Kegiatan penelitian yang dilakukannya antara lain didanai oleh Hibah DRPM Dikti. Pengabdian Masyarakat yang dilakukan antara lain berupa Pelatihan dan Workshop pada Guru-guru SD, SMP dan SMA tentang tema-tema yang berkaitan dengan pendidikan Matematika. Penulis aktif sebagai pembicara pada seminar/konferensi dan aktif sebagai penulis artikel pada jurnal Nasional dan jurnal Internasional bereputasi terindeks Scopus.

Muhammad Ikram lulus S1 di Program Pendidikan Matematika Universitas Negeri Makassar tahun 2010. Lulus S2 di Program Magister Pendidikan Matematika di Universitas Negeri Makassar tahun 2013. Lulusan S3 di Program Doktoral Pendidikan Matematika di Universitas Negeri

Malang tahun 2020. Saat ini adalah dosen di Program Studi Sarjana dan Magister Pendidikan Matematika di Universitas Cokroaminoto Palopo. Aktif dalam berbagai kegiatan penulisan ilmiah, pelatihan, dan narasumber di bidang matematika.

Nasrun Syahrir lahir di Sungguminasa Kabupaten Gowa pada tanggal 28 Juni 1981. Anak ke empat dari lima bersaudara dari pasangan Bapak Muh. Syahrir dan Ibu St. Khalidjah. Riwayat pendidikan Dasar sampai Strata 1 ditempuh di Makassar yakni SD Negeri Bontopajja (lulus tahun 1992), SMP Negeri 15 Makassar (lulus tahun 1996), SMA Yapip Makassar (lulus tahun 1999), dan S1 Universitas Muhammadiyah Makassar Program studi Pendidikan Matematika (lulus tahun 2003).

Program Magister Pendidikan (M.Pd) di Program Pascasarjana Universitas Negeri Malang program studi pendidikan matematika dasar (lulus tahun 2008). Karir sebagai tenaga pengajar mulai tahun 2006 sebagai Dosen persyarikatan di Universitas Muhammadiyah Makassar sampai sekarang. Penulis ditugaskan melanjutkan studi S3 Pendidikan Matematika Universitas Negeri Malang dan mendapatkan beasiswa unggulan selama mengikuti proses perkuliahan. Penulis telah menghasilkan beberapa tulisan selama melanjutkan studi, yakni (1) The Influence of Teacher's Gestures to Strengthening the Understanding of Mathematics Students; (2) Developing 'Fort Defending'Game as a Learning Design for Mathematical Literacy Integrated to Primary School Curriculum in Indonesia; (3) Application of cooperative learning model type Think Pair

Share (TPS) on mathematical communication ability; (4) Investigation of Students' Skills in Generating Different Representations to Solve Word Problems: A Case Study in an Elementary School in Indonesia; dan (5) What translation do students make when they carry out a mathematical word problem? Sementara proses review.

Sirajuddin, dilahirkan di Sinjai pada tanggal 29 November 1989. Riwayat pendidikan Dasar dan Menengah ditempuhnya di Kabupaten Sinjai, yaitu SD Negeri 142 Borong Ampirie I, Kelurahan Mananti, Kecamatan Tellulimpoe (lulus tahun 2001), SMP Negei 5 Sinjai Selatan (lulus tahun 2004), dan SMA Negeri 1 Sinjai Selatan (lulus tahun 2007). Pendidikan berikut ditempuhnya di Universitas Muhammadiyah Makassar, Jurusan Pendidikan Matematika dan lulus tahun 2012. Gelar Magister Pendidikan (M.Pd) diraihnya pada tahun 2015 di Program Pascasarjana Universitas Negeri Surabaya. penulis melanjutkan studi S3 Pendidikan Matematika Universitas Negeri Malang pada tahun 2017-2021 dan didaulat sebagai Wisudwan terbaik di Universitas Negeri Malang. Karir sebagai Dosen mulai tahun 2016-sekarang di Universitas Muhammadiyah Makassar. Tergabung dalam Oragnisasi ETDC Indonesia dan I-MES.

Pemrograman MATLAB
Cara Cepat dan Mudah Memahami Bahasa Pemrograman

Penulis:
Muhammad Muzaini
Muhammad Ikram
Nasrun Syahrir
Sirajuddin

Pemrograman MATLAB

MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. MATLAB adalah singkatan dari *Matrix Laboratory* karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks.

Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga memiliki keunggulan umum lainnya, seperti analisis dan eksplorasi data, pengembangan algoritma, pemodelan dan simulasi, visualisasi plot dalam bentuk 2D dan 3D, hingga pengembangan aplikasi antar muka grafis. Dalam ruang lingkup perguruan tinggi, MATLAB digunakan sebagai alat pembelajaran pemrograman matematika, teknik, dan sains pada level pengenalan dan lanjutan, sedangkan dalam dunia industri, MATLAB dipilih sebagai alat penelitian, pengembangan, dan analisis produk industri.

Penerbit Haura Utama
Anggota IKAPI Jawa Barat
Instagram: @haurautama
Website: penerbithaura.com
Email: haurautama@gmail.com

REFERENSI
ISBN 978-623-492-096-3
9 786234 920963