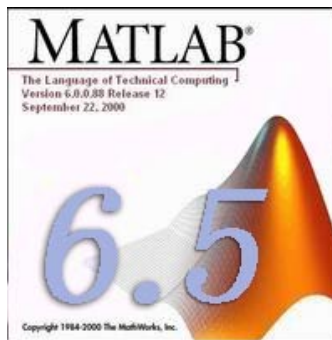




مسئله یکی از قویترین وسایل محاسباتی در رشته های مهندسی کامپیوتر ها می باشند؛ و در انجام محاسبات مهمترین نقش را دارا می باشند.

حال که بر حسب نیازمان کامپیوتر را بعنوان ماشین حسابی قدرتمند تعریف کردیم نیاز به برنامه ای داریم تا تواناییهایش را بکار ببریم : MATLAB

در این جا به معرفی این نرم افزار خواهیم پرداخت، شما را با کاربردها و تواناییهای آن آشنا خواهیم ساخت به این امید که مورد توجه و عنایت شما عزیزان قرار بگیرد.



MATLAB یک نرم افزار قوی جهت دانشجویان و محققین رشته های ریاضی و مهندسی است که اولین نگارشهای آن در دانشگاه نیومکزیکو و استنفورد در سال ۱۹۷۰ در جهت حل مسائل تئوری ماتریسها، جبر خطی و آنالیز عددی بوجود آمد و امروزه صدها هزار کاربر دانشگاهی، آکادمیک، صنعتی و ... در زمینه های بسیار متنوع مهندسی نظیر ریاضیات پیشرفته، جبر خطی، مخابرات، مهندسی سیستم و ... با MATLAB بعنوان یکی از اولین محیط های محاسباتی و تکنیکی که قادر به حل مسائل آنهاست، آشنا می شوند. ریاضیات، زبان مشترک بسیاری از علوم مهندسی است. ماتریسها، معادلات دیفرانسیل، رشته های عددی اطلاعات، ترسیمات و گرافها از لوازم اصلی بکار گرفته در ریاضیات و نیز در MATLAB هستند.

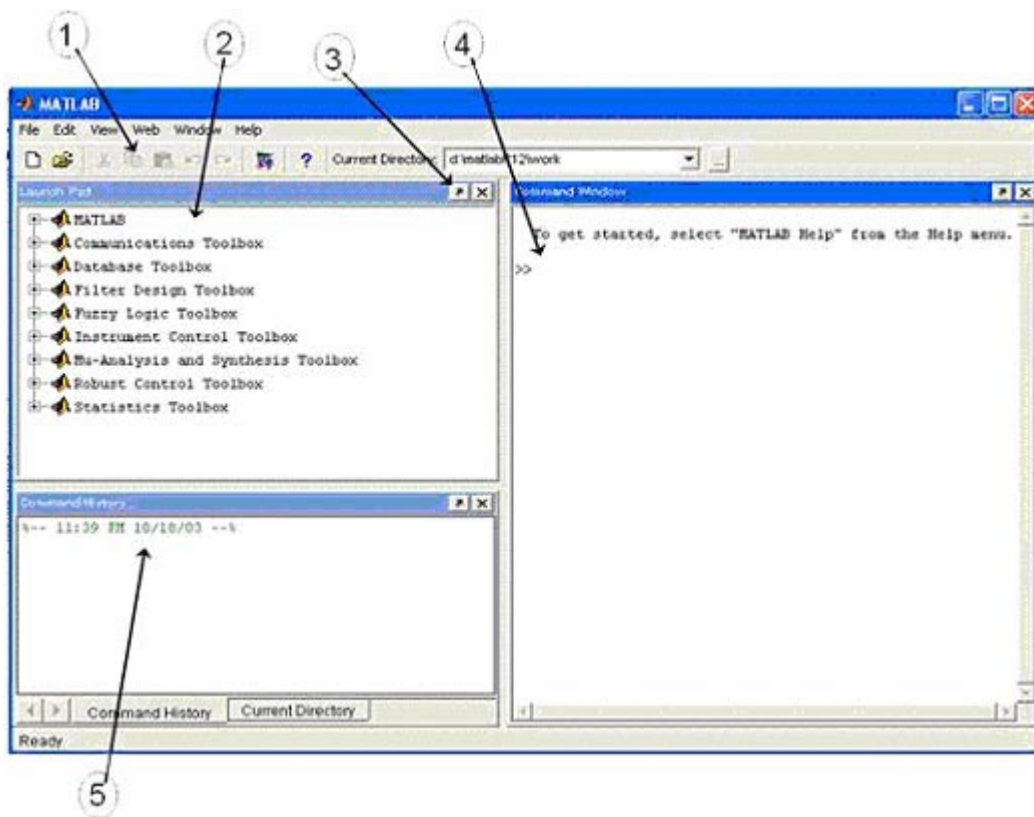
MATLAB اکنون یک سیستم مؤثر و زبان برنامه نویسی بسیاری از محاسبات علمی و مهندسی است.

شما می توانید بسادگی، توابع و برنامه های خاص خودتان را با استفاده از کدها و توابع MATLAB بنویسید و در صورتیکه تعداد آنها زیاد باشد با اختصاص یک زیر شاخه برای آنها از مجموعه آنها یک جعبه ابزار درست کنید. در حقیقت MATLAB یک زبان برنامه نویسی آسان با مشخصات بسیار پیشرفته و ساده تر از زبانهای کامپیوتری نظیر پاسکال و یا C است. این نرم افزار یک محیط پر قدرت برای تصویر کردن اطلاعات را از طریق قابلیت های گرافیکی فراهم می کند.


به دلیل وسعت زیاد MATLAB فقط به موارد کاربردی تر می پردازیم. در صورت نیاز به اطلاعات بیشتر می توانید به help های آخر هر قسمت مراجعه کنید

آشنایی با پنجره های MATLAB :

هنگامی که برای اولین بار MATLAB را اجرا می کنید پنجره ای مطابق شکل زیر مشاهده می شود:



البته می توانید بر اساس نیاز خود پنره های اضافه را بسته و یا پنجره هایی که در این جا دیده نمی شود را باز کرد.
(۱) نشان دهنده جعبه ابزار MATLAB می باشد. که به ترتیب شامل گزینه های New, Open, Cut, Copy, Past می باشد.

بعد از آن دکمه Help بشکل  قرار دارد و در آخر current directory مشخص کننده Directory جاری MATLAB می باشد.

(۲) شماره ۲ کادر Launch Pad نام دارد. این کادر حاوی مجموعه ای از راهنما، مثالها، ابزارهای جنبی و صفحات Web مرتب شده بصورت موضوعی می باشد. با کلیک بر روی هر کدام از مربعهای کنار اسامی موضوعات به گزینه های آن می توان دسترسی داشت.

(۳) روی  کلیک کنید تا کاربرد آن را بفهمید.

(۴) محیط کاری اصلی MATLAB می باشد، تمام دستورات و توابع MATLAB در این قسمت درج می شوند

دستورات جلوی علامت '>>' تایپ می شود. دستور help باعث باز شدن متن راهنما می شود. دستور doc همین کار را انجام می دهد با این تفاوت که نتایج را در پنجره help نمایش می دهد.

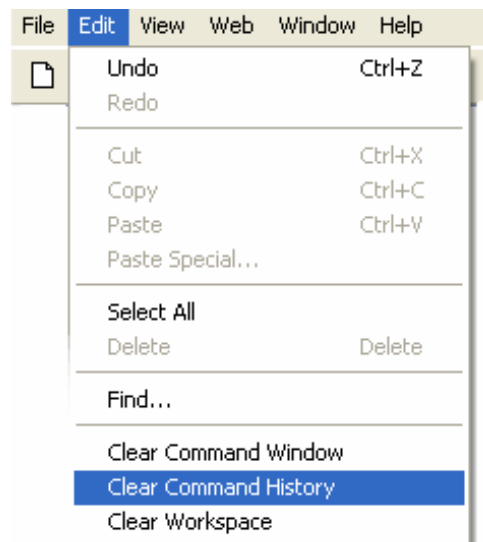
۵) هر بار که شما برنامه MATLAB را باز می کنید زمان و تاریخ ورود شما در پنجره command history به رنگ سبز نشان داده می شود مثلاً: --% ۱:۲۷ ۱۰/۳۱/۰۳ PM --%

سیس دستوراتی را که به برنامه می دهید نیز پشت سر هم در این پنجره ذخیره می شوند. جالب اینکه این دستورات حتی بعد از خارج شدن از برنامه نیز همچنان سر جای خود باقی می مانند و حذف نمی شوند. با کلیک بر روی این دستورات ذخیره شده شما می توانید آنها را مجدداً اجرا کنید.

پنجره workspace

یکی دیگر از پنجره های MATLAB پنجره workspace می باشد. در این پنجره می توانیم تمام متغیرهایی را که در MATLAB تعریف کرده ایم را لیست وار مشاهده کنیم و حتی در آنها تغییراتی ایجاد کرده و یا آنها را بصورت نمودار مشاهده کنیم.

محتویات تمام این پنجره ها را می توان از طریق منوی Edit پاک کرد.



مقدمات کار با MATLAB :

MATLAB اعمال ساده ریاضی را به راحتی یک ماشین حساب انجام می دهد:

```
>> ۲+۴-۱
ans =
    ۵
>> ۲+۸/۲
ans =
    ۷
```

راه دیگر انجام محاسبات این است که مقادیر را در چند متغیر ذخیره کرده و روی متغیرها عملیات محاسباتی را انجام دهیم:

```
>> a=۲
a =
    ۲
>> b=۴
b =
    ۴
>> c=۱
c =
    ۱
>> a+b-c
ans =
    ۵
```

در نام گذاری متغیرها باید موارد زیر را رعایت کرد:

- ۱) MATLAB نسبت به حروف کوچک و بزرگ حساس است.
- ۲) اسامی متغیرها حداکثر می تواند ۳۱ کرکتر باشد.
- ۳) اسامی متغیرها حتما باید با حرف شروع شود. (کرکتر اول نباید عدد باشد)
- ۴) جز کلمات تعریف شده برای MATLAB نباشد. (این کلمات به رنگ آبی نوشته می شود، مثل **for**) برای شناسایی کلمات کلیدی MATLAB میتوان از دستور `iskeyword` استفاده کرد. این تابع در صورتی که عبارت یک کلمه کلیدی باشد مقدار یک و در غیر این صورت مقدار صفر را برمی گرداند.

```
>> iskeyword('for')
ans =
    ۱
>> iskeyword('keyword')
ans =
    ۰
>> iskeyword('ans')
ans =
    ۰
>> iskeyword('if')
ans =
    ۱
```

۵) تمام دستورات و عبارات کلیدی MATLAB با حروف کوچک نوشته می شود. بنابراین می توان اسامی آن ها را با حروف بزرگ برای نام گذاری متغیرها به کار برد.

همان طور که در مثال بالا دیدید MATLAB حاصل $a+b-c$ را در متغیر `ans` ذخیره کرده است. این متغیر به طور پیش فرض برای ذخیره اطلاعات به کار می رود مگر این که کاربر نتایج را در یک متغیر دیگر ذخیره کند.

```
>> D=a+b-c
D =
    5
```

اگر بخواهیم چند دستور را در یک خط بنویسیم باید از کاما (،) و سمیکالن (;) استفاده کنیم. سمیکالن باعث می شود محاسبات انجام شود ولی نتایج نمایش داده نشود.

```
>> a=۳ , b=۴ ; c=۱;
a =
    ۳
```

برای انجام اعمال ساده ریاضی می توان از عملگرهای معمول (+ - / \ (تقسیم) * (ضرب) ^ (توان)) استفاده کرد.

برای صرفه جویی در وقت با استفاده از کلیدهای `up` و `Down` (کلیدهای جهتی) می توانید دستوراتی که قبلاً اجرا شده را مرور کنید. همچنین سایر کلیدهای ویرایشی (`Home`, `End`, `Page Up`, ...) و وظایف استاندارد خود را دارا می باشند.

در MATLAB متغیرهای ویژه ای وجود دارد که هر یک مقادیر خاصی را در خود ذخیره می کنند:

`ans` برای ذخیره مقادیری که کاربر متغیری را برای ذخیره آنها در نظر نگرفته است.
`pi` مقدار عدد پی (۳,۱۴۱۶)
`eps` کوچکترین عدد مثبت بزرگتر از صفر.
`inf` به عنوان علامت بینهایت
`NaN` یا `nan` مقدار غیر عددی (نتیجه تمام عملگرها روی `NaN` ها، `NaN` است)

بهترین روش برای خواناتر شدن یک برنامه استفاده از جملات توضیحی می باشد در MATLAB این جملات بعد از علامت % می آید و MATLAB عبارت بعد از % را ویرایش نمی کند.

گاهی اوقات یک فرمان ممکن است آن قدر طولانی باشد که نتوان آن را در یک خط نمایش داد. برای حل این مشکل می توان در آخر خط سه نقطه (...) گذاشته و ادامه دستور را در خط بعد تایپ کرد. برای توقف پردازش برنامه از `Ctrl + c` استفاده کنید.

در MATLAB اعداد با فرمت های مختلفی به نمایش در می آیند. از مهمترین آن ها می توان به `Format short` اشاره کرد که فرمت پیش فرض است و اعداد را با دقت ۴ رقم اعشار نمایش می دهد. همچنین `Format bank` که بر اساس سیستم بانکی (دلار و سنت) ایجاد شده اعداد را با دقت ۲ رقم اعشار نمایش می دهد.

برای گرد کردن اعداد روش های مختلفی وجود دارد تمام این روش ها را می توان در MATLAB یافت:

`fix` گرد کردن به طرف صفر
`floor` گرد کردن به طرف منهای بینهایت
`ceil` گرد کردن به طرف مثبت بینهایت
`round` گرد کردن به طرف نزدیکترین عدد صحیح

حال به معرفی چند دستور کلیدی می پردازیم:

`Date` دستور ، تاریخ را نمایش می دهد

```
>> date
ans =
05-Jul-2004
```

با استفاده از دستور Clear می توان تمام یا تعدادی از متغیرها را پاک کرد.

```
>> clear a
>> a
??? Undefined function or variable 'a'.
```

```
>> b
b =
     5
>> clear
>> b
??? Undefined function or variable 'b'.
```

همچنین با استفاده از دستور delete می توان فایل های مورد نظر و موجود در دایرکتوری جاری MATLAB را پاک کرد. به عنوان مثال دستور زیر تمام p-file های موجود در دایرکتوری جاری را پاک می کند.

```
>> delete *.p
```

دستور disp مقادیر یک متغیر را بدون نمایش نام آن متغیر چاپ می کند.

شاید بخواهیم عملیاتی را که در یک دوره انجام داده ایم ذخیره کرده و از آن پرینت گرفته و یا بعدها از آن استفاده کنیم. برای این کار از دستور Diary استفاده می کنیم. با اجرای دستور diary on ، MATLAB مانند یک دفترچه یادداشت عمل کرده و تمام مطالب موجود در prompt MATLAB در یک فایل ذخیره می شود تا هنگامی که diary off اجرا شود.

اگر دستور format compact را وارد کرده Enter بزنید MATLAB خطوطی را که بصورت خالی بین خروجی قرار می دهد را حذف می کند. عکس این دستور، format loose است که خطوط خالی حذف شده را بر می گرداند.

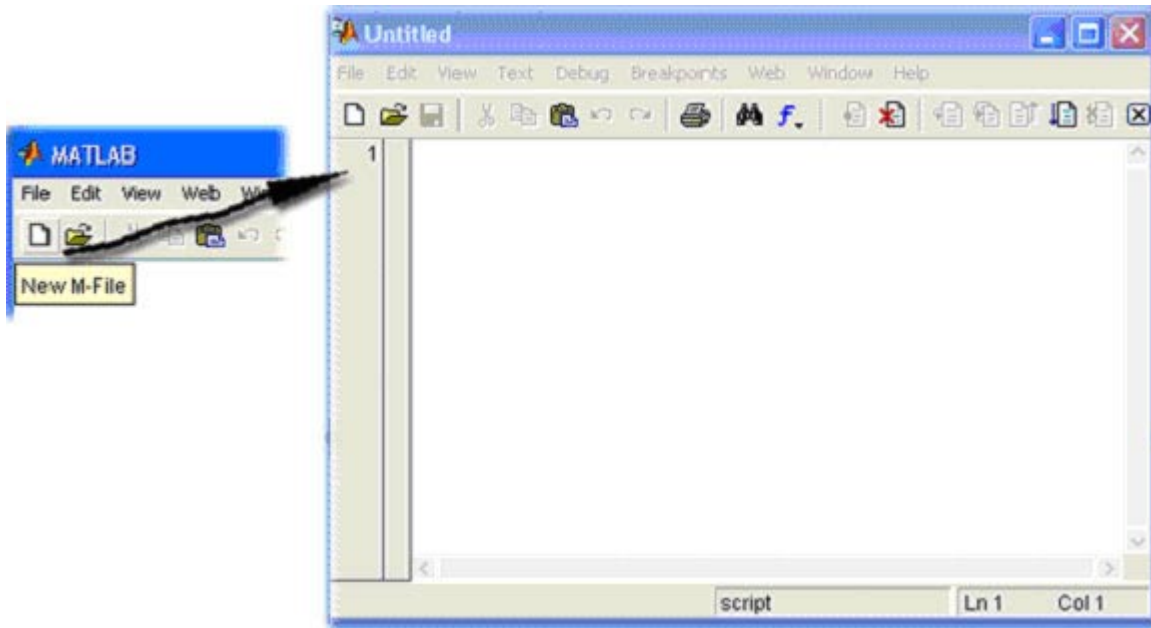
بوسیله دستور format compact می توانیم داده های بیشتری را در پنجره Command window جا دهیم.

همان طور که قبلاً توضیح دادم با استفاده از دستورات Help و Doc می توان به متن راهنمای یک دستور یا تابع دست پیدا کرد. به این طریق می توان با عملکرد آن دستور یا تابع آشنا شده و روش های به کار بردن آن را آموخت. همچنین با سایر دستورات مرتبط آشنا می شویم.

در هنگام اجرای دستورات تکراری یا هنگام آزمایش کردن مقادیر مختلف در یک متغیر ممکن است تایپ دستورات خسته کننده باشد. برای حل این مشکل MATLAB یک راه حل دارد و آن استفاده از M-file می باشد.

با استفاده از M-file ها می توانید دستورات را در یک فایل ذخیره کرده و با باز کردن آن همانند آن که آن ها را در خط فرمان تایپ کرده باشید اجرا کنید. برای ایجاد یک M-file می توانید از گزینه New M-file استفاده کنید.

ما در این جا فقط به آموزش موارد مهم و کلیدی می پردازیم و در آخر هر بخش help-file هایی که می تواند در تکمیل مطلب مفید باشد معرفی خواهیم کرد.



```
>>help format
>>doc help
>>doc doc
>>help clear
>>help diary
>>help save
>>help load
>>help date
>>help ops
>>help fix
```

را په ها درر MATLAB به سادگي ايجاد مي شوندد. ساده ترين و ابتدائي ترين روش تايپ تمام مقادير بين دو [] مي باشد:

```
>> a=[1,2,3,4]
a =
    1    2    3    4
```

اما براي توليد آرايه هاي بزرگتر استفاده از روش بالا بسيار وقت گير است. MATLAB چند دستور براي توليد آرايه ها دارد:

`linspace(a,b,c)`
اين دستور c نقطه با فاصله هاي مساوي در بازه [a,b] را برمي گرداند.

`logspace(a,b,c)`
اين دستور نيز c نقطه با فواصل لگاريتمي در بازه $10^a, 10^b$ بر مي گرداند.

`a:b:c`
مفهوم كلي اين دستور را مي توان به اين صورت بيان كرد؛ از a شروع كن b تا b تا برو جلو تا به c برسي. در اين روش b كه گام حركت نام دارد مي تواند منفي باشد.

با استفاده از دستورات فوق و توابع رياضي مي توان ماتريس هاي متنوعي توليد كرد.

```
>> x=0:.5:2*pi;
>> y=sin(x)
y =
Columns 1 through 10
    0    0.4794    0.8415    0.9975    0.9093    0.5985    0.1411   -0.3508   -0.7568   -0.9775
Columns 11 through 13
-0.9589   -0.7055   -0.2794
```

```
>> z=tanh(x)
z =
Columns 1 through 10
    0    0.4621    0.7616    0.9051    0.9640    0.9866    0.9951    0.9982    0.9993    0.9998
Columns 11 through 13
0.9999    1.0000    1.0000
```

آرايه ها در MATLAB تنها به آرايه هاي سطري محدود نمي شوندد بلکه مي توان آرايه ها ي ستوني نيز معرفي كرد. براي اينكه به MATLAB بفهمانيم كه قصد توليد سطر جديد را داريم از سميكالن (!) استفاده مي كنيم

```
>> b=[1;3;5;7]
b =
    1
    3
    5
    7
>> b=linspace(1,7,4)'
b =
    1
    3
    5
    7
```

همان طور که در مثال بالا مشاهده کردید با استفاده از عملگر ترانهاده (') يك آرایه سطري را به آرایه ستوني تبدیل کردیم.

برای دسترسی به درایه ها از اندیس آن استفاده می شود. به عنوان مثال b(3) سومین درایه آرایه b (یعنی 5) را برمی گرداند.

اگر بخواهیم درایه های خاصی از يك آرایه را انتخاب کنیم می توان با استفاده از دو نقطه به آنها دسترسی داشته باشیم. اگر طول آرایه مشخص نباشد می توان برای دسترسی به آخرین عنصر از end استفاده کرد

```
>> z(end:-3:4)
ans =
    1.0000    0.9998    0.9951    0.9051
```

همچنین می توان با استفاده از يك آرایه به مقادیر يك آرایه دیگر با ترتیب مورد نظر خودمان استفاده کنیم.

```
>> c=[1,5,3,5,1];
>> z(c)
ans =
     0    0.9640    0.7616    0.9640     0
```

همان طور که مشاهده کردید با استفاده از آرایه c عناصر اول، پنجم، سوم، پنجم و اول را فراخوانی کردیم.

نکته دیگر که باید به آن توجه کرد این است که اندیس يك عدد صحیح مثبت است. اگر کاربر يك عدد غیر صحیح و یا منفي را به عنوان اندیس وارد کند MATLAB يك پیغام خطا بر می گرداند.

```
>> b(3.4)
??? Subscript indices must either be real positive integers or logicals.
```

```
>> b(-4)
??? Subscript indices must either be real positive integers or logicals.
```

دو آرایه مفروض a و b را می توان با استفاده از دستورات [a,b] (الحاق سطري) و [a;b] (الحاق ستوني) به یکدیگر الحاق کرد و به این ترتیب آرایه هایی با سطرها و ستون های متعدد داشت؛ البته در الحاق سطري تعداد سطرها و در الحاق ستوني تعداد ستون های دو آرایه باید برابر باشد. همچنین می توان تعداد آرایه های الحاقی را به طور دلخواه افزایش داد.

```
>> a=[1 2 3];
>> b=[4 5 6];
>> c=[7 8 9];
>> d=[a b],f=[a;b;c]
d =
     1     2     3     4     5     6
f =
     1     2     3
     4     5     6
     7     8     9
```

به این ترتیب متوجه شدید که آرایه ها در MATLAB می توانند دارای سطرها و ستون های متعدد باشند. حال ممکن است این سوال پیش بیاید که چگونه می توان این آرایه ها را تولید کرد. همان طور که قبلا ملاحظه نمودید برای معرفی ستون ها از فاصله یا کاما (,) و برای سطرها از سمیکالن (!) استفاده می شود. راه دیگر ایجاد سطر این است که بعد از معرفی سطر اول با زدن کلید Enter به خط بعد رفته و به معرفی سطر بعد پردازید.

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

```

A =
    1     2     3
    6     5     4

>> B = [10 11 12
13 14 15]
B =
    10    11    12
    13    14    15

```

در این مورد به چند نکته باید توجه کرد:
تعداد ستون ها در هر سطر باید برابر باشد در غیر این صورت پیغام خطایی نمایش داده می شود.
تعداد فواصلی که برای جدا کردن اعداد به کار می رود برای MATLAB مهم نیست. یعنی MATLAB فضاها
خالی اضافی را تشخیص داده و حذف می کند.

محاسبات عددی آرایه ها:

MATLAB عملگرهای فراوانی برای اعمال بر روی آرایه ها دارد. تعدادی از آنها را در مثال های زیر مشاهده می کنید:

```

>> A
A =
    1     2     3
    6     5     4
>> A-2
ans =
   -1     0     1
    4     3     2
>> A*2 + B
ans =
    12    15    18
    25    24    23
>> A + B
ans =
    11    13    15
    19    19    19

```

همان طور که در بالا دیدید اعمال ساده ریاضی را می توان روی آرایه ها انجام دهید. $A-2$ باعث می شود از تمام درایه های آرایه A دو واحد کم شود. همچنین دستور $A+B$ درایه های نظیر به نظیر دو آرایه را جمع می کند.

ضرب و توان ماتریس ها نیز به سادگی قابل تعریف اند:

```

>> A
A =
    1     2     3
    6     5     4
>> C=[1 2;3 4];
>> C*A
ans =
    13    12    11
    27    26    25
>> C^2

```

```
ans =  
 7 10  
15 22
```

حال اگر بخواهیم تک تک درایه ها را به توان n برسانیم یا درایه های دو آرایه یا ماتریس را نظیر به نظیر در هم ضرب یا تقسیم کنیم باید قبل از این عملگرها یک نقطه قرار دهیم. به مثال های زیر توجه کنید:

```
>> A.*B  
ans =  
 10 22 36  
 78 70 60
```

```
>> B.^A  
ans =  
 10 121 1728  
4826809 537824 50625
```

```
>> C.^2  
ans =  
 1 4  
 9 16
```

آرایه های استاندارد:

```
>> ones(2,3)  
ans =  
 1 1 1  
 1 1 1
```

$ones(n,m)$ ماتریسی $n \times m$ با درایه های یک ایجاد می کند.

```
>> zeros(2,4)  
ans =  
 0 0 0 0  
 0 0 0 0
```

$zeros(n,m)$ ماتریسی $n \times m$ با درایه های صفر ایجاد می کند.

```
>> eye(3,4)  
ans =  
 1 0 0 0  
 0 1 0 0  
 0 0 1 0
```

این دستور هم ماتریسی $n \times m$ ایجاد می کند که درایه های روی قطر اصلی آن یک است. اگر این دستور به صورت $eye(n)$ به کار رود یک ماتریس همبندی به وجود می آید.

```
>> rand(2,3)  
ans =  
 0.9501 0.6068 0.8913  
 0.2311 0.4860 0.7621
```

این دستور هم یک ماتریس با درایه های اتفاقی بین صفر و یک ایجاد می کند.

```
>> randperm(8)  
ans =  
 2 4 1 5 8 6 3 7
```

```
>> randperm(10)
```

```
ans =  
10 7 1 8 2 5 9 6 4 3
```

این دستور اعداد 1 تا n را به صورت اتفاقی در یک بردار سطری قرار می دهد.

```
>> magic(3)
```

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

این دستور ماتریسی که به ماتریس جادویی معروف است تولید می کند. ویژگی این ماتریس این است مجموع درایه های هر سطر، ستون و قطر با هم برابر است.

توجه به این نکته لازم است که دو دستور آخر تنها یک ورودی دارند، اما دستورات اول می توانند دارای دو آرگومان ورودی باشند. در صورتی که دستورات اول با یک آرگومان به کار بروند یک ماتریس مربعی $n \times n$ ایجاد می شود.

اندازه ماتریس ها:

در MATLAB دستوراتی وجود دارد که به وسیله آنها می توان اطلاعاتی در مورد یک ماتریس از قبیل تعداد سطر، ستون و تعداد کل عناصر آن را به دست آورد.

```
>> a = [1 2 3  
1 2 3]
```

```
a =  
1 2 3  
1 2 3
```

```
>> size(a)
```

```
ans =  
2 3
```

این دستور همان طور که از اسم آن مشخص است اندازه (تعداد سطر و ستون) ماتریس را برمی گرداند. در دستوراتی مانند دستور فوق که 2 خروجی یا بیشتر دارند می توان هر خروجی را در یک متغیر ذخیره کرد برای این کار به صورت زیر عمل می کنیم:

```
>> [s t] = size(a)
```

```
s =  
2
```

```
t =  
3
```

```
>> length(a)
```

```
ans =  
3
```

این دستور بزرگترین مقدار بین سطر و ستون را برمی گرداند.

```
>> numel(a)
```

```
ans =  
6
```

این دستور هم تعداد عناصر (number of elements) ماتریس را برمی گرداند.

فرض کنید می خواهیم عناصر قطر اصلی یک ماتریس را به دست آوریم و یا مجموع آنها را به دست آوریم. دستوری زیر به ما کمک خواهد کرد.

```
>> diag(a)
ans =
     1
     2
```

```
>> trace(a)
ans =
     3
```

البته دستور `diag` کاربردهای دیگری نیز دارد. به عنوان مثال اگر به صورت `diag(a,n)` به کار رود قطر n ام ماتریس a را می دهد. n می تواند مثبت یا منفی باشد. در صورتی که به صورت `diag(a)` به کار رود و a یک بردار باشد ماتریسی ایجاد می کند که قطر اصلی آن بردار a و سایر عناصر آن صفر است.

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

```
>> diag(c)
ans =
     1     0     0
     0     2     0
     0     0     3
```

```
>> minfo(ans)
3 rows 3 cols: regular MATLAB matrix
```

دستور آخر هم اطلاعاتی در مورد ماتریس مورد نظر را برمی گرداند.

ایجاد تغییر در ماتریس ها:

MATLAB با اختصاص یک اندیس به هر عضو آرایه راه های زیادی را برای ایجاد تغییر در درایه های ماتریس ها به وجود می آورد.

```
>> c = ans
c =
     1     0     0
     0     2     0
     0     0     3
```

```
>> c(3,3) = 8
c =
     1     0     0
     0     2     0
     0     0     8
```

```
>> c(9)=4
c =
     1     0     0
     0     2     0
     0     0     4
```

همان طور که قبلا گفته شد برای دسترسي به عناصر آرایه ها 2 راه وجود دارد. راه اول با 2 آرگومان ورودی که سطر و ستون را مشخص می کنند. راه دیگر استفاده از يك آرگومان که در این صورت اندیس هر درایه به صورت زیر (ماتریس S) تعیین می شود:

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

```
>> c(4)=5
```

```
c =  
 1  5  0  
 0  2  0  
 0  0  4
```

```
>> c(:,1)=7
```

```
c =  
 7  5  0  
 7  2  0  
 7  0  4
```

مفهوم این دستور این است که تمام سطرها در ستون اول را مساوی 7 قرار بده.

```
>> c([1 end],[1 end])=8
```

```
c =  
 8  5  8  
 7  2  0  
 8  0  8
```

این دستور نیز مثل دستور بالا عمل می کند یعنی در سطر اول و آخر ستون های اول و آخر را مساوی 8 قرار می دهد.

```
>> c(1,:)= []
```

```
c =  
 7  2  0  
 8  0  8
```

این دستور هم با مساوی قرار دادن سطر اول با آرایه تهی این سطر را حذف می کند.

تغییراتی که در ماتریس ها می توان ایجاد کرد تنها به تغییر عناصر آن محدود نمی شود بلکه می توان ابعاد، ترتیب و جای درایه ها را تغییر داد.

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

```
>> flipud(a)
```

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

این دستور ماتریس را از بالا به پایین می چرخاند . یعنی جای سطرها را نسبت به سطر وسط عوض می کند.

```
>> b=fliplr(a)
```

```
b =  
 3  2  1  
 6  5  4
```

9 8 7

این دستور هم مانند دستور فوق است با این تفاوت که روی ستون ها عمل می کند.

```
>> rot90(b)
```

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

این دستور ماتریس را 90 درجه در خلاف حرکت عقربه های ساعت می چرخاند. البته اگر این دستور به صورت $\text{rot90}(a,n)$ به کار رود n تعداد چرخش را مشخص می کند.

```
>> triu(ans)
```

```
ans =  
    1    4    7  
    0    5    8  
    0    0    9
```

```
>> tril(ans)
```

```
ans =  
    1    0    0  
    0    5    0  
    0    0    9
```

این دو دستور هم ماتریس های بالا و پایین مثلثی تولید می کنند.

```
>> c=horzcat(ans,b)
```

```
c =  
    1    0    0    3    2    1  
    0    5    0    6    5    4  
    0    0    9    9    8    7
```

```
>> c=vertcat(ans,b)
```

```
c =  
    1    0    0  
    0    5    0  
    0    0    9  
    3    2    1  
    6    5    4  
    9    8    7
```

این دو دستور هم الحاق سطری و ستونی را انجام می دهد.

```
>> reshape(c,2,9)
```

```
ans =  
    1    0    6    0    0    5    0    9    4  
    0    3    9    5    2    8    0    1    7
```

اگر بخواهیم ابعاد یک ماتریس را تغییر بدهیم می توانیم از این دستور استفاده کنیم. البته باید تعداد المنت ها قبل و بعد از تغییر با هم برابر باشد.

```
>> repmat(ans(:,1),2,4)
```

```
ans =  
    1    1    1    1  
    0    0    0    0  
    1    1    1    1  
    0    0    0    0
```

این دستور با تکرار يك ماتریس ماتریسی با ابعاد جدید تولید می کند. به عنوان مثال اگر دستور به این صورت باشد `repmat(a,m,n)` MATLAB ماتریس `a` را به عنوان يك المنت در نظر گرفته و `mxn` بار تکرار می کند.

يکي ديگر از دستورات کاربردي MATLAB دستور `cat(n,a,b)` است. در این دستور `a` و `b` دو ماتریس و `n` مشخص کننده جهت الحاق می باشد. مثال هاي زیر بهتر مفهوم مورد نظر را می رساند. مثال سوم نمونه اي از ماتریس هاي 3 بعدي می باشد. به نحوه نمایش این ماتریس ها دقت کنید.

```
>> d=cat(1,a,b)
```

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

```
>> d=cat(2,a,b)
```

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

```
>> d=cat(3,a,b)
```

```
d(:, :, 1) =  
 1  2  3  
 4  5  6  
 7  8  9  
d(:, :, 2) =  
 3  2  1  
 6  5  4  
 9  8  7
```

در زیر يکي ديگر از دستورات ایجاد تغییر در ماتریس ها را مشاهده می کنید.

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

```
>> b=[2 3];
```

```
>> kron(a,b)
```

```
ans =  
 2  3  4  6  
 6  9  8  12
```

```
>> kron(b,a)
```

```
ans =  
 2  4  3  6  
 6  8  9  12
```

مثال اخير را می توان به این صورت نیز نمایش داد:

```
>> [ b(1)*a , b(2)*a ]
```

```
ans =  
 2  4  3  6  
 6  8  9  12
```

مرتب کردن آرایه ها:

یکی از امکانات مفید MATLAB وجود دستوری برای مرتب کردن آرایه ها می باشد. در زیر ابتدا یک ماتریس با درایه های بین 0 و 20 تولید می کنیم و سپس با استفاده از دستور `sort(a,n)` که `n` مشخص کننده سطر یا ستون می باشد آن را مرتب می کنیم.

```
>> a=fix(20*rand(3,4))
a =
     8     8     7     2
    18    17    16     4
    18     1     0     3
```

```
>> sort(a,1)
ans =
     8     1     0     2
    18     8     7     3
    18    17    16     4
```

```
>> [sor,pos]=sort(a,2)
sor =
     2     7     8     8
     4    16    17    18
     0     1     3    18
pos =
     4     3     1     2
     4     3     2     1
     3     2     4     1
```

همان طور که مشاهده می کنید در صورتی که از MATLAB دو خروجی بخواهیم آنگاه دو ماتریس برمی گرداند؛ ماتریس اول همان ماتریس مرتب شده و ماتریس دوم اندیس مربوط به درایه های مرتب شده را نمایش می دهد. به عبارت دیگر نشان دهنده مکان درایه قبل از مرتب شدن می باشد.

در این گونه دستورات در صورتی که `n` توسط کاربر مشخص نشود 2 حالت پیش می آید.
(1) در صورتی که ماتریس 2 بعدی یا یک بردار ستونی باشد ستون ها مورد بررسی قرار می گیرند.
(2) در صورتی که ماتریس یک بردار سطری باشد سطرها بررسی می شوند.

جستجو در آرایه ها:

در MATLAB دستورات زیادی برای جستجو در آرایه ها، پیدا کردن عناصر خاص و ... وجود دارد. در زیر به برخی از آنها اشاره می شود.

```
>> a=[0 1 0;2 0 3;1 3 4];
>> nnz(a)
ans =
     6

>> find(a)
ans =
     2
     3
     4
     6
```

8
9

```
>> nonzeros(a)
ans =
     2
     1
     1
     3
     3
     4
```

این 3 دستور تقریباً با هم در ارتباط اند. دستور اول تعداد درایه های غیر صفر را برمی گرداند. دستور دوم اندیس مربوط به این درایه ها و دستور سوم خود درایه های غیر صفر را برمی گرداند.

```
>> all(a)
ans =
     0     0     0
```

```
>> all(a,2)
ans =
     0
     0
     1
```

دستور $all(a,n)$ در صورتی که تمام درایه های سطر یا ستون (بستگی به n دارد) غیر صفر باشند مقدار یک و در غیر این صورت صفر را برمی گرداند.

در این دستور هم چنانچه n مشخص نشود مانند آنچه گفته شد عمل می شود.

```
>> any(a,2)
ans =
     1
     1
     1
```

این تابع در صورتی که یکی از المنت ها غیر صفر باشند مقدار یک و در صورتی که همه صفر باشند مقدار صفر را برمی گرداند.

```
>> isempty(a)
ans =
     0
```

این تابع در صورتی که ماتریس یک ماتریس تهی باشد مقدار 1 را بر می گرداند.

```
>> b=[9 3 0 8];
>> ismember(b,a)
ans =
     0     1     1     0
```

این دستور در صورتی که اعضای ماتریس b (که می تواند یک عدد باشد) عضو a نیز باشند مقدار یک و در غیر این صورت مقدار صفر را برمی گرداند.

```
>> unique(a)
ans =
     0
     1
     2
     3
```

4

و این تابع يك بردار ستونی شامل عناصر غير تكراري a که به ترتیب صعودی مرتب شده اند را برمی گرداند.

کاربرد در داده های آماری:

MATLAB دستورات فراوانی برای کار کردن روی داده ها و انجام دادن کارهای آماری دارد. در زیر به مهمترین آنها اشاره می شود.

```
>> a=fix(20*rand(3,4))
a =
    19     9     9     8
     4    17     0    12
    12    15    16    15
```

یکی از پرکاربردترین دستورات 2 دستور زیر است که به ترتیب مینیمم و ماکسیمم هر ستون را به دست می آورند. در صورتی که ماتریس يك بردار سطری باشد این کار را روی سطر انجام می دهند.

```
>> min(a)
ans =
     4     9     0     8
```

```
>> [s,t]=max(a)
s =
    19    17    16    15
t =
     1     2     3     3
```

همان طور که ملاحظه می کنید در صورتی که مانند دستور sort دو خروجی بخواهیم خروجی دوم مشخص کننده مکان درایه مینیمم یا ماکزیمم خواهد بود. در صورتی که این دستور به صورت $\max(a,b)$ (یا \min) به کار رود که b يك ماتریس با ابعاد ماتریس a یا يك عدد است_ آنگاه خروجی يك ماتریس است با درایه های بزرگتر بین a و b.

```
>> b=fix(45*rand(3,4))
b =
    41    18    18    15
    33    42    40    36
     7    41     2     0
```

```
>> max(a,b)
ans =
    41    18    18    15
    33    42    40    36
    12    41    16    15
```

همچنین می توان میانگین و عضو میانی هر سطر یا ستون را به دست آورد.

```
>> mean(b,2)
ans =
    23.0000
    37.7500
    12.5000
```

```
>> median(b,2)
ans =
```

```
18.0000
38.0000
4.5000
```

```
>> median(b,1)
ans =
    33    41    18    15
```

به این نکته دقت کنید که تابع median پس از مرتب کردن سطر یا ستون عنصر میانی را برمی گرداند. و در صورتی که تعداد سطرها یا ستون ها زوج باشد میانگین 2 عضو وسط را برمی گرداند.

```
>> sum(b)
ans =
    81   101    60    51
```

مجموع هر سطر یا ستون را برمی گرداند.

```
>> cumsum(b)
ans =
    41    18    18    15
    74    60    58    51
    81   101    60    51
```

عملکرد آن مشابه سیگما در ریاضی است. یعنی مجموع هر درایه با درایه های قبل از آن را بر می گرداند.

```
>> prod(b)
ans =
    9471    30996    1440     0
```

حاصلضرب درایه های هر ستون را برمی گرداند.

```
>> cumprod(b)
ans =
     41     18     18     15
    1353    756    720    540
    9471   30996   1440     0
```

این دستور حاصلضرب هر درایه در درایه های ماقبل را بر می گرداند. از این دستور می توان در شبیه سازی فاکتوریل استفاده کرد.

در دستورات می توان جهت انجام عملیات (سطر یا ستون) را مشخص کرد. در صورتی که جهت مشخص نشود مانند دیگر دستورها که در بالا گفته شد عمل می شود.

یکی دیگر از دستورات که برای مقایسه دو ماتریس به کار می رود دستور زیر است:

```
>> isequal(a,b)
ans =
     0
```

در صورتی که دو ماتریس برابر باشند مقدار یک را برمی گرداند.

ماتریس به عنوان ضرایب چند جمله ای:

یک دیگر از کاربردهای ماتریس ها استفاده از آنها به عنوان ضرایب یک چندجمله ای است.

```
>> x=[2 3 6 10];
>> y=[1 2 3 4];
```

فرض کنید تعدادی داده آماری دارید و می خواهید برای ارتباط دادن آنها با یکدیگر تابعی را پیدا کنید. MATALAB این کار را به راحتی و به وسیله $\text{polyfit}(x,y,n)$ انجام می دهد. در اینجا x,y داده ها و n مشخص کننده درجه چندجمله ای مورد نظر است. نتیجه این تابع یک ماتریس است.

```
>> p=polyfit(x,y,3)
p =
    0.0193   -0.3795    2.5298   -2.6964
```

```
>> polyval(p,x)
ans =
    1.0000    2.0000    3.0000    4.0000
```

حال فرض کنید یک چند جمله ای داریم و می خواهیم مقادیر آن را به ازای مقادیر مختلف یک متغیر به دست آوریم. برای این کار از تابع polyval استفاده می کنیم.

```
>> r=roots(p)
r =
    9.1557 + 4.8026i
    9.1557 - 4.8026i
    1.3040
```

همان طور که گفته شد می توان ماتریس ها را به عنوان ضرایب یک چندجمله ای در نظر گرفت. در این صورت اگر p یک بردار شامل ضرایب چندجمله ای باشد ریشه های آن به روش بالا به دست می آید.

```
>> p2=poly(r)
p2 =
    1.0000   -19.6154   130.7692  -139.3846
```

این دستور عکس دستور roots می باشد. یعنی با داشتن ریشه های یک چندجمله ای می توانید ضرایب آن را به دست آورید.

در محاسبات ریاضی گاهی باید 2 چندجمله ای را در هم ضرب یا بر هم تقسیم کرد. 2 دستور زیر به ترتیب این کارها را انجام می دهند.

```
>> p1=[2 0 3 -1];
>> p2=[2 4];
>> conv(p1,p2)
ans =
     4     8     6    10    -4
```

```
>> deconv(p1,p2)
ans =
    1.0000   -2.0000    5.5000
```

جمع و تفریق چندجمله ای ها نیز به سادگی جمع و تفریق ماتریس ها می باشد. البته دو ماتریس باید برابر باشند.

```
>> p3=[3 2 0 1];
>> p1 + p3
ans =
     5     2     3     0
```

ماتریس به عنوان مجموعه:

یک دیگر از کاربردهای گسترده ماتریس ها در نظر گرفتن آنها به عنوان یک مجموعه می باشد.

```
>> a=randperm(8)
a =
    4    3    2    6    8    1    5    7
```

```
>> b=2:2:8
b =
    2    4    6    8
```

```
>> union(a,b)
ans =
    1    2    3    4    5    6    7    8
```

```
>> intersect(b,a)
ans =
    2    4    6    8
```

دستور اول اجتماع و دستور دوم اشتراک دو مجموعه را به دست می آورد.

```
>> setxor(a,b)
ans =
    1    3    5    7
```

اعضایی که یا فقط در a هستند یا فقط در b . به عبارت دیگر اجتماع منهی اشتراک. (تفاضل متقارن)

```
>> setdiff(a,b)
ans =
    1    3    5    7
```

اعضایی از a که در b نیستند.

```
>> setdiff(b,a)
ans =
Empty matrix: 1-by-0
```

اعضایی از b که در a نیستند.

محاسبات ماتریسی:

در زیر به ترتیب معکوس، دترمینان و ترانزاده ماتریس a را محاسبه می کنیم.

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

```
>> inv(a)
ans =
    0.3913    0.2391    0.2174
   -0.0435   -0.3043    0.0870
   -0.0870   -0.1087    0.1739
```

```
>> det(a)
```

```
ans =  
    -46
```

```
>> a'  
ans =  
     2     0     1  
     3    -4    -1  
    -4     2     5
```

مدیریت فایل ها و متغیرها:

MATLAB چند دستور برای آگاهی کاربر از متغیرها و فایل های موجود دارد که در زیر به آنها اشاره می شود.

```
>> what  
M-files in the current directory F:\MATLAB6_5\work
```

```
bub_dew    cumprod2    exm        mostafa    prod2     size2     sum2  
calculate  cumsum2    moadel     name       seri      star
```

این تابع M-file های موجود در دایرکتوری جاری را نمایش می دهد. برای تغییر دایرکتوری می توانید همانند سیستم عامل dos از دستور cd استفاده کنید.

```
>> who  
Your variables are:  
a  ans  b  p  p2  pos  r  sor  x  y
```

```
>> whos  
Name      Size      Bytes Class  
a          3x3       72   double array  
ans        1x5       40   double array  
b          1x4       32   double array  
p          1x4       32   double array  
p2         1x4       32   double array  
pos        3x4       96   double array  
r          3x1       48   double array (complex)  
sor        3x4       96   double array  
x          1x4       32   double array  
y          1x4       32   double array
```

Grand total is 61 elements using 512 bytes

یکی از دستورات جالب MATLAB که بیشتر جنبه سرگرمی دارد دستور why است. این دستور را امتحان کنید.

```
>> why  
Some hamster insisted on it.
```

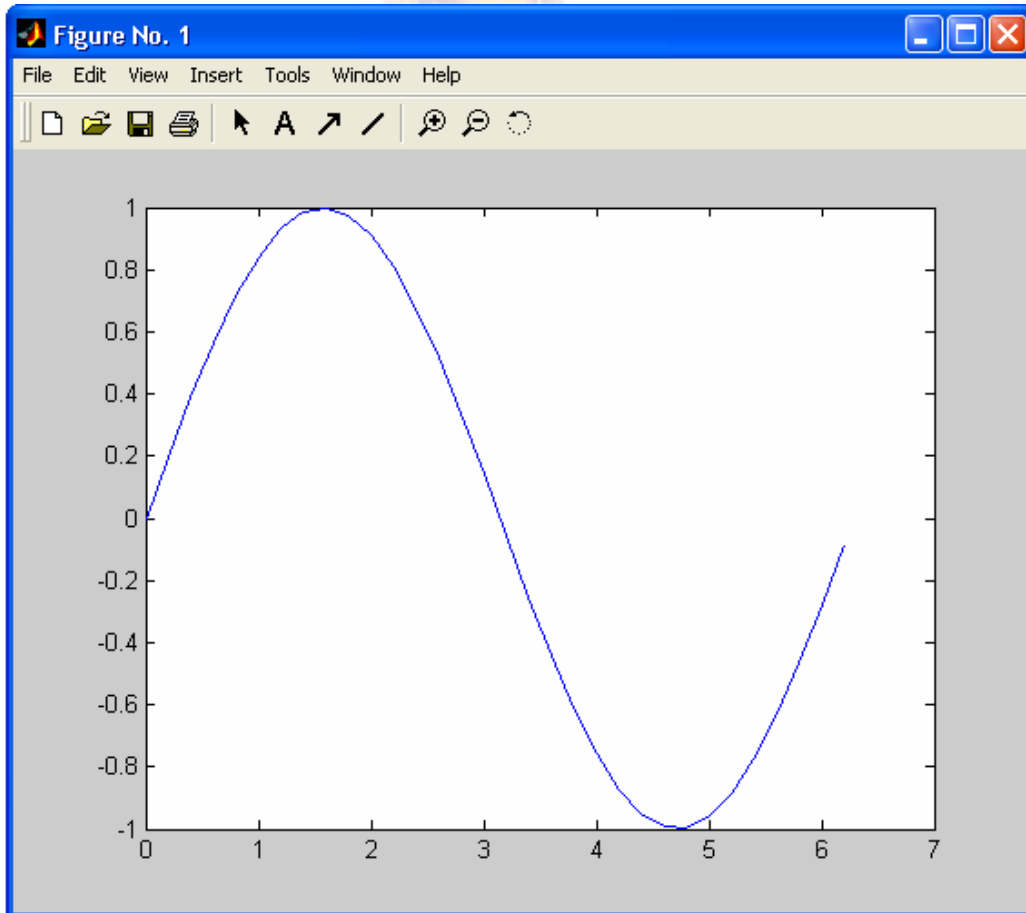
دسته ديگر توابع MATLAB توابع مربوط به رسم نمودار مي باشند. نمودارها قادر به انتقال اطلاعاتي هستند كه شايد خيلي از جداول و ليست ها قادر به انتقال آن نباشند. به همين دليل اين بخش را به معرفي توابع مربوط به رسم نمودار اختصاص داديم. البته به خاطر گستردگي اين توابع تنها توابع نمودارهاي 2 بعدي را معرفي مي كنيم. شناخت اين توابع كار با ديگر توابع نموداري را راحت مي كند.

تابع plot :

متداول ترين تابع رسم نمودارهاي 2 بعدي اين تابع مي باشد. اين تابع مجموعه اي از آرايه هاي داده ها را بر روي محورهاي مختصات رسم کرده و نقاط تعيين شده را با خطوط مستقيم به هم متصل مي كند.

```
>> x = 0:0.2:2*pi;  
>> y = sin(x);  
>> plot(x,y)
```

در مثال بالا x محور افقي و y محور عمودي را مي سازند. (در تابع آرگومان اول محور افقي و آرگومان دوم محور عمودي را مشخص مي كند). تابع plot پنجره گرافيكي figure را باز مي كند، سپس اندازه محورهاي مختصات را مطابق داده ها تنظيم مي كند. بعد از رسم نقاط آنها را با خطوط راست به يكديگر متصل مي كند. در زير نتيجه دستورات بالا و پنجره figure را مشاهده مي كنيد.

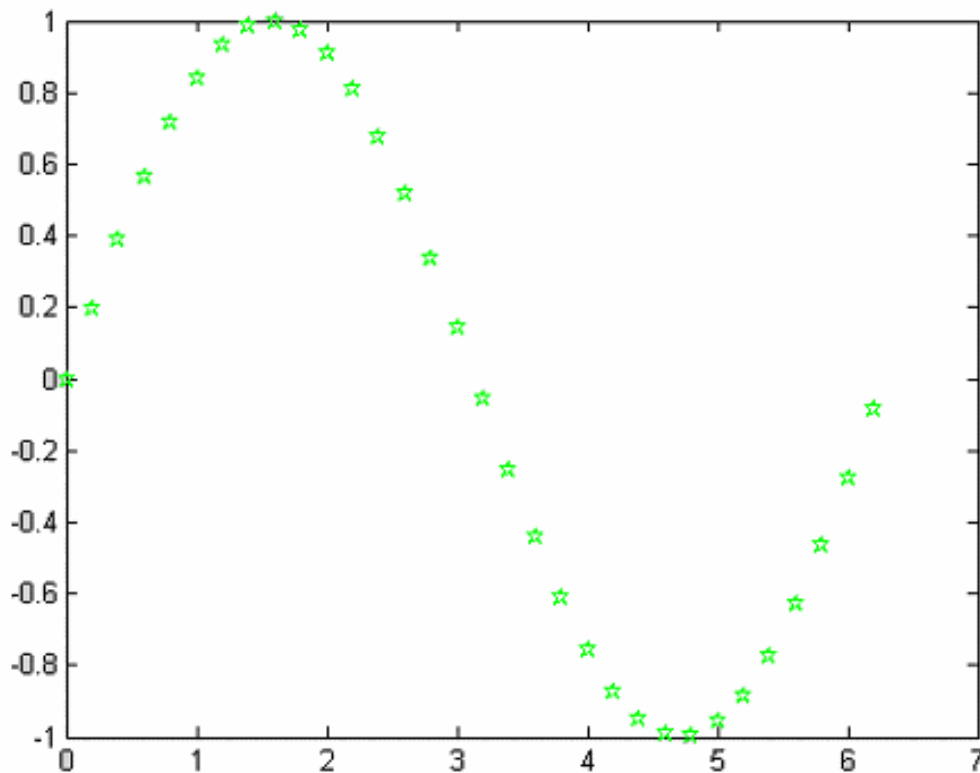


تابع plot را مي توان به همراه آرگومان سومي نيز به كار برد. اين آرگومان كه پس از x و y مي آيد يك رشته كرتري است كه مشخص كننده نوع خطوط و رنگ آنها مي باشد. اين رشته شامل يك يا چند كرتري از جدول زير است.

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		h	hexagram		
		p	pentagram		

به مثال زیر توجه کنید:

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



جدول فوق به سه ستون تقسیم می شود. ستون اول (از چپ) رنگ، دوم نقاط و سومین ستون نوع خط را مشخص می کند. تا هنگامی که نوع خط را مشخص نکنید نقاط به هم متصل نمی شوند. در استفاده از کرکتهای این جدول از هر ردیف تنها یک کرکتر را انتخاب کنید در غیر این صورت پیغام خطایی نمایش داده می شود. به این صورت رشته مورد نظر حداکثر دارای 3 کرکتر است.

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

??? Error using ==> plot
Error in color/linetype argument.

این کرکترها و ترکیب آنها با یکدیگر را امتحان کنید تا با آنها بیشتر آشنا شوید.

در صورتی که ترتیب آرگومان ها را تغییر دهید نمودار هم 90 درجه دوران پیدا می کند. یعنی نمودار y بر حسب x به نمودار x بر حسب y تبدیل می شود.

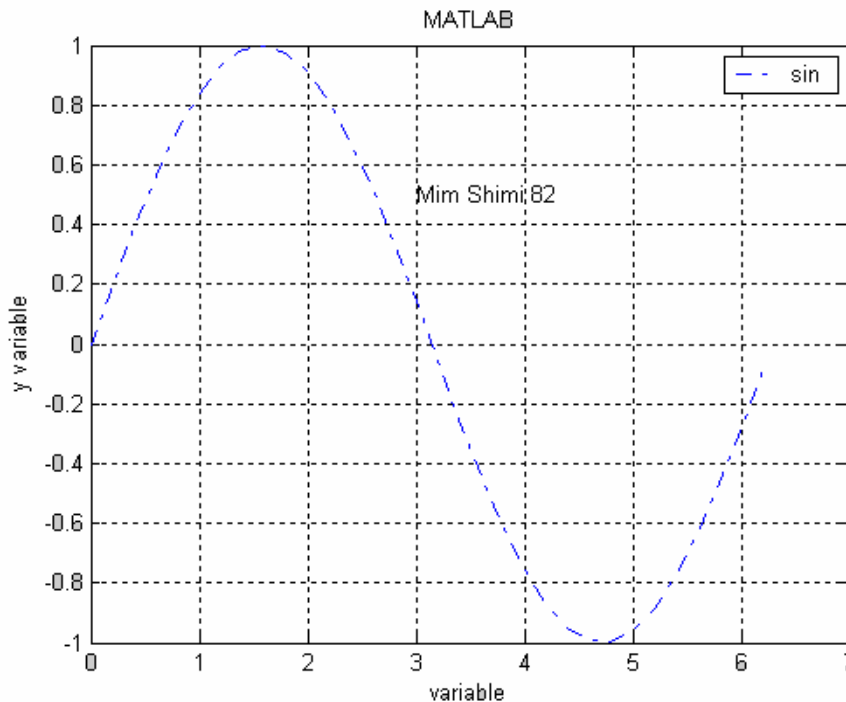
برچسب ها، تنظیمات و ... :

دستورات xlabel و ylabel برچسب محورها را مشخص می کنند. دستور title عنوان را بالای نمودار قرار می دهد.

راهنمای نمودار نیز در صورت رسم چند نمودار روی یک پنجره می تواند مفید باشد. دستور legend این کار را انجام می دهد. این دستور در قسمت بعد بیشتر شرح داده می شود. دستور grid on خطوط شبکه ای را روی نمودار فعال می کند و grid off آنها را حذف می کند. اگر بخواهیم متنی را روی نمودار قرار دهیم از تابع text استفاده می کنیم. پنجره clf را پاک می کند. clc نیز پنجره command window را پاک می کند.

حال در مثال زیر روش استفاده از این دستورات را مشاهده می کنید.

```
>> clf
>> plot(x,y,'-o')
>> xlabel(' variable ')
>> ylabel(' y variable ')
>> title(' MATLAB ')
>> legend(' sin ')
>> grid
>> text(3,.5,' Mim Shimi 82')
```



دستور grid در صورتی که به تنهایی به کار رود، در صورتی که شبکه ها روشن باشد آن را خاموش و در صورت خاموش بودن آن را روشن می کند.

در دستور text آرگومان اول و دوم مختصات ابتدا متن و آرگومان سوم متن مورد نظر است. در صورتی که مختصات متن را ندانیم می توانیم از دستور gtext استفاده کنیم. با اجرا این دستور خطوط متقاطع روی صفحه نمایش داده می شود و مکان مورد نظر با کلیک ماوس تعیین می شود. حالت کلی این دستور به این شکل است که TEXT متن مورد نظر است:

gtext ('TEXT')

همان طور که مشاهده فرمودید تقریباً در تمام دستورات فوق از رشته های کرکتری استفاده می شود. MATLAB علاوه بر رشته های معمولی امکاناتی دارد تا بتوان متن هایی شامل کرکترهای ویژه (مثل ∞ و \neq) و در چند خط، همچنین عبارات توان دار و اندیس دار را به نمودارها اضافه کرد.

اضافه کردن کرکترهای ویژه به راحتی انجام می گیرد. با قرار دادن یکی از موارد زیر در رشته میتوان آن را به متن اضافه کرد.

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	α	\upsilon	υ	\sim	\sim
\beta	β	\phi	ϕ	\leq	\leq
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	\clubsuit
\epsilon	ϵ	\omega	ω	\diamondsuit	\diamondsuit
\zeta	ζ	\Gamma	Γ	\heartsuit	\heartsuit
\eta	η	\Delta	Δ	\spadesuit	\spadesuit
\theta	θ	\Theta	Θ	\leftrightarrow	\leftrightarrow
\vartheta	ϑ	\Lambda	Λ	\leftarrow	\leftarrow
\iota	ι	\Xi	Ξ	\uparrow	\uparrow
\kappa	κ	\Pi	Π	\rightarrow	\rightarrow
\lambda	λ	\Sigma	Σ	\downarrow	\downarrow
\mu	μ	\Upsilon	Υ	\circ	\circ
\nu	ν	\Phi	Φ	\pm	\pm
\xi	ξ	\Psi	Ψ	\geq	\geq
\pi	π	\Omega	Ω	\propto	\propto
\rho	ρ	\forall	\forall	\partial	∂
\sigma	σ	\exists	\exists	\bullet	\bullet
\varsigma	ς	\ni	\ni	\div	\div

<code>\tau</code>	τ	<code>\cong</code>	\cong	<code>\neq</code>	\neq
<code>\equiv</code>	\equiv	<code>\approx</code>	\approx	<code>\aleph</code>	\aleph
<code>\Im</code>	\Im	<code>\Re</code>	\Re	<code>\wp</code>	\wp
<code>\otimes</code>	\otimes	<code>\oplus</code>	\oplus	<code>\oslash</code>	\oslash
<code>\cap</code>	\cap	<code>\cup</code>	\cup	<code>\supseteq</code>	\supseteq
<code>\supset</code>	\supset	<code>\subseteq</code>	\subseteq	<code>\subset</code>	\subset
<code>\int</code>	\int	<code>\in</code>	\in	<code>\circ</code>	\circ
<code>\rfloor</code>	\rfloor	<code>\lceil</code>	\lceil	<code>\nabla</code>	∇
<code>\lfloor</code>	\lfloor	<code>\cdot</code>	\cdot	<code>\ldots</code>	\dots
<code>\perp</code>	\perp	<code>\neg</code>	\neg	<code>\prime</code>	\prime
<code>\wedge</code>	\wedge	<code>\times</code>	\times	<code>\emptyset</code>	\emptyset
<code>\rceil</code>	\rceil	<code>\surd</code>	\surd	<code>\mid</code>	\mid
<code>\vee</code>	\vee	<code>\varpi</code>	ϖ	<code>\copyright</code>	\copyright
<code>\langle</code>	\langle	<code>\rangle</code>	\rangle		

برای ایجاد متن‌های چند خطی می‌توانید از آرایه‌های رشته‌ای به صورت زیر استفاده کنید.

`text ({'LINE1' , 'LINE2' })`

برای قرار دادن توان بر روی یک عبارت از علامت توان بعد از عبارت استفاده می‌شود. در صورتی که عبارتی که در توان قرار می‌گیرد بیش از یک کرکتر باشد آن را بین دو { } قرار می‌دهیم. و برای ایجاد اندیس از کرکتر '_' استفاده می‌کنیم.

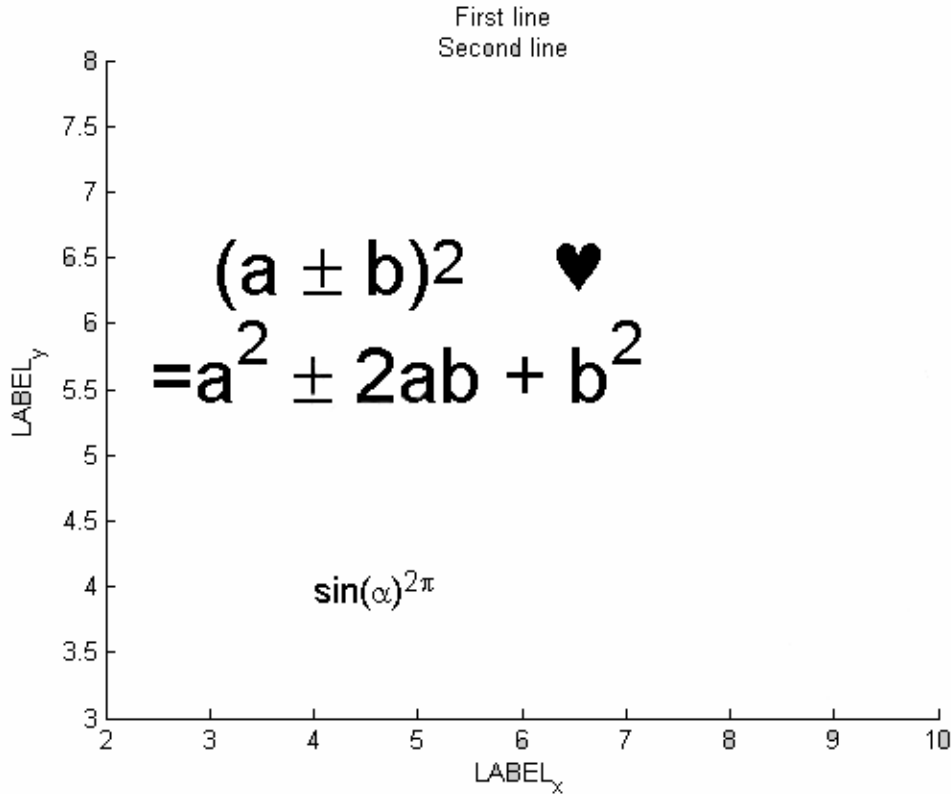
حتی با استفاده از دستور `\fontsize` می‌توان اندازه متن را نیز مشخص کرد.

همچنین ممکن است تنها نمایش قسمتی از نمودار برای ما مهم باشد. دستور `axis` با مشخص کردن حدود محورها این کار را انجام می‌دهد. همان‌طور که در زیر می‌بینید آرگومان ورودی دستور شامل یک بردار که مشخص‌کننده حدود محورها است می‌باشد.

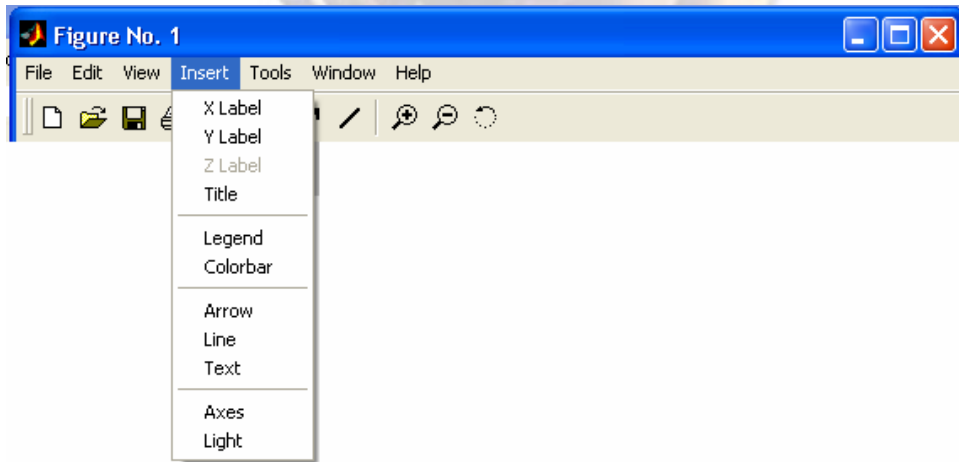
`axis ([XMIN XMAX YMIN YMAX])`

در مثال زیر با روش استفاده از دستورات اخیر آشنا می‌شوید.

```
>> axis ([2 10 3 8])
>> xlabel ('LABEL_x')
>> ylabel ('LABEL_y')
>> title ('First line','Second line')
>> text (6,6.5,'\fontsize{34} \heartsuit')
>> gtext ('\fontsize{30} (a \pm b)^2','=a^2 \pm 2ab + b^2')
>> text (4,4,'\fontsize{14}sin(\alpha)^{2\pi}')
```



بسیاری از دستوراتی که در بالا توضیح داده شد بدون تایپ در پنجره command و از طریق منوی insert پنجره figure قابل دسترسی هستند. اگر احتیاجی به یاد گرفتن دستورات بالا نمی بینید می توانید به این طریق عمل کنید.



تا به حال نمودارهایی را رسم کردیم که محورهای مختصات آنها به صورت خطی تقسیم بندی شده بود؛ ولی در برخی از مواقع لازم است که یک یا هر دو محور را با تقسیمات لگاریتمی نمایش دهیم. برای این کار نیز دستوراتی وجود دارد.

از تابع semilogx برای نموداری که محور x آن برحسب مقدار لگاریتمی تقسیم بندی شده و از تابع semilogy برای نموداری با محور y لگاریتمی استفاده کنید. همچنین تابع loglog نموداری رسم می کند که هر دو محور آن لگاریتمی است. آرگومان های ورودی این توابع مانند تابع plot میباشد.

نمودارهای متعدد:

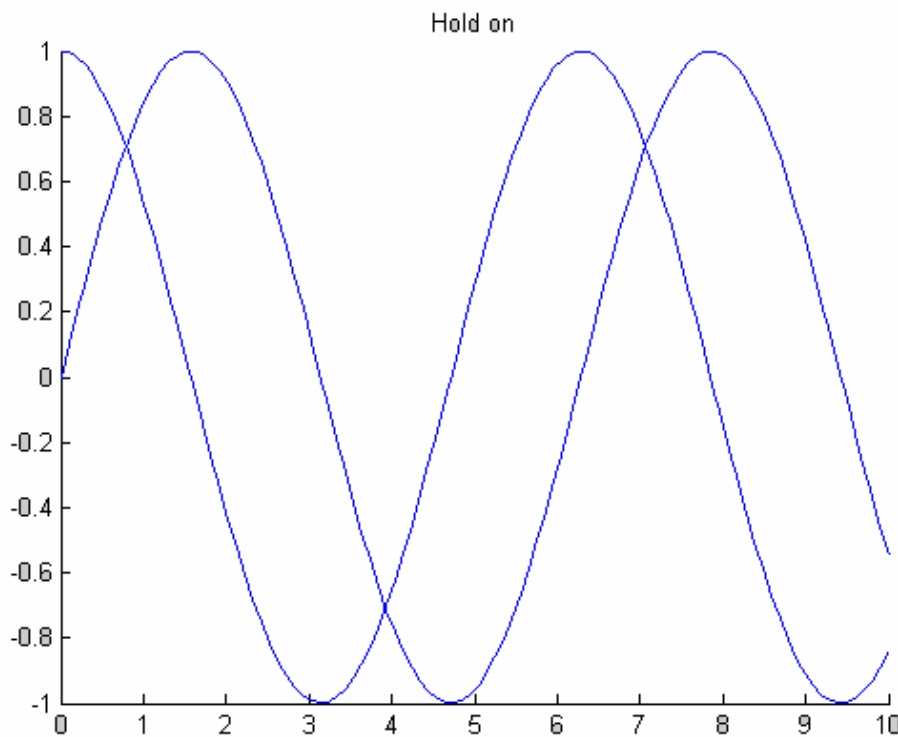
تا اینجا دستورات متنوعی برای رسم نمودار آموختیم. ولی امکان دارد که بخواهیم چند نمودار را همزمان بخواهیم و این خواسته با توجه با این که پنجره figure با رسم نمودار جدید پاک می شود و نمودار جدید جایگزین قبلی می شود به روش معمولی امکان پذیر نیست.

در زیر چند روش را برای این کار بیان می کنیم.

« روش اول:

در این روش از دستور hold استفاده می شود. این دستور محتویات پنجره figure را نگه داشته و نمودار جدید را روی نمودار قبلی رسم می کند. در این روش با توجه به این که نمودارها به یک رنگ رسم می شوند بهتر است رنگ و نوع خط نمودار توسط کاربر مشخص شود.

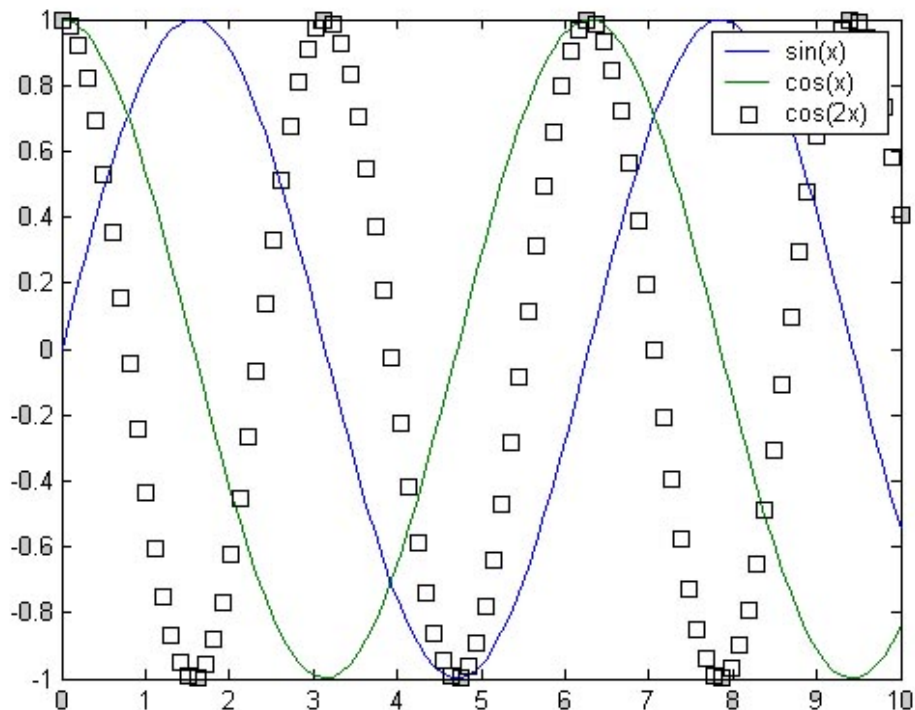
```
>> x=linspace(0,10);  
>> y=sin(x);z=tan(x);s=cos(x);t=x.^2;  
>> hold on  
>> plot(x,y)  
>> plot(x,s)  
>> title('Hold on')  
>> hold off
```



« روش دوم:

در این روش از تابع plot استفاده می شود. در این تابع می توان بعد از جفت آرگومان اول، جفت آرگومان مربوط به نمودار بعدی را به عنوان آرگومان های بعدی وارد کرد. به این ترتیب این تابع می تواند بیشمار آرگومان ورودی داشته باشد. MATLAB این نمودارها را با رنگ های مختلف رسم می کند. در صورتی که بخواهیم نوع خط و ... را مشخص کنیم باید بعد از هر جفت آرگومان این کار را انجام دهید.

```
>> plot(x,y,x,s,x,cos(2*x),'sk')  
>> legend('sin(x)', 'cos(x)', 'cos(2x)')
```



« روش سوم:

در این روش دو نمودار با محور X مشترک و محور Y مختص به خود که تقسیم بندی متفاوتی دارند رسم می شود. این کار توسط تابع plotyy انجام می گیرد. این تابع حداکثر دو نمودار را رسم می کند، به این ترتیب دارای دو جفت آرگومان ورودی است. حالت کلی آن را در زیر می بینید:

```
plotyy(x1,y1,x2,y2,' fun1' , ' fun2')
```

دو آرگومان آخر مشخص کننده نوع محورهای مختصات برای نمودار اول و دوم می باشد؛ و می تواند یکی از موارد زیر باشد.

semilogx, semilogy, plot, loglog, stem

در این تابع نمی توان به سادگی تابع plot نوع خطوط و ... را مشخص کرد. برای این کار باید از اشاره گرها یا روش های دیگر استفاده کرد. به دلیل گستردگی این بحث تنها به یک مثال قناعت می کنیم.

« روش چهارم:

در این روش از پنجره های متعدد استفاده می شود. به این طریق که قبل از هر تابع رسم نمودار از دستور figure(n) استفاده می کنیم که n مشخص کننده شماره پنجره است که برای فراخوانی پنجره از آن استفاده می شود. این دستور پنجره figure جدیدی را باز کرده و نمودار را در این پنجره رسم می کند. دستورات زیر را تایپ کنید و نتیجه آن را مشاهده کنید.

```
>> figure(1)
>> x=linspace(0,10);
>> y=sin(x);s=cos(x);
>> plot(x,y)
>> figure(2)
>> plot(x,s)
```

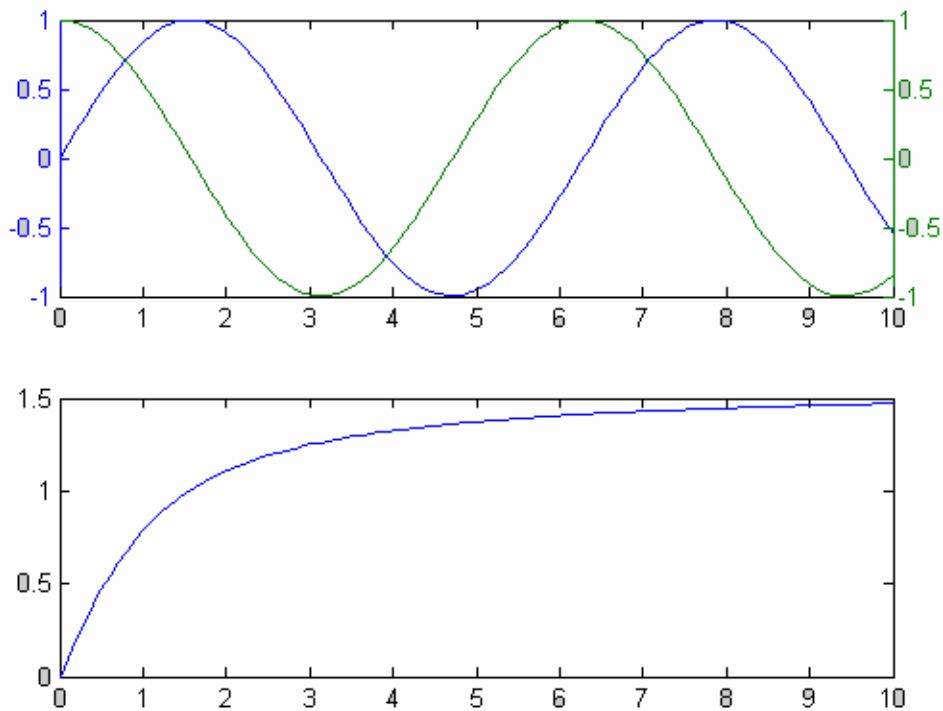
« روش پنجم:

در این روش پنجره را به چند قسمت تقسیم کرده و هر نمودار را در یکی از این قسمت ها رسم می کنیم. این تقسیم توسط دستور subplot انجام می شود. حالت کلی این دستور به صورت زیر است:

subplot (m,n,p) or subplot(mnp)

این دستور پنجره figure را به یک ماتریس $m \times n$ تقسیم می کند و p امین خانه آن را انتخاب می کند. شماره هر خانه به صورت ردیفی تعیین می شود.

```
>> subplot (2,1,1)
>> plotyy (x,y,x,s)
>> subplot (2,1,2)
>> plot (x,atan(x))
```



« روش ششم:

در این روش نمودار جدید با محورهای جدید و مقیاس متفاوت روی نمودار قبلی قرار می گیرد. تابع مورد استفاده در روش تابع axes می باشد. حالت کلی آن به صورت زیر است:

axes (' position' , [left, bottom, width, height])

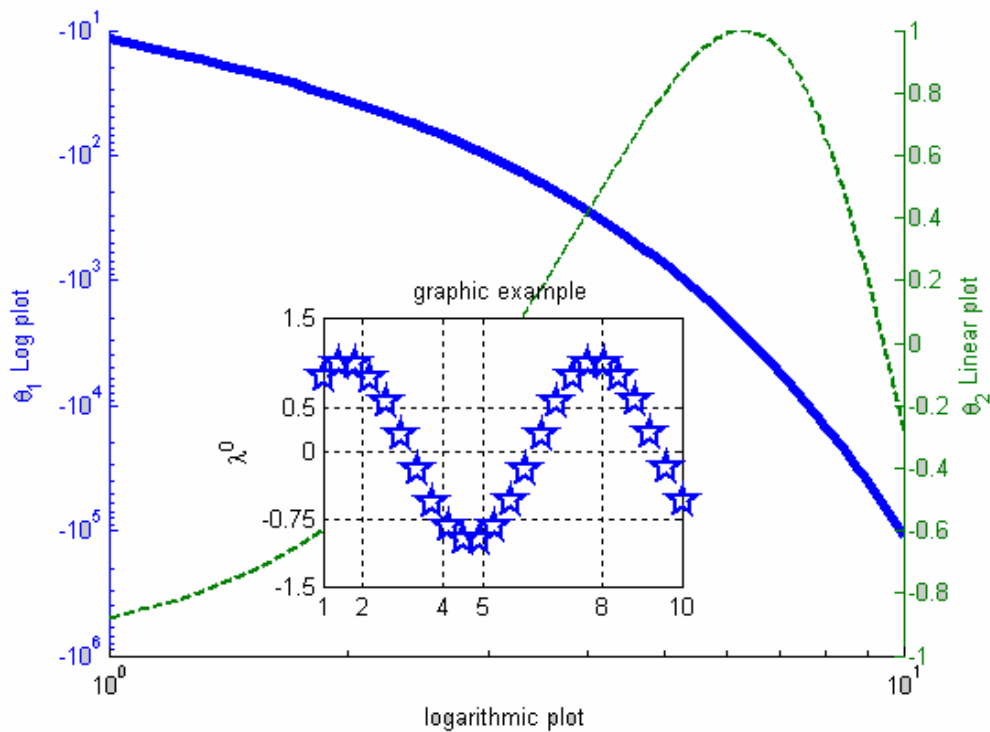
این دستور دارای دو آرگومان ورودی است؛ آرگومان اول یک رشته کارکتری به صورت بالا و آرگومان بعدی یک بردار است. دو عنصر اول بردار مشخص کننده مکان نمودار جدید و دو عنصر بعدی مشخص کننده اندازه آن است.

این دستور مختصات (0,0) را برای گوشه پایین سمت چپ و (1,1) را برای گوشه بالا سمت راست در نظر می گیرد.

مثال زیر نمونه ای از یک M-file است که تقریباً در بر گیرنده تمام دستوراتی است که در این بخش توضیح داده شد. به تنظیماتی که می توان روی نمودارها اعمال کرد دقت کنید و سعی کنید دستور مربوط به هر یک را پیدا کنید.

```
x=1:1:10;
y=-10*sinh(x);
z=-cos(x/2);
[AX,H1,H2]=plotyy(x,y,x,z,'loglog','semilogx')
set(get(AX(1),'Ylabel'),'String','\theta_1 Log plot');
set(get(AX(2),'Ylabel'),'String','\theta_2 Linear plot');
set(get(AX(2),'Xlabel'),'String','logarithmic plot');
set(H1,'LineWidth',4);
set(H2,'LineStyle','--','LineWidth',2);

axes('Position',[.34 .2 .35 .35]);
x1=linspace(1,10,24);
y1=sin(x1);
f=plot(x1,y1,'p');
axis([1 10 -1.5 1.5]);
set(f,'MarkerSize',12,'LineWidth',2);
ylabel('\lambda^0');
title('graphic example');
set(gca,'XTick',[1 2 4 5 8 10],'YTick',[-1.5 -.75 0 .5 1.5]);
grid on;
```



در این بخش دستوراتی را معرفی می‌کنیم که در نوشتن برنامه‌های مختلف به ما کمک می‌کند. این دستورات در اغلب زبان‌های برنامه‌نویسی مشترک است و تنها گرامر آن‌ها متفاوت است.

ساخت function file:

تا به حال تنها از توابعی استفاده می‌کردیم که قبلاً برای MATLAB تعریف شده بود؛ ولی ممکن است این توابع نتوانند نیازهای ما را پاسخ دهند، یا بخواهیم توابعی با کاربری خاص بنویسیم.

یک تابع (function file) مانند یک M-file است با این تفاوت که خط اول آن به صورت زیر است:

```
function [outputs]= name(inputs)
```

این خط مشخص می‌کند که این M-file یک تابع است. همچنین تعداد ورودی‌ها و خروجی‌ها را مشخص کرده و هر یک را در یک متغیر قرار می‌دهد. در صورتی که تنها یک ورودی داشته باشیم نیازی به کلمه ([]) نیست. name نیز نام تابع را مشخص می‌کند.

بهتر است برای خواناتر شدن برنامه از عبارات توضیحی استفاده کنیم. این عبارات باید ورودی‌ها و خروجی‌ها را مشخص کند. همچنین می‌توان نام برنامه نویس و تاریخ نوشتن آن را نیز مشخص کرد. این خطوط با اجرا دستور `>>help name` به نمایش درمی‌آیند. به عنوان مثال M-file زیر دستور `prod` را شبیه‌سازی می‌کند.

```
function p=prod2(x)
% function p=PRODY(x)
% shabih sazi farman PROD
% a:radif b:sotoon
[a,b]=size2(x);
p(1,:)=x(1,:);
for i=2:a,
%   satr aval p dar satr haye x zarb shode
%   va dar satr aval p zakhir mishavad
    p(1,:)=p(1,:).*x(i,:);
end
% if x is rowvector
while a==1;
    p=1;
    for i=1:b,
        p=p*x(i);
    end
    a=0;
end
```

حلقه‌های تکرار:

این دستورات در اغلب زبان‌های برنامه‌نویسی به خصوص C وجود دارند.

« حلقه For

این حلقه این امکان را به وجود می‌آورد که تعدادی از دستورات به تعداد دفعات از قبل تعیین شده تکرار شوند. شکل کلی آن به صورت زیر است:

```
for variable = a
    statement 1
    statement 2
    ...
end
```

که a يك ماتريس است. در هر بار تکرار حلقه يك ستون ماتريس a در variable قرار مي گيرد. به اين ترتيب حلقه به تعداد ستون هاي a تکرار مي شود. اين حلقه را مي توان به صورت تو در تو استفاده کرد. مثال زير با استفاده از حلقه هاي تو در تو جدول ضرب ايجاد مي کند.

```
for i=۱:۵
  for j=۱:۵
    s(i,j)=i*j;
  end
end
```

« حلقه While »

اين حلقه چند دستور را به تعداد دفعات نامحدود تکرار مي کند. از اين دستور هنگامی استفاده مي شود که تعداد دفعات تکرار مشخص نباشد. شکل کلي اين دستور به صورت زير است:

```
while expression
  statements
end
```

expression يك عبارت شرطي است و تا هنگامی که درست باشد، حلقه تکرار مي شود. (عبارات شرطي در قسمت بعد شرح داده مي شود)

عملگرهاي رابطه اي:

اين عملگرها شامل موارد زير مي باشد:

شرح	عملگرهاي رابطه اي
کوچکتر از	<
کوچکتر يا مساوي	<=
بزرگتر	>
بزرگتر يا مساوي	>=
مساوي يا	==
مخالف يا (نامساوي)	~=

عملگرهاي منطقي:

اين عملگرها را در جدول زير مشاهده مي کنيد

شرح	عملگر منطقي
AND	&
OR	
NOT	~
OR انحصاري (در صورتی که تنها یکی (x یا y) مقدار درستی داشته باشند مقدار True برمی گرداند)	xor(x,y)

ساختارهاي تصميم:

« شرط If - Else – End »

حتما با عملکرد اين دستور در زبان هاي برنامه نويسي ديگر آشنا شده ايد. شکل کلي اين دستور را در زير مي بينيد.

```

if expression ۱
  statements ۱
elseif expression ۲
  statements ۲
...
elseif expression n
  statements n
else
  statements
end

```

همان طور که مشاهده می کنید در حالت کلی می توان از يك If، بیشمار Elseif و يك Else و End استفاده کرد. استفاده از Else و Elseif اختیاری است.

اگر شرط مقابل If درست باشد دستورات شماره ۱ اجرا می شوند، در غیر این صورت شرط ۲ (مقابل Elseif) بررسی می شود در صورتی که درست باشد دستورات ۲ و در غیر این صورت شرط ۳ بررسی می شود ... در صورتی که n شرط بررسی شد و درست نبود دستورات قسمت Else اجرا می شود.

« شرط Switch-Case

از این ساختار برای تصمیم گیری چندگانه بر اساس مقادیر مختلف يك عبارت استفاده می شود. به طوری کلی در تمام تصمیم گیری ها که بیش از ۳ انتخاب وجود داشته باشد از این دستور استفاده می شود.

به عنوان مثال فرض کنید متغیری مثل X مقادیر ۱، ۲، ۳ را اختیار می کند و می خواهید بر اساس مقادیر مختلف X تصمیم گیری مختلفی را انجام دهید. اگر برابر ۱ بود دستورات ۱، اگر برابر ۲ بود دستورات ۲ و اگر برابر ۳ بود دستورات ۳ اجرا شوند و در صورتی که هیچ کدام از این ها نبود دستورات ۴ (otherwise) اجرا شوند. حالت کلی این دستور را مشاهده می کنید:

```

switch switch_expr
case case_expr,
  statement, ..., statement
case {case_expr۱, case_expr۲, case_expr۳,...}
  statement, ..., statement
...
otherwise,
  statement, ..., statement
end

```

به چند نکته در این مورد باید دقت کرد:

- ۱) پس از اجرای هر يك از دستورات روند اجرا برنامه به بعد از End منتقل می شود و سایر Case ها کنترل نمی شوند.
- ۲) در بالا در مورد Case دوم در صورتی که عبارت مورد نظر با هر يك ۳ عبارت داخل کلوشه ({}) برابر باشد دستورات اجرا می شوند.
- ۳) استفاده از Otherwise نیز اختیاری است.

بلوك Try-Catch:

شکل کلی این دستور به این صورت می باشد:

```
try
    commands
catch
    commands
end
```

عملکرد این دستور به این صورت است که دستورات زیر Try اجرا می شوند؛ در صورتی که خطایی رخ دهد کنترل برنامه به Catch منتقل شده و دستورات موجود در این قسمت اجرا می شود. این خاصیت باعث می شود از آن برای خطایابی برنامه ها استفاده شود.

توقف روند اجرای برنامه:

« Break

هنگامی که این دستور اجرا می شود MATLAB به اولین دستور که بعد از حلقه For قرار دارد می رود. در صورتی که این دستور در حلقه های تو در تو (For یا While) به کار رود MATLAB فقط از حلقه جاری خارج می شود.

« Error

این دستور باعث توقف اجرا برنامه شده و می تواند یک رشته کرکتری را برگرداند.

```
error (' STATEMENT ')
```

« Return

هر گاه روند اجرا برنامه به این دستور برسد مقدار مورد نظر را برمی گرداند (در Command window نمایش می دهد)؛ و ادامه اجرای برنامه متوقف می شود.

از این دستور برای نمایش زود هنگام مقادیر یعنی قبل از به پایان رسیدن کامل برنامه استفاده می شود. به این ترتیب هرگاه جواب مورد نظر به دست آمد روند اجرای برنامه نیز متوقف می شود و مقدار مورد نظر را برمی گرداند.