فصل اول

ویژگیهای اصلی MATLAB

کلاس آموزشی

فصل اول: ویژگیهای اصلی MATLAB

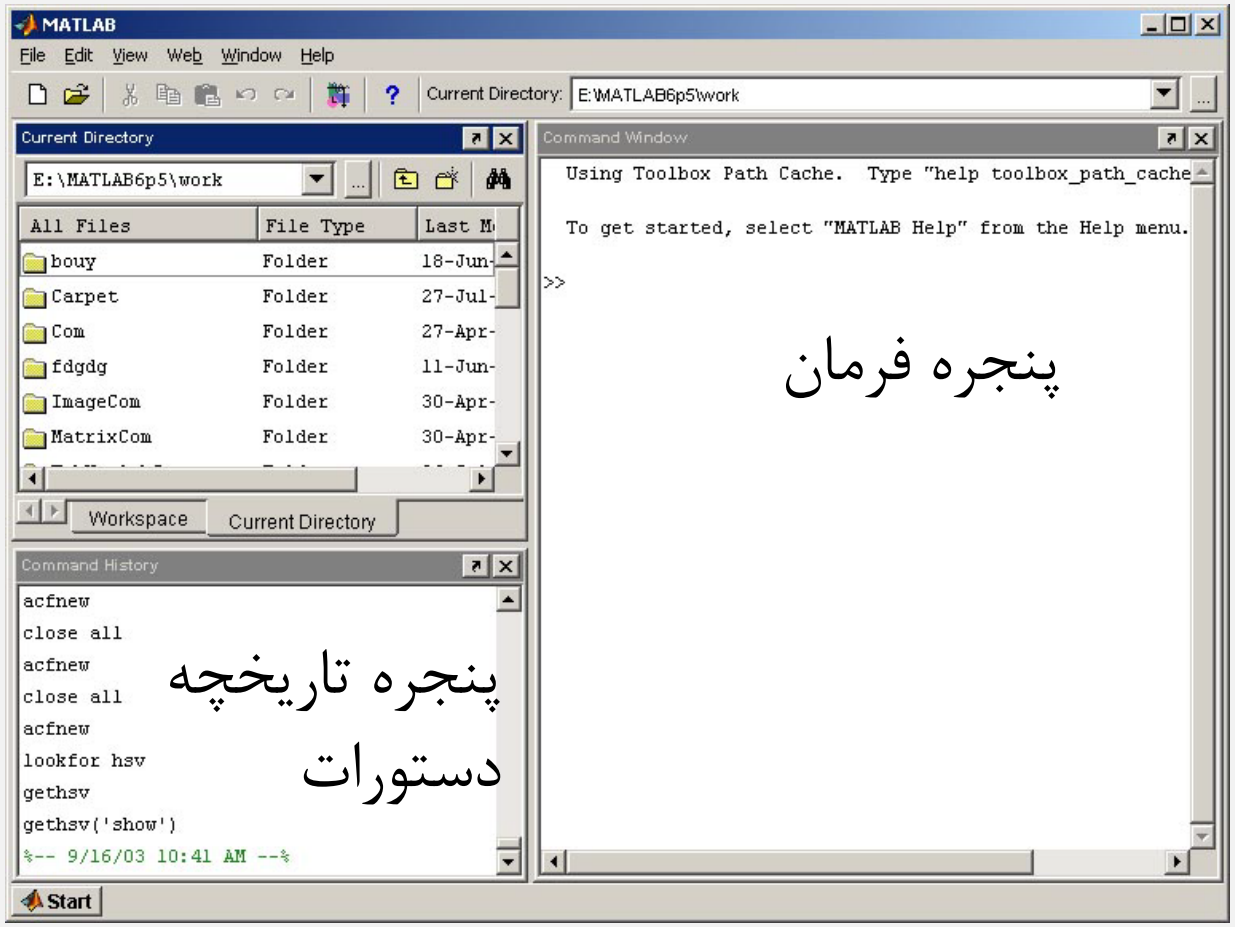
- آشنایی با محیط متلب
- عملیات ریاضی ساده
- عملگرهای ریاضی متلب
- فضای کاری متلب (Workspace)
- فرمت نمایش اعداد
- انواع متغیرها
- نامگذاری متغیرها
- متغیرهای ویژه
- علائم نقطه گذاری و جملات توضیحی
- اعداد مختلط
- بعضی از توابع ریاضی در متلب
- راهنمای متلب
- فایلهای متنی یا m-فایلها
- مدیریت فایل در متلب

فصل اول: ویژگیهای اصلی MATLAB

۱-۱- آشنایی با محیط متلب

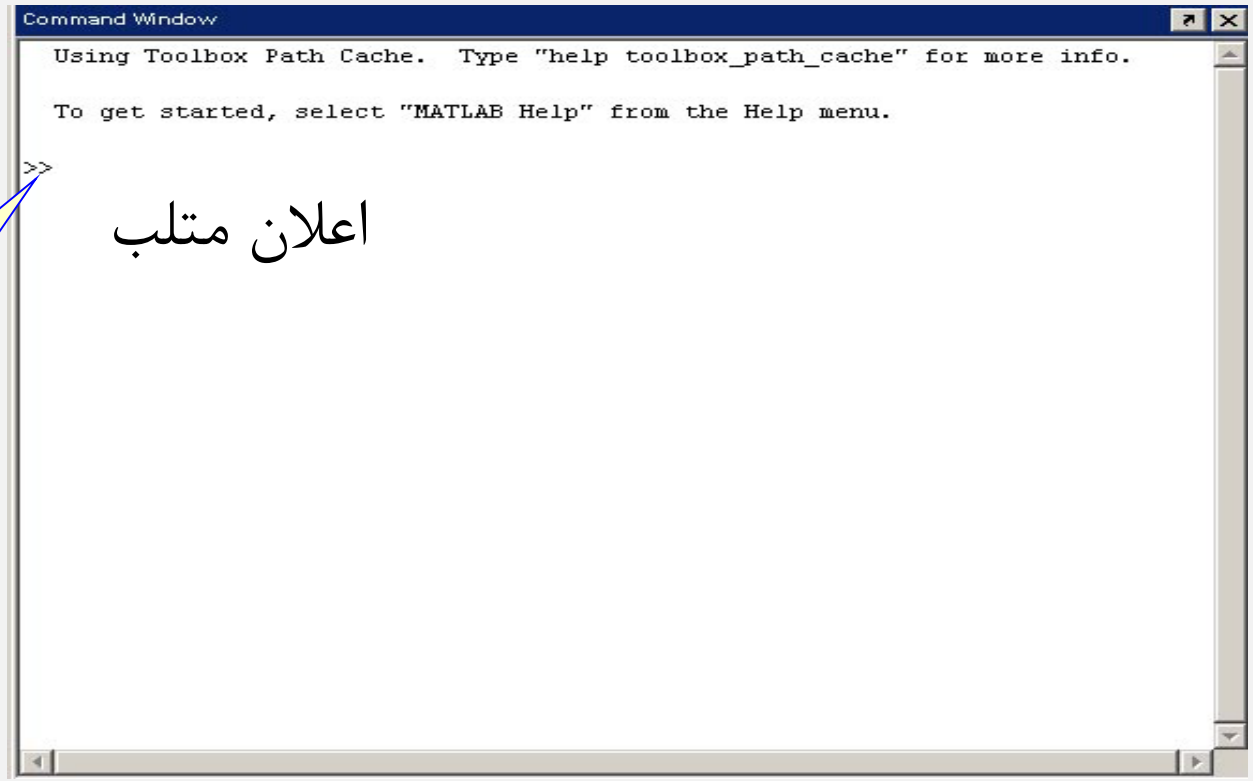
- پنجره فرمان : Command window
- پنجره تاریخچه دستورات: Command History
- پنجره دایرکتوری جاری : Current Directory
- پنجره فضای کاری : Work Space
- دایرکتوری جاری
- منوی Start

فصل اول: ويژگيهاي اصلي MATLAB



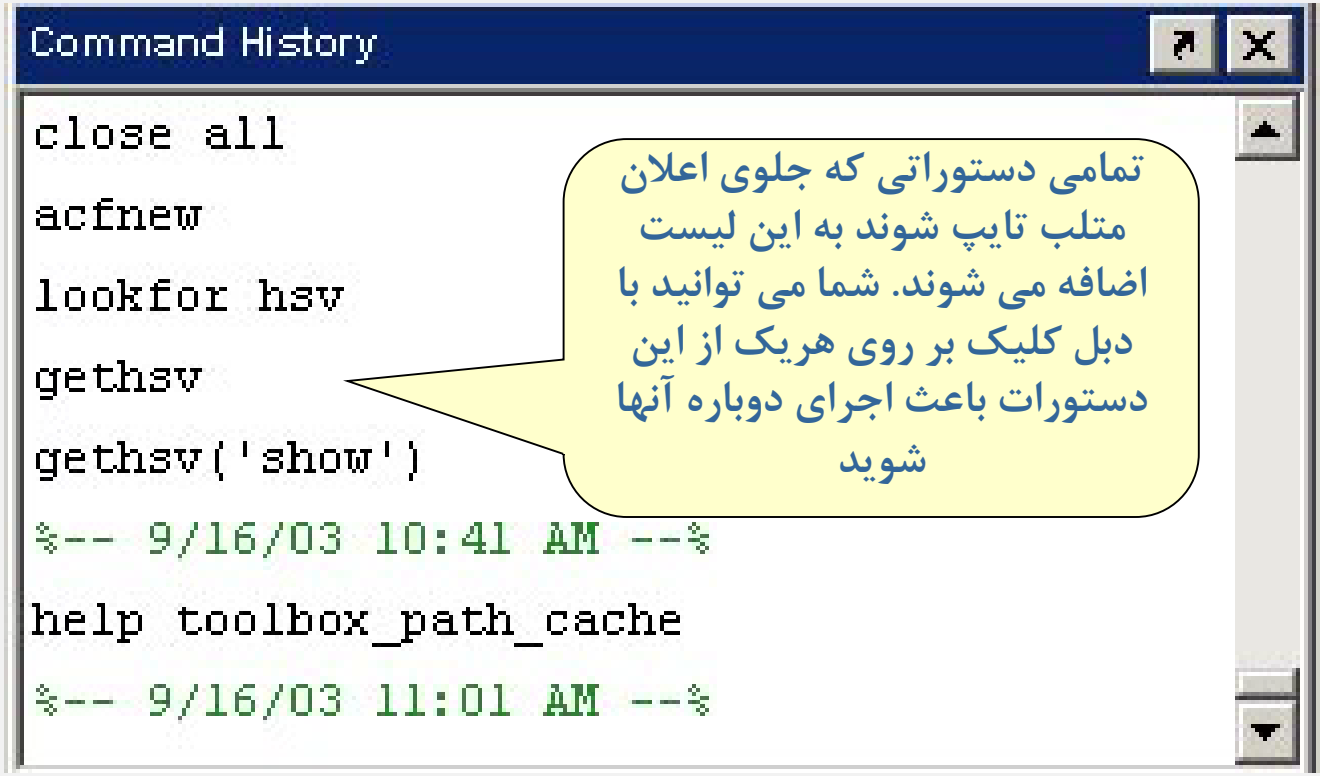
فصل اول: ويژگيهاي اصلي MATLAB

پنجره فرمان : Command Window



فصل اول: ویژگیهای اصلی MATLAB

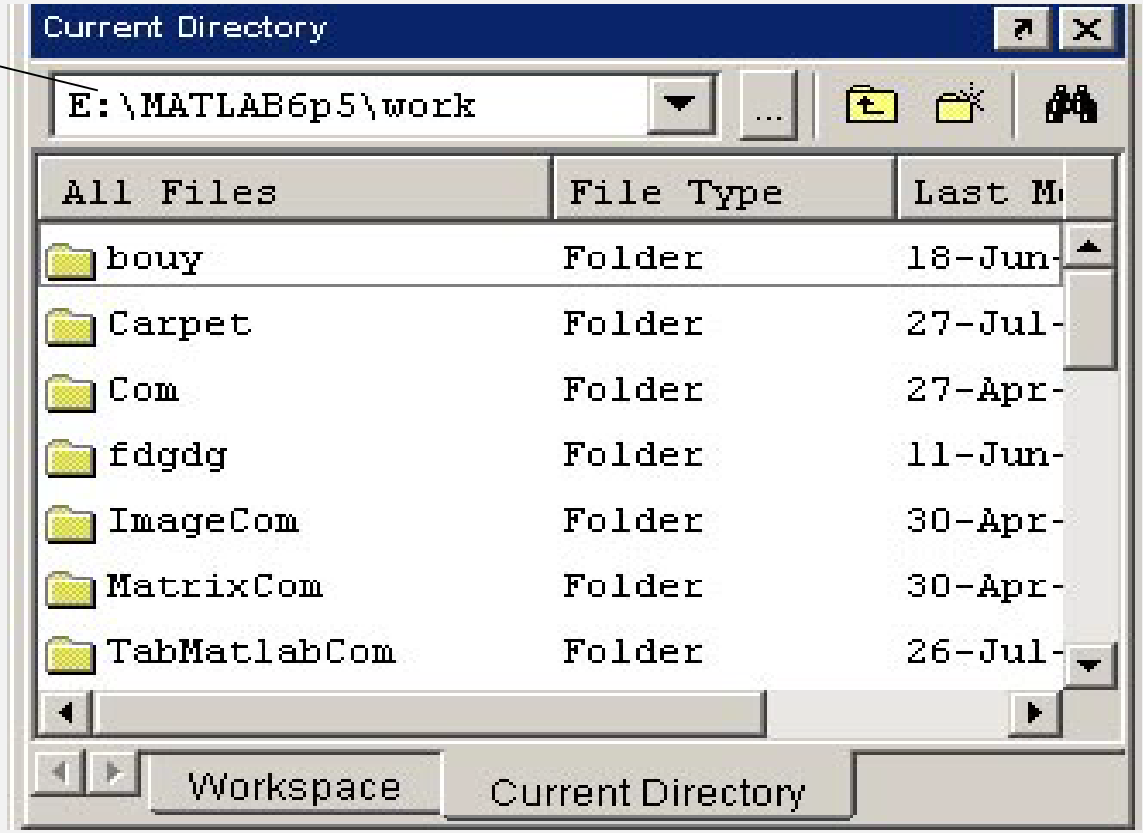
پنجره تاریخچه دستورات: Command History



فصل اول: ویژگیهای اصلی MATLAB

پنجره دایرکتوری جاری : Current Directory

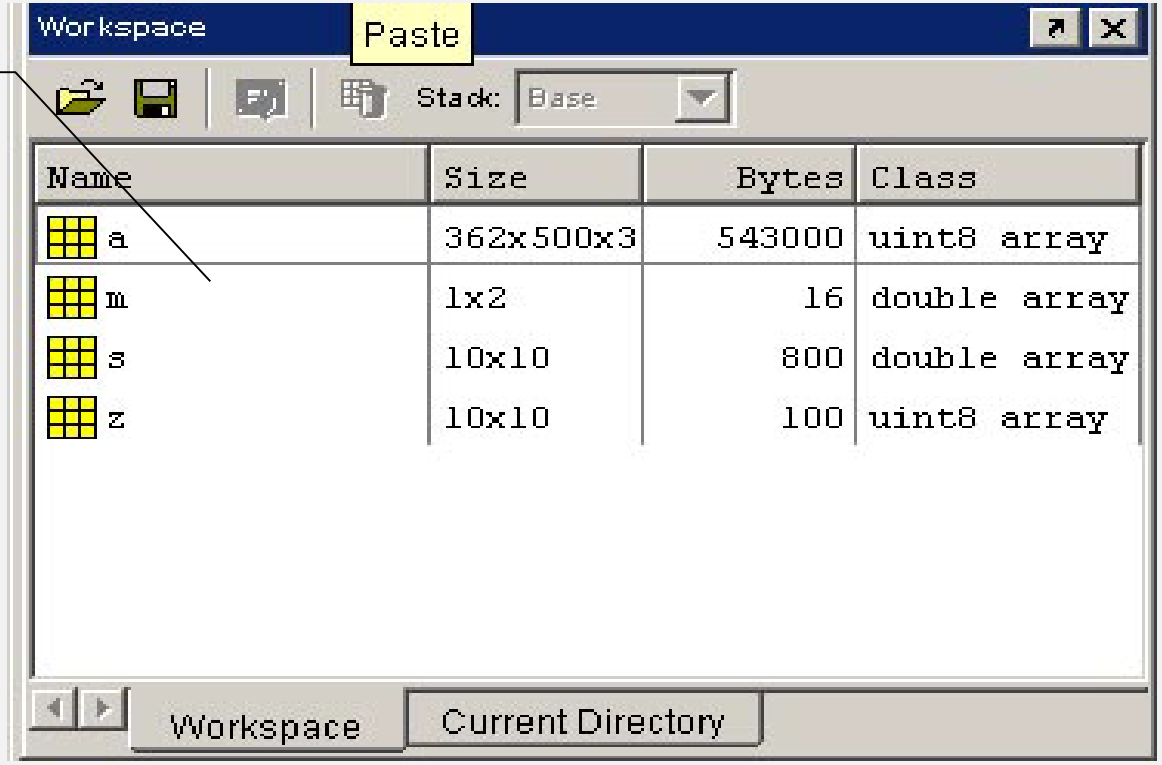
در هر زمان تنها یک دایرکتوری یا پوشه به عنوان دایرکتوری جاری در متلب شناخته می شود. هر فایل متلب(برنامه نوشته شده توسط شما) که نام آن جلوی اعلان متلب تایپ شود تنها درصورتی اجرا می شود که در دایرکتوری جاری یا در مسیر متلب باشد



فصل اول: ویژگیهای اصلی MATLAB

فضای کاری : Work Space

متغیرهایی که در حال حاضر در محیط کاری متلب وجود دارند و شما می توانید از مقادیر آنها استفاده کنید یا آنها را تغییر دهید



فصل اول: ویژگیهای اصلی MATLAB

۱-۲-عملیات ریاضی ساده

مثال: محاسبه یک عبارت:

راه اول:

```
>> 4*25 + 6*22 + 2*99 ↵
   ans=
            430
```

فصل اول: ویژگیهاي اصلي MATLAB

۱-۲-عملیات ریاضی ساده

مثال: محاسبه یک عبارت:

راه دوم:

```
>>a=25;
>>b=22; c=99;
>>d=4*a+6*b+2*c
  d=
     430
>>
```

نکته ۱: علائم ; و ,

نکته ۲: تعریف متغیرها

نکته ۳: متغیرهای ویژه

فصل اول: ویژگیهای اصلی MATLAB

۱-۳-عملگرهای ریاضی متلب:

\/ , * , - , + , ^

مثال:

```
>>5^2
 ans=
       25
```

/ و \ هر دو عملگر تقسیم میباشند. / تقسیم از چپ و \ تقسیم از راست است. مثلا حاصل 56/8 و 8\56 یکسان است.

- ترتیب حق تقدم: - + > * \ / > ^

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-فضای کاری متلب Work space

متغیرهایی که در محیط متلب ایجاد می شوند در بخشی از حافظه بنام محیط کاری متلب ذخیره می گردند. فضای کاری برنامه های اسکریپت متلب با فضای کاری متلب یکسان است. یعنی اگر متغیری در محیط متلب تعریف شده باشد در یک برنامه اسکریپت می توان از آن استفاده کرد و برعکس. اما برنامه های تابعی متلب دارای فضای کاری مختص به خود هستند و متغیرهای آنها در فضای کاری متلب وارد نمی شود.

- در مورد انواع برنامه های متلب در فصلهای آتی توضیح داده خواهد شد.

فصل اول: ویژگیهاي اصلي MATLAB

۱-۴-فضای کاری متلب Work space

نکاتی در مورد فضای کاری متلب:

- زمان اعتبار متغیرها:
- دستور who و whos
- ذخیره و بازیابی متغیرها: دستورات save و load

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۱-زمان اعتبار متغیرها

متغیرهایی که در فضای کاری تعریف می شوند تنها در دو حالت زیر از حافظه پاک خواهند شد:

- خروج متلب
- استفاده از دستور clear :

تمامی متغیرها از حافظه پاک می شوند >> clear

تنها متغیرهای نامبرده شده از حافظه پاک می شوند >> clear a b c

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۲-دستورات who و whos

با استفاده از این دو دستور می توان اسامی(و مشخصات) متغیرهای موجود در فضای کاری را بدست آورد.

```
>> who
Your variables are:
      a b c

>> whos
  Name      Size                Bytes  Class

  a         1x1                     8  double array
  b         1x1                     8  double array
  c         1x1                     8  double array
```

یادآوری: پنجره workspace نیز مشخصات متغیرهای موجود در فضای کاری را مانند دستور whos نشان می دهد.

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۳- ذخیره و بازیابی متغیرها: دستورات save و load:

در صورتیکه بخواهیم پس از خروج از محیط متلب همه یا بعضی از متغیرهای موجود در فضای کاری برای استفاده های بعدی ذخیره گردند از دستور save استفاده می کنیم. با دستور load می توان متغیرهای ذخیره شده را به فضای کاری بازگرداند.

مثال:

```
>>a=5; b=4; c=7;
>>save c:\myfile.mat a c;
>>clear  همه متغیرها پاک می شوند
>>a
??? Undefined function or variable 'a‘
>> load c:\myfile.mat
>>a
    a=
        5
>>b
??? Undefined function or variable 'a‘
```

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۳- ذخیره و بازیابی متغیرها: دستورات save و load:

فرم کلی کاربرد دستورات save و load بصورت زیر است:

save [filename] [variables]

Load [filename] [variables]

درصورتیکه اسم فایل نوشته نشود. فایل پیش فرض matlab.mat مورد استفاده قرار خواهد گرفت و درصورتیکه نام متغیرها نوشته نشود تمامی متغیرهای موجود در فضای کاری ذخیره و یا تمامی متغیرهای ذخیره شده در فایل بازیابی میشوند.

فصل اول: ویژگیهای اصلی MATLAB

۱-۵-فرمت نمایش اعداد (دستور Format)

با استفاده از این دستور می توان نحوه نمایش اعداد در پنجره فرمان متلب را تغییر داد.

```
>>Format [option]
```

Option: short, long, short e, long e, short g, long g, hex,
+ , ...

دقت کنید که این دستور دقت محاسبات را تغییر نمی دهد و تنها بر نحوه نمایش اعداد تاثیر خواهد گذاشت.

فصل اول: ویژگیهای اصلی MATLAB

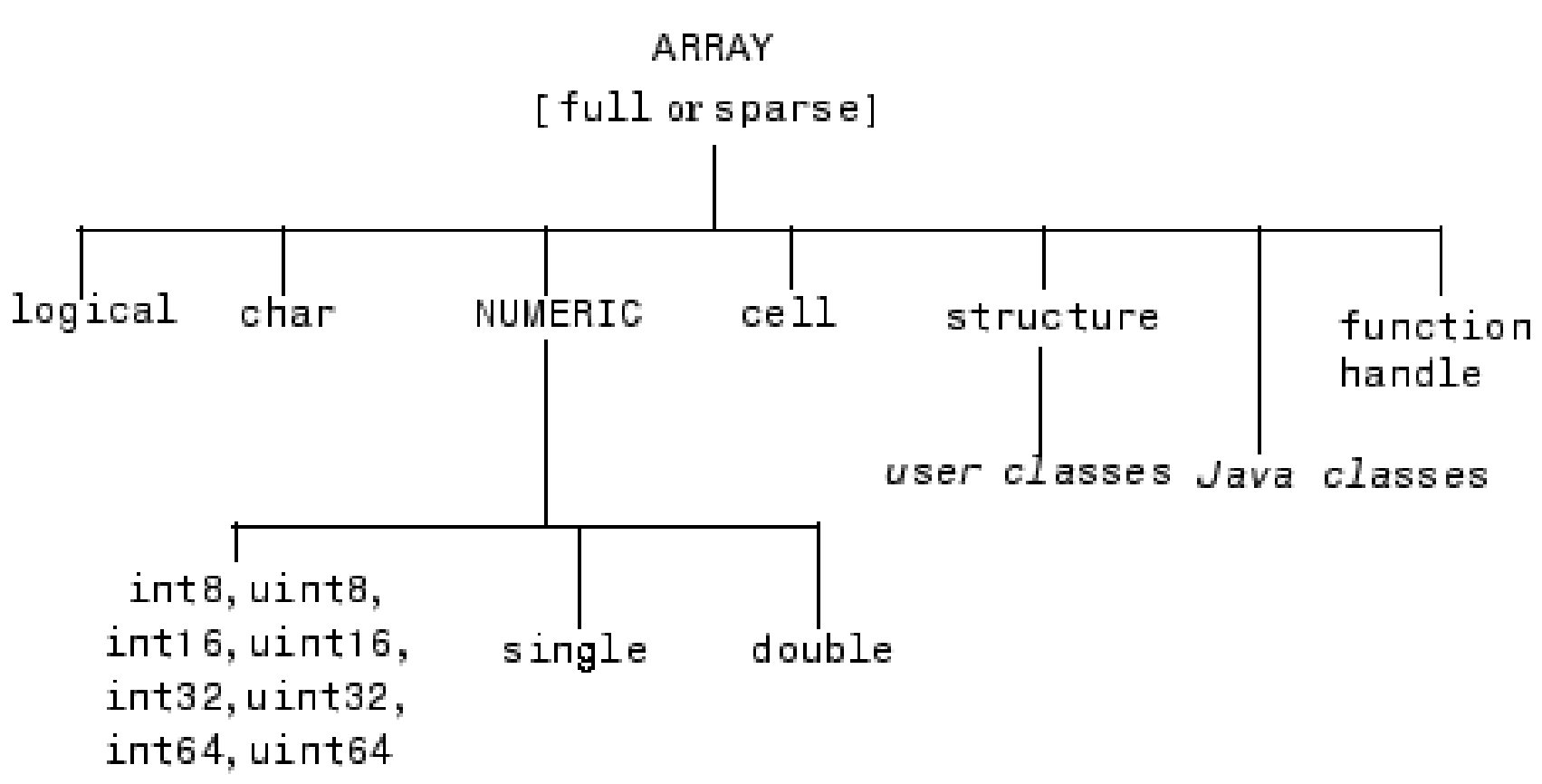
۱-۶- انواع متغیرها

بعضی از مهمترین انواع متغیر در متلب:

double	نقطه اعشار با دقت مضاعف (۸ بایت):
struct	: نوع تعریف شده توسط کاربر
single	نقطه اعشار (۴ بایت) :
uint8	عدد صحیح بی علامت ۸ بیتی :
uint16	عدد صحیح بی علامت ۱۶ بیتی:
uint32	: عدد صحیحی بی علامت ۳۲ بیتی
uint64	عدد صحیحی بی علامت ۶۴ بیتی :
int8	عدد صحیح ۸ بیتی :
int16	عدد صحیح ۱۶ بیتی :
int32	عدد صحیح ۳۲ بیتی :
int64	عدد صحیح ۶۴ بیتی :

- برای دیدن لیست کامل انواع متلب در پنجره فرمان از دستور help datatypes استفاده کنید

فصل اول: ويژگيهاي اصلي MATLAB



فصل اول: ویژگیهای اصلی MATLAB

۱-۶- انواع متغیرها

باید دقت کرد که اگرچه متلب انواع مختلفی از متغیرها را پشتیبانی می کند اما نوع پیش فرض، نوع "دقت مضاعف" است. و برای تبدیل نوع یک متغیر باید دستور کلی زیر را بکار برد:

```
a=TypeName(a);
```

```
>> a=uint8(a);
```
در اینجا نوع متغیر به صحیح بی علامت ۸ بیتی تغییر می کند.

```
>> b = uint32(345);
```
در اینجا یک متغیر از ابتدا از نوع صحیح بی علامت ۳۲ بیتی تعریف شده است

- دقت: در هنگام تبدیل یا ایجاد یک متغیر باید دقت کنید که مقدار انتساب داده شده خارج از دامنه مقادیر آن نوع خاص نباشد. برای انواع صحیح می‌توانید از دستور زیر برای تعیین دامنه استفاده کنید:

```
>> intmin('int16')
>> intmax('int16')
```

استثناء: در مورد جعبه ابزار پردازش تصویر نوع پیش فرض نوع uint8 است.

فصل اول: ویژگیهاي اصلي MATLAB

۱-۷- نامگذاری متغیرها

- اختلاف حروف کوچک و بزرگ
- با حرف الفبا باید شروع شود
- کاراکترهای مجاز: حروف الفبا، اعداد و _
- حداکثر طول نام: با استفاده از تابع namelengthmax در هر نسخه از MATLAB می‌تواند تعیین شود. در نسخه ۲۰۰۶، حداکثر ۶۳ کاراکتر است.
- مراقب باشید متغیر شما با یک تابع درونی MATLAB یا تابعی که توسط خود شما نوشته شده است همنام نباشد. برای اطمینان از دستور which –all varName استفاده کنید

مثال:

```
>>This_Is_a_Variable=5;
```

فصل اول: ویژگیهاي اصلي MATLAB

۱-۸-متغیرهای ویژه

متغیرهای زیر در محیط متلب بصورت پیش‌فرض وجود دارند.

ans
NaN
pi
eps
inf

فصل اول: ویژگیهای اصلی MATLAB

۱-۹-علائم نقطه‌گذاری و جملات توضیحی

- برای درج یک متن توضیحی در برنامه‌های متلب باید از کاراکتر % استفاده شود.

```
>> a=5; %"a" is a variable
```

- برای نوشتن ادامه یک جمله در سطر بعد باید از ... استفاده کرد:

```
>> b=a+a^2+...
      3*a^3;
```

فصل اول: ویژگیهاي اصلي MATLAB

۱-۱۱- بعضی از توابع ریاضی در متلب

abs	conj	log10
acos	exp	real
asin	fix	imag
acosh	round	rem(x,y)
asinh	gcd(x,y)	sign
atan	lcm(x,y)	sqrt
atanh	log	

فصل اول: ویژگیهاي اصلي MATLAB

۱-۱۲-راهنمای متلب

متلب دارای دستورات راهنمای متفاوتی است که هم از طریق منوی start و هم از طریق اعلان متلب قابل دسترسند.

demo
help
lookfor

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳-فایلهای متنی(Script) یا فایلهای m

بمنظور اجرای چند دستور بطور همزمان و بدون نیاز به تایپ مجدد، از فایلهای متنی استفاده می‌شود.

این فایلها باید دارای پسوند m باشند.

فصل اول: ویژگیهاي اصلي MATLAB

۱-۱۳-۱- مراحل ایجاد فایلهای متنی

1. باز کردن یک فایل جدید در ویرایشگر متلب:

File>New>m-file

2. تایپ کردن دستورات متلب در فایل مذکور
3. ذخیره کردن فایل با نامی مشخص:

File>Save As...

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳-۲-روش اجرای یک فایل متنی

برای اجرای یک فایل متنی کافی است نام آنرا در جلوی اعلان متلب تایپ کرده کلید Enter را بزنیم.

نکته: از این پس متن برنامه ها(کد نوشته شده در فایلهای m) با رنگ سبز نشان داده خواهد شد.

مثال: برنامه sample1.m

```
% SAMPLE1: A Simple m-file
n=10;a=2;b=4;
c=n*a^3/b + 3*n*a^2/b^2+6*n*a/b^3
```

```
>> sample1
    c=
        29.3750
```

فصل اول: ویژگیهاي اصلي MATLAB

۱-۱۳-۳- توابع و دستورات مفید در فایلهای m

1. تابع disp(x): این تابع مقدار یک متغیر یا یک رشته متنی را نمایش می‌دهد.

مثال:

```
>> n=10;
>>disp(n)
     10
>> disp('This is a string')
     This is a string
```

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳-۴- توابع و دستورات مفید در فایلهای m

2. تابع x=input(s): برای گرفتن مقدار یک متغیر از ورودی.

مثال:

```
n=input('Please tell me "n" value: ')
----------------------------------------------
Please tell me "n" value: 10
n=
     10
```

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳-۴- توابع و دستورات مفید در فایلهای m

3. دستور pause: توقف موقت در حین اجرا.

```
pause
pause(n)  % n seconds
```

مثال:

```
%SAMPLE2: Enhanced Sample1
n=10;
a=input(‘ “a” value=  ‘);
b=input(‘ “b” value=  ‘);
c=n*a^3/b + 3*n*a^2/b^2 + 6*n*a/b^3;
disp(‘Please wait 5 seconds only!’);pause(5);
disp(‘Press any key to see answer.’); pause;
disp(‘ ”C” Value is= ‘); disp(c)
```

فصل اول: ویژگیهای اصلی MATLAB

تکلیف ۱-۱: برنامه‌ای بنویسید که یک عدد را از کاربر بگیرد و آنرا در متغیری به نام x ذخیره کند. با استفاده از آن، عبارت زیر را محاسبه کند و مقدار y را با پیغام مناسب نمایش دهد.

```
y=x^3 + 3*x^2 +6*x + 6;
```

- با تایپ نام برنامه در جلوی اعلان MATLAB، آنرا اجرا کنید.
- با استفاده از ویرایشگر MATLAB، برنامه خود را اجرا و trace کنید.

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۴- مدیریت فایل: کار کردن با فایلها و شاخه‌ها

بعضی از دستورات مفید:

- دستور cd: تغییر و یا نمایش شاخه جاری :

```
>>cd
    C:\Matlab\Work
>>cd C:\MyDir
>>cd
    C:\MyDir
```

- دستور dir: نمایش نام فایلها و زیرشاخه‌های دایرکتوری جاری
- دستور delete: حذف(پاک کردن) فایل:

```
>>delete sample1
```

فصل دوم
آرایه‌ها

www.MatlabKar.com

کلاس آموزشی

فصل دوم: آرایه‌ها

۲-۱- ایجاد آرایه

روشهای ایجاد آرایه:

1. با استفاده از علائم ; ، , و [ ]
2. با استفاده از علامت :
3. با استفاده از توابع linspace و logspace
4. با استفاده از ترکیبی از روشهای فوق

فصل دوم: آرایه‌ها

۲-۱-۱- ایجاد آرایه با استفاده از علائم ; ، , و [ ]

از علامت ; برای تعیین سطر جدید و از علامت , برای تعیین ستون جدید استفاده می‌شود.

مثال:

```
>> a=[1,2,3;4,5,6]
a=
    1 2 3
    4 5 6
>> b=[1,2,3,4,5,6]
b=
    1 2 3 4 5 6
```

فصل دوم: آرایه‌ها

۲-۱-۱- ایجاد آرایه با استفاده از علائم ; ، , و [ ]

نکته: بجای علامت ; از enter و بجای علامت , از فاصله خالی نیز می‌توان استفاده کرد

مثال:

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

فصل دوم: آرایه‌ها

۲-۱-۲- ایجاد آرایه با استفاده از علامت ":"

در مواقعی که عناصر یک آرایه رابطه خطی با یکدیگر داشته باشند از این روش می‌توان استفاده کرد.

شکل کلی دستور بصورت زیر است:

ArrayName=first : step : last

- اگر step حذف شود، مقدار ۱ بجای آن بکار خواهد رفت.
- اگر last کوچکتر از first باشد، باید step منفی باشد. در غیر اینصورت مقدار آرایه تهی خواهد شد.

فصل دوم: آرایه‌ها

۲-۱-۲- ایجاد آرایه با استفاده از علامت ":"- ادامه...

مثال:

```
>> x=(0 : 0.1 : 1) * pi;
>> y=sin(x);

>>z=1:5
z=
     1 2 3 4 5
>>t=5:1
t =
   Empty matrix: 1-by-0
```

فصل دوم: آرایه‌ها

۲-۱-۳- ایجاد آرایه با استفاده از توابع linsapce و logspace

با ارائه عناصر اول و آخر و طول آرایه به این توابع می‌توان آرایه‌هایی خطی و یا لگاریتمی بدست آورد.

```
ArrayName=linspace(first,last,length)
```

مثال:

```
>>x=linspace(0,1,11)*pi;
>>y=logspace(1,3,3)
y=
    10  100  1000
```

فصل دوم: آرایه‌ها

۲-۱-۳- ایجاد آرایه با استفاده از ترکیبی از علائم فوق

مثال:

```
>> x=[0,1,2, 4:2:12 ,18,19]
x=
    0 1 2  4 6 8 10 12 18 19
>> y=[10,1,7,4,6,-1 ; linspace(0,10,6) ; 5:-1:0]
y=
    10  1  7  4  6  -1
    0   2  4  6  8  10
   5   4  3  2  1   0
```

فصل دوم: آرایه‌ها

۲-۱-۴- ماتریسهای ویژه

- [ ] : ماتریس تهی
- eye : یک ماتریس یکه با ابعاد داده شده ایجاد می‌کند
- ones : یک ماتریس که تمامی عناصر آن یک می‌باشند با ابعاد داده شده ایجاد می‌کند
- zeros : یک ماتریس صفر با ابعاد داده شده ایجاد می‌کند
- rand : یک ماتریس با عناصر راندوم با توزیع یکنواخت به ابعاد داده شده ایجاد می‌کند
- randn : یک ماتریس با عناصر راندوم با توزیع نرمال به ابعاد داده شده ایجاد می‌کند

فصل دوم: آرایه‌ها

۲-۱-۴- ماتریسهای ویژه- ادامه...

مثال:

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

    1    1    1
    1    1    1
>>ones(2)
ans =

    1    1
    1    1
```

تمرین: سایر توابع فوق را خودتان آزمایش کنید.

فصل دوم: آرایه‌ها

۲-۲- عملیات ریاضی بر روی آرایه‌ها

1. عملیات اسکالر-آرایه: * , \ , / , ^ , + , -
2. عملیات عنصری: .* , .\ , ./ , .^ , + , -
3. عملیات ماتریسی: * , \ , / , ^ , + , - (بعدا توضیح داده خواهد شد)

فصل دوم: آرایه‌ها

۲-۲-۱- عملیات ریاضی اسکالر-آرایه

با استفاده از عملگرهای ریاضی متلب براحتی می‌توان عملیات ریاضی اسکالر-آرایه را انجام داد.

مثال:

```
>> x=[1 2 3;4 5 6; 7 8 9];
>> y=2*x + 4
y=
     6     8    10
    12    14    16
    18    20    22
```

فصل دوم: آرایه‌ها

۲-۲-۲- عملیات ریاضی عنصری بین دو آرایه

بدین منظور باید دو آرایه حتما هم بعد باشند.

مثال:

```
>> a=[2 4 6; 3 5 6; 10 -1 0];
>> b=[-1 0 0; 2 1 1; 0 0 3];
>> c= (2*a ./ (b+1)) .^ 2
c =
   Inf    64   144
     4    25    36
   400     4     0
```

فصل دوم: آرایه‌ها

۲-۳- ترانهاده یک ماتریس

برای محاسبه ترانهاده یک ماتریس از علامت ' استفاده می‌شود.

مثال:

```
>> a=[2 1 7
      4 5 -1
      6, 6, 0];
>>b=a'
    2     4     6
    1     5     6
    7    -1     0
```

فصل دوم: آرایه‌ها

۲-۴- بکاربردن توابع ریاضی بر روی آرایه‌ها

توابع متلب بصورت ماتریسی عمل می‌کنند. یعنی لازم نیست تابعی مانند sin را یک به یک بر روی عناصر یک آرایه اعمال کرد. بلکه براحتی می‌توان با یک دستور مقدار سینوس کل عناصر آرایه را محاسبه نمود.

مثال:

```
>>a=[2 4 6; 3 5 6; 10 -1 0];
>>SinA=sin(abs(a) / 10)
SinA =
    0.1987    0.3894    0.5646
    0.2955    0.4794    0.5646
    0.8415    0.0998         0
```

فصل دوم: آرایه‌ها

تمرین ۲-۱

1. برنامه ای بنویسید که عدد صحیح n را از کاربر بگیرد و برداری ۱۰۰ عنصری بین ۰ و $2n\pi$ ایجاد نموده در متغیر x قرار دهد. سپس مقادیر y را از رابطه زیر محاسبه کرده نمایش دهد:

$$y=|\sin(x)|*x^2$$

2. برنامه فوق را طوری تغییر دهید که علاوه بر مقدار n، عددی بین ۰ و ۱ را نیز از کاربر بگیرد و در متغیر جدید d قرار دهد. سپس بردار x را بین ۰ و $2n\pi$ اما با گامهایی برابر با d محاسبه نماید.

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه

m2=m1(آرایه‌ای از اندیس‌ها , آرایه‌ای از اندیس‌ها)

مثال:

```
>>a=[1 2 3
      4 5 6
      7 8 9];
>>k1=[1,2];k2=[2,3];
>>b=a(k1,k2)
b=
    2   3
    5   6
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>c=a([1 2 3],[1,3])
c=
     1   3
     4   6
     7   9
>>d=a([3,2],[3,1])
d=
     9   7
     6   4
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>e=a([1,2,3],2)
e=
    2
    5
    8
>>f=a(1:2:3 , 3:-2:1)
f=
    3   1
    9   7
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>g=a(1:3 , 1:2)
g=
    1   2
    4   5
    7   8
>>h=a(1:2:3, : )
h=
    1   2       3
    7   8       9
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>> k=a( : , : )
k=
     1   2    3
     4   5    6
     7   8    9
>>l=a(1:end,end)
l=
     3
     6
     9
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

نکته:

```
>>n=a([1 1 1] , :)
n=
     1     2     3
     1     2     3
     1     2     3
>>m=a( : , [3 3 3 3])
m=
     3     3     3     3
     6     6     6     6
     9     9     9     9
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

نکته:

```
>>p=a( : )
p=
     1
     4
     7
     2
     5
     8
     3
     6
     9
```

فصل دوم: آرایه‌ها

$$\left[\begin{array}{c|c|cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array}\right]$$

تمرین ۲-۲

1. ماتریس سمت راست را بدون وارد کردن مستقیم عناصر ایجاد کنید.

2. ماتریسی شامل ستونهای سوم تا هشتم و سطرهای چهارم تا نهم ماتریس فوق ایجاد کنید.

فصل دوم: آرایه‌ها

۲-۶- حذف بخشی از آرایه

بمنظور حذف بخشی از یک آرایه می‌توان ماتریس تهی را به آن بخش نسبت داد:

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

>>a(1 : 2 , : ) = [ ]
a=
     7     8     9
```

فصل دوم: آرایه‌ها

۲-۷-جستجوی زیرآرایه

بمنظور یافتن عناصری از آرایه که در شرط خاصی صدق می‌کنند می‌توان از دستور find استفاده کرد (این دستور عناصر را بصورت ستونی شمارش می‌کند):

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

فصل دوم: آرایه‌ها

۲-۷-جستجوی زیرآرایه-ادامه-

```
>>b=a(k)
b=
     7
     8
     6
     9
```

فصل دوم: آرایه‌ها

۲-۷-جستجوی زیرآرایه-ادامه-

دستور find در صورتیکه با دو آرگومان خروجی بکار برده شود، شماره سطر و ستون عناصر را باز می‌گرداند:

```
>>[k1,k2]=find( a > 5)
k1=                    k2=
    3                       1
    3                       2
    2                       3
    3                       3
```

فصل دوم: آرایه‌ها

۲-۸- اندازه آرایه:

با استفاده از دستورات length و size می‌توان ابعاد یک آرایه را بدست آورد.

دستور length اگر بر روی یک بردار بکار برده شود، تعداد عناصر آنرا باز می‌گرداند و اگر بر روی یک ماتریس بکار رود، بزرگترین بعد آنرا باز می‌گرداند.

دستور size انعطاف‌پذیرتر بوده و می‌تواند به روشهای زیر بکار برده شود:

- اگر با یک آرگومان ورودی بکار برده شود، طول و عرض ماتریس را بازمی‌گرداند.
- اگر با دو آرگومان ورودی بکار برده شود، بطوریکه آرگومان دوم ۱ یا ۲ باشد، بترتیب تعداد سطرها یا ستونهای ماتریس را بازمی‌گرداند
- اگر با یک آرگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را در یک بردار سطری دو عنصری بازمی‌گرداند
- اگر با دو ارگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را بترتیب در آرگومان اول و دوم باز می‌گرداند

فصل دوم: آرایه‌ها

۲-۸- اندازه آرایه-ادامه-

مثال:

```
>>a=[1     2     3     4
       5     6     7     8];
>>size(a)
ans=
     2     4
>>[r , c] = size(a)
r =
     2
c=
     4
```

```
>>r=size(a , 1)
r=
     2
```

```
>>c=size(a,2)
c=
     4
```

فصل دوم: آرایه‌ها

۲-۸- اندازه آرایه-ادامه-تعداد سطر یا ستون هر کدام که بیشتر است

مثال:

```
>>b=[1      2      3      4];
>>l=length(b)
l=
    4
>>a=[1      2      3      4
      5      6      7      8];
>>la=length(a)
la=
    4
```

فصل دوم: آرایه‌ها

تمرین ۲-۳

1. برنامه ای بنویسید که ماتریسی دو ستونی را که مقادیر ستون اول آن نمرات دروس مختلف یک ترم یک دانشجو و مقادیر ستون دوم آن تعداد واحد مربوط هر یک از آن دروس می باشد را از کاربر بگیرد و عملیات زیر را بر روی انجام دهد
   - محاسبه تعداد واحدها
   - محاسبه معدل ترم
   - نمایش نتایج با پیغام مناسب

فصل سوم

توابع و عملیات ماتریسی

MATLAB®

www.MatlabKar.com

کلاس آموزشی

فصل سوم: توابع و عملیات ماتریسي

۳-۱- حل دستگاه معادلات خطی

با استفاده از عملیات ضرب و تقسیم ماتریسی در متلب براحتی می‌توان دستگاههای معادلات خطی را حتی در مواردی که تعداد معادلات با تعداد متغیرها مساوی نباشند، حل کرد. بدین‌منظور باید بردار سمت راست معادلات را بر ماتریس ضرایب متغیرها تقسیم کرد.

فصل سوم: توابع و عمليات ماتريسي

۳-۱- حل دستگاه معادلات خطی-ادامه

مثال:

$$\begin{cases} x+2y+3z=366 \\ 4x+5y+6z=804 \\ 7x+8y=351 \end{cases}$$

```
>>a=[1  2  3;
      4  5  6;
      7  8  0];
>> x= a \ b     یا     >>x=a ^ (-1) * b  یا     >>x=inv(a) * b
x=
    25
    22
    99
```

فصل سوم: توابع و عملیات ماتریسي

۳-۲- تعدادی از توابع ماتریسی

- det : دترمینان ماتریس را محاسبه می‌کند
- inv : معکوس ماتریس را محاسبه می‌کند
- trace: مجموع عناصر قطر اصلی یک ماتریس را بازمی‌گرداند

فصل سوم: توابع و عملیات ماتریسی

تکلیف۳-۱: دستگاه معادلات خطی زیر را حل کنید و بهترین جواب را بدست آورید:

$$\begin{cases} x+2y+3z+7t=4 \\ 6x+7y+22z+32t=5 \\ 98x+5y-23z+t=7 \\ 32x+5y-75z+23t=1 \\ 22x+2y+3z+t=0 \end{cases}$$

تکلیف ۳-۲: برنامه‌ای بنویسید که ماتریس ضرایب و مقادیر سمت راست یک دستگاه معادلات خطی را از کاربر بگیرد و پاسخ دستگاه را با پیغام مناسب نمایش دهد.

فصل چهارم

عملیات منطقی و رابطه‌ای

www.MatlabKar.com

کلاس آموزشی

فصل چهارم: عملیات منطقي و رابطه‌اي

- تعریف: عملیاتی که بر اساس مقادیر منطقی true و false (یا ۰ و ۱) استوار باشد را عملیات منطقی می‌گویند.

۴-۱- عملگرهای رابطه‌ای

عملگرهای رابطه‌ای زیر در متلب تعریف شده‌اند:

```
< , > , <= , >= , == , ~=
```

فصل چهارم: عملیات منطقي و رابطه‌اي

۴-۱-۱- مقایسه دو آرایه

با استفاده از عملگرهای رابطه‌ای می‌توان دو آرایه را عنصر به عنصر با یکدیگر مقایسه کرد. به ازای نقاطی که در شرط ذکر شده صدق می‌کنند، مقدار ۱ و به ازای سایر نقاط مقدار ۰ باز گردانده می‌شود.

```
>> a= [1 , 2, 3 , 4 , 5];
>>b=[10 , 2 , 13 , 4 , 8];
>>tf=(a == b)
tf=
    0   1       0       1       0
```

متغیر tf یک متغیر از نوع منطقی(logical) خواهد بود. یعنی تنها می‌تواند مقادیر ۰ و ۱ را در خود نگهدارد. بعنوان تمرین سعی کنید عنصر سوم tf را با ۵۰ جایگزین کنید.

فصل چهارم: عمليات منطقي و رابطه‌اي

۴-۱-۲- مقایسه یک آرایه با یک عدد

در این حالت تمامی عناصر آرایه با یک عدد مقایسه می‌شوند:

```
>> a = [1 , 2 , 3 ; 4 , 2 , 2 ; 1 , 10 , 0];
>> t= a >=2
t=
     0     1     1
     1     1     1
     0     1     0
```

فصل چهارم: عملیات منطقي و رابطه‌اي

تکلیف ۴-۱: برنامه‌ای بنویسید که نمرات دروس ریاضی۱(۴ واحد)، مکانیک(۳ واحد) و معارف اسلامی(۲ واحد) چند دانشجو را بصورت یک ماتریس(n*3) از کاربر بگیرد و موارد زیر را محاسبه و با پیغام مناسب نمایش دهد:

- تعداد دانشجویان
- معدل هر دانشجو
- معدل هر درس
- معدل کل دروس برای تمامی دانشجویان(یک عدد)
- میانگین نمرات زیر ۱۰ بدون احتساب واحد هر درس

فصل چهارم: عمليات منطقي و رابطه‌اي

۴-۲- عملگرهای منطقی

عملگرهای & و | و ~ عملگرهای منطقی در متلب هستند که به ترتیب معادل AND و OR و NOT می‌باشند.

فصل چهارم: عمليات منطقي و رابطه‌اي

۴-۲-عملگرهای منطقی(ادامه...)

مثال:

```
>> a= 1 : 9;
>> t = a > 3
   0   0    0   1   1   1   1   1   1

>> f = ~ ( a > 3)
   1   1    1   0   0   0   0   0   0

>> tf = ( a > 3) & (a <=7)
   0   0    0   1   1   1   1   0   0
```

فصل چهارم: عملیات منطقي و رابطه‌اي

۴-۳- توابع رابطه‌ای و منطقی

علاوه بر عملگرهای رابطه‌ای و منطقی در متلب توابعی نیز بدین منظور وجود دارد که عبارتند از:

all (x) : درصورتیکه تمامی عناصر یک بردار نامساوی ۰ باشد مقدار ۱ و در غیر اینصورت ۰ باز می‌گرداند

any(x) : درصورتیکه حداقل یکی از عناصر یک بردار نامساوی ۰ باشد مقدار ۱ و در غیر اینصورت ۰ باز می‌گرداند

xor(x,y) : یای انحصاری

فصل چهارم: عمليات منطقي و رابطه‌اي

۴-۳- توابع رابطه‌ای و منطقی-ادامه...

مثال:

```
>>x=[1 1 0];
>>y=[0 1 0];
>>tor= x | y
tor=
    1   1     0
```

```
>>txor=xor(x , y)
txor=
    1     0     0
```

فصل چهارم: عمليات منطقي و رابطه‌اي

۴-۳- توابع رابطه‌ای و منطقی-ادامه...

مثال:

```
>>a= [1 1 1 0];
>>t=any(a)                    >>t=all(a)
   t=                          t=
      1                           0
>>a=[3 2 4];
>>t=any(a==2)
t=
   1
```

فصل پنجم :

تصمیم‌گیری و کنترل روند، استفاده از حلقه‌ها و دستورات شرطی در متلب

MATLAB®

www.MatlabKar.com

کلاس آموزشی

فصل ششم: تصمیم‌گیری و کنترل روند

در این فصل در مورد جملات شرطی و انواع حلقه‌های تکرار صحبت خواهیم کرد.

۶-۱-حلقه for:

شکل کلی حلقه for در متلب بصورت زیر است:

```
for   x= آرایه
    دستورات
end
```

در اینصورت حلقه فوق به تعداد ستونهای آرایه مشخص‌شده تکرار خواهد شد و در هر تکرار یکی از ستونهای این آرایه در متغیر x قرار گرفته و در بدنه حلقه قابل استفاده است. در صورتیکه آرایه یک بردار باشد، هر بار یک عنصر از آن در متغیر x قرار خواهد گرفت.

تذکر: با توجه به تواناییهای ماتریسی متلب از کاربرد حلقه‌ها در متلب تا حد ممکن باید پرهیز گردد زیرا اینکار باعث کند شدن شدید برنامه می‌شود و نیاز به کد نویسی بسیار بیشتری دارد.

فصل ششم: تصمیم‌گیري و کنترل روند

۶-۱- حلقه for-ادامه-

مثال:

```
for n=1:10
    x(n) = sin(n * pi / 10);
end;
--------------
for k=[1,2,3,7]
    x(k) = k+1;
end;
>>x
x=
    2   3   4   0   0   0   0   8
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۲- حلقه while :

در مواردی که بخواهیم یک یا چند دستور تا برقراری شرط خاصی تکرار گردند از این حلقه استفاده می‌کنیم. شکل کلی حلقه while بصورت زیر است:

```
while    شرط
    دستورات
end
```

حلقه فوق تا زمانیکه شرط ذکر شده برقرار باشد تکرار خواهد شد.

فصل ششم: تصمیم‌گیري و کنترل روند

۶-۲- حلقه while –ادامه-

مثال:

```
t=1;
while  t ~= -1
    t = input( ‘ Enter a number to continue or -1 to exit from
    this block: ‘);
    …
end
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۳- ساختار if-else-end

هرگاه بخواهیم یک یا چند جمله در صورت برقرار بودن شرط خاصی (یکبار) اجرا شود، از بلوک if استفاده می‌کنیم. شکل کلی استفاده از این دستور بصورت زیر است:

```
if  شرط ۱
      دستورات
elseif  شرط ۲
      دستورات
elseif  ...
      ...
else
      دستورات
end;
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۳- ساختار if-else-end –ادامه-

مثال:

```
Epsilon = 1;
while 1 > 0
    Epsilon = Epsilon / 2;
    if Epsilon + 1 == 1
        break;
    end
end
```

- نکته: با دستور break می‌توان یک حلقه while یا for را شکست. در اینصورت اجرای برنامه از نخستین دستور بعد از حلقه ادامه خواهد یافت.

فصل ششم: تصمیم‌گیری و کنترل روند

تکلیف ۶-۱: برنامه‌ای بنویسید که نمرات چند دانشجو را به صورت یک بردار بگیرد و عملیات زیر را انجام دهد:

- در صورتیکه ورودی کاربر بردار نباشد(ماتریس یا اسکالر باشد) پیام خطا دهد.(راهنمایی برای دادن پیام خطا می‌توانید از تابع error به جای disp استفاده کنید)
- با استفاده از حلقه for و دستورات شرطی if-else-end تک تک نمرات را چک کند و به صورت زیر آنها را تغییر دهد:
  - نمرات کمتر از ۵ را به ۹ تغییر دهد
  - نمرات بین ۵ و ۸ را به ۹,۵ تغییر دهد.
  - نمرات بین ۸ و ۱۰ را به ۱۰ تغییر دهد.
  - نمرات بین ۱۰ و ۱۵ را ۱ نمره افزایش دهد
  - نمرات بیشتر از ۱۵ و کمتر از ۲۰ را ۰,۵ نمره افزایش دهد.

تکلیف ۶-۲: برنامه دیگری بنویسید که همان کارهای برنامه ۶-۱ را بدون استفاده از حلقه انجام دهد.

فصل ششم:

ایجاد توابع در متلب

Functions

MATLAB®

www.MatlabKar.com

کلاس آموزشی

فصل هفتم: ایجاد توابع در متلب

۷-۱-مزایای استفاده از توابع به جای فایلهای اسکریپت

1. سرعت بالاتر
2. صرفه‌جویی در حافظه کامپیوتر
3. توسعه توانایی‌های متلب

توابع بر خلاف فایلهای اسکریپت در هنگام اجرا یکبار کامپایل شده و اجرا می‌شوند. در حالیکه فایلهای اسکریپت سطر به سطر کامپایل و اجرا می‌گردند. این امر باعث افزایش سرعت اجرای توابع در مقایسه با فایلهای اسکریپت می‌شود.

متغیرهای تعریف شده در توابع پس از پایان اجرای آن از حافظه پاک می‌شوند و بطور کلی فضای کاری توابع مستقل از فضای کاری متلب است. خصوصا در مواقعی که برنامه با ماتریسهای بزرگ(مانند تصاویر) کار می‌کند بهتر است از توابع استفاده شود

فصل هفتم: ایجاد توابع در متلب

۷-۱-مزایای استفاده از توابع به جای فایلهای اسکریپت-ادامه-

اکثر دستورات اصلی متلب و جعبه‌ابزارهای آن با استفاده از توابع نوشته شده است. به بیان دیگر به راحتی می‌توان قابلیتهایی که در حال حاضر در متلب وجود ندارد را با نوشتن یک مجموعه از توابع به آن افزود. همین امر باعث شده است که در دهه گذشته قابلیتهای متلب در رشته‌های مختلف علمی و فنی با سرعت چشمگیری توسعه یابد.

نکته : بهتر است در هنگام نوشتن یک برنامه آنرا بصورت اسکریپت بنویسیم تا اشکالزدایی آن آسانتر باشد اما پس از کامل شده برنامه آنرا به فانکشن تبدیل کنیم تا سرعت و کیفیت آن افزایش یابد.

فصل هفتم: ایجاد توابع در متلب

۷-۲- تفاوتهای توابع و فایلهای متنی

1. فایلهای متنی سطر به سطر ترجمه و اجرا می‌شوند اما توابع یکبار بطور کامل ترجمه و سپس اجرا می‌گردند.
2. محیط کاری فایلهای متنی همان محیط کاری متلب است اما محیط کاری هر تابعی مختص خود اوست یعنی اگر متغیری در یک تابع تعریف شود تنها در آن تابع قابل دسترسی است و برعکس متغیرهای تعریف شده در محیط کاری متلب در داخل توابع تعریف شده نیستند.(مگر اینکه بصورت عمومی تعریف شده باشند)
3. توابع تنها از طریق آرگومانهایشان با محیط خارج در ارتباطند

فصل هفتم: ایجاد توابع در متلب

۷-۳- نحوه ایجاد توابع

تنها تفاوت ظاهری یک تابع و یک فایل متنی آن است که سطر اول یک تابع با کلمه کلیدی function شروع می‌شود که شکل کلی آن بصورت زیر است:

```
function [argout1 , argout2, ... ] =
    funcname(argin1,argin2,...)
% معرفی فانکشن در یک سطر
% راهنمای استفاده
% از این فانکشن

% نویسنده فانکشن ، نسخه و سال ساخت
    بدنه تابع
    ...
```

فصل هفتم: ایجاد توابع در متلب

۷-۳- نحوه ایجاد توابع-ادامه-

نکات:

1. تابع ممکن است هیچ آرگومان ورودی یا خروجی نداشته باشد.
2. اولین سطر بعد از اعلان تابع، یک جمله توضیحی است که در هنگام استفاده از دستور lookfor در متلب مورد جستجو قرار می‌گیرد
3. تمامی سطرهای توضیحی تا نخستین سطر غیر توضیحی در هنگام استفاده از دستور help نمایش داده میشود.

نکته :بهتر است هنگام نوشتن یک تابع حتما یکی دو سطر در مورد نحوه استفاده از آن و عملکرد آن توضیح داده شود تا کاربر بتواند با استفاده از دستور help متلب با روش استفاده از آن تابع و قابلیتهای آن آشنا شود.

فصل هفتم: ایجاد توابع در متلب

۷-۴- فرمانهای return و error

با استفاده از این دو دستور می‌توان اجرای یک تابع را پیش از رسیدن به انتهای آن متوقف کرد.

تفاوت دستور error با دستور return آن است که دستور error می تواند یک پیغام خطا نیز بمنظور آگاهسازی کاربر نمایش دهد.

مثال:

```
s= input( ‘Please enter a scalar value= ‘);
if  length (s) > 1
    error(‘Error! Your input isn’’t a scalar!’);
end
a= linspace( 0 , abs(s) , 100);
```

فصل هفتم: ایجاد توابع در متلب

۷-۵- تعیین تعداد آرگومانهای بکار رفته در حین اجرا

در متلب می‌توان توابع را با تعداد آرگومان کمتر از تعداد آرگومان موجود در تعریف تابع نیز فراخوانی کرد. مثلا تابع size در متلب با دو آرگومان نوشته شده است اما با یک آرگومان نیز قابل اجراست که البته مقدار بازگشتی به تعداد آرگومانهای مورد استفاده بستگی خواهد داشت.

در صورتیکه بخواهیم از تعداد آرگومانها در حین اجرا مطلع شویم باید از توابع nargin و nargout به ترتیب برای تعداد آرگومانهای ورودی و تعداد آرگومانهای خروجی استفاده کنیم.

همچنین توابع nargchk و nargoutchk تعداد آرگومانهای ورودی و خروجی را چک می‌کنند و درصورتیکه با تعداد درخواست شده برابر نباشند پیام خطای مناسب را نشان می‌دهند.

فصل هفتم: ایجاد توابع در متلب

۷-۶- نکاتی در مورد توابع

- در یک فایل می‌توان بیش از یک تابع تعریف کرد. در اینصورت تمامی این توابع می‌توانند یکدیگر را فراخوانی کنند اما تنها نخستین تابع از خارج از این فایل قابل فراخوانی است.
- نام فایل با نام نخستین تابع آن باید یکسان باشد. در غیر اینصورت بمنظور اجرای تابع باید از نام فایل به جای نام تابع استفاده گردد که البته کار درستی نیست.

فصل هفتم: ایجاد توابع در متلب

مثال ۷-۱- تابعی بنویسید که یک بردار(آرایه سطری یا ستونی) را از کاربر بگیرد و مراحل زیر را انجام دهد:

- تعداد آرگومان ورودی و خروجی که توسط کاربر وارد شده است را چک کند و در صورتیکه تعداد آرگومان ورودی بیشتر یا کمتر از یک و تعداد آرگومان خروجی بیشتر از یک باشد، پیام خطا نمایش داده از تابع خارج شود.
- ابعاد آرگومان ورودی را چک کند و درصورتیکه آرایه‌ای غیر سطری یا غیر ستونی باشد(یعنی در صورتیکه به جای بردار، ماتریس باشد)، با پیام خطا از تابع خارج شود.
- عبارت زیر را بر روی مقادیر ورودی اعمال نموده به عنوان خروجی بازگرداند.

$$y=2\exp(4x^2)+3\sin(2\pi x)+10$$

- تعداد آرگومان خروجی را چک کند و در صورتیکه برابر با صفر باشد، نمودار تغییرات y در مقابل x را رسم کند.(راهنمایی: برای رسم نمودار از تابع plot(x,y) استفاده کنید.

فصل هفتم: ایجاد توابع در متلب

تکلیف ۷-۱-تابعی بنویسید که یک عبارت ریاضی دلخواه را از کاربر (به صورت یک رشته کاراکتری) به عنوان آرگومان اول و یک آرایه را به عنوان آرگومان دوم بگیرد و :

- چک کند که تعداد آرگومان ورودی دقیقا دو عدد باشد(با استفاده از تابع nargchk)
- چک کند که تعداد آرگومان خروجی دقیقا یک عدد باشد.(با استفاده از تابع nargoutchk)
- چک کند که آرگومان اول حتما یک رشته کاراکتری باشد و آرگومان دوم حتما یک متغیر عددی. (از توابع isstr و isnumeric استفاده کنید)
- با استفاده از تابع eval عبارات ریاضی وارد شده توسط کاربر را بر روی تمامی عناصر آرایه ورودی اعمال نموده، باز گرداند.

فصل هفتم:

نمودارهای دو بعدی

MATLAB®

www.MatlabKar.com

کلاس آموزشی

فصل نهم: نمودار هاي دوبعدي

۹-۱-تابع plot

شکل کلی:

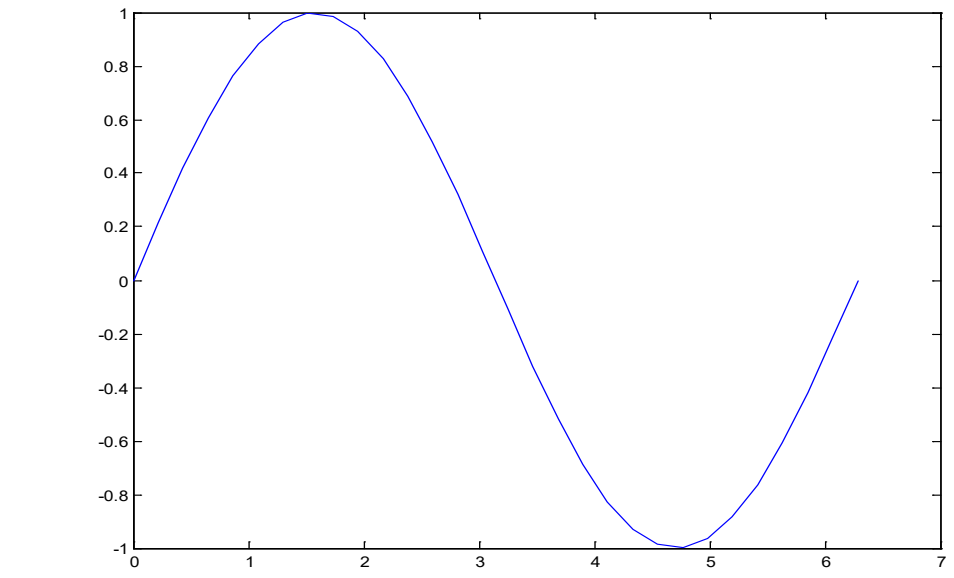
```
plot (x1,y1,’c1s1’,x2,y2,’c2s2,x3,y3,’c3s3’,…)
```

فصل نهم: نمودار هاي دوبعدي

۹-۱- تابع plot – ادامه

مثال:

```
>> x= linspace(0,2*pi , 30); y= sin(x);
>> plot(x,y);
```



فصل نهم: نمودار هاي دوبعدي

۹-۲- رسم چند نمودار مجزا در یک پنجره شکل

بمنظور تقسیم پنجره شکل به چند بخش می‌توان از تابع subplot استفاده کرد.

شکل کلی:

```
subplot( m ,n , p)
```

در این رابطه m تعداد بخشهای افقی، n تعداد بخشهای عمودی و p شماره بخش جاری است. هر دستور ترسیمی بعداز این دستور در مکان p ام اعمال خواهد شد. خانه‌ها بصورت ستونی شمارش می‌شوند.

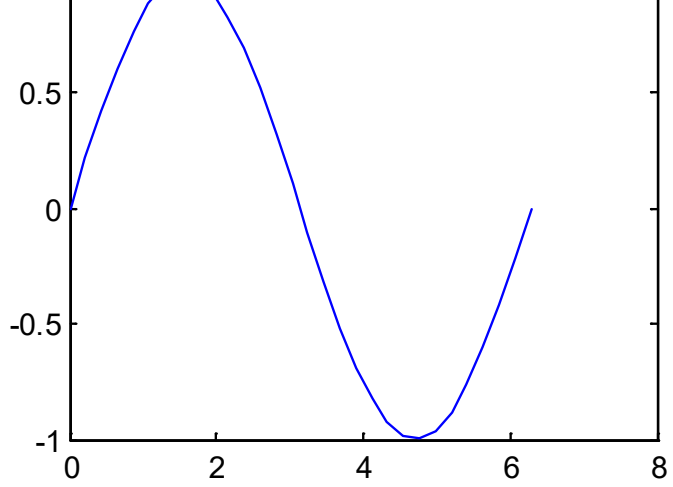
واضح است که مقدار p باید بین ۱ و m*n باشد در غیر اینصورت متلب اعلان خطا می‌کند.

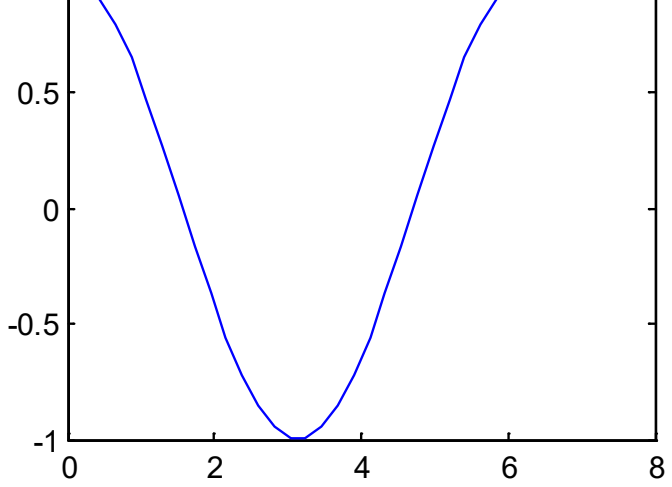
فصل نهم: نمودار هاي دوبعدي

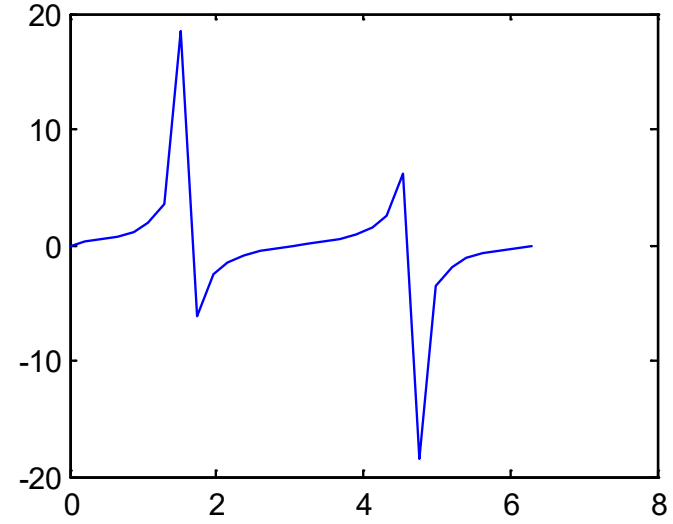
۹-۲- رسم چند نمودار مجزا در یک پنجره شکل-ادامه

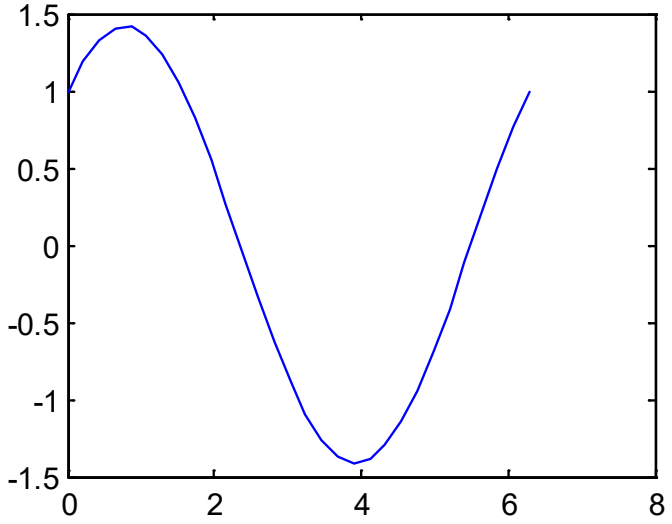
مثال:

```
>> x=linspace(0,2*pi,30);
>> subplot(2,2,1);plot(x,sin(x));
>> subplot(2,2,2);plot(x,cos(x));
>> subplot(2,2,3);plot(x,tan(x));
>> subplot(2,2,4);plot(x,sin(x)+cos(x));
```


0.5
0
-0.5
-1
0
2
4
6
8


0.5
0
-0.5
-1
0
2
4
6
8


20
10
0
-10
-20
0
2
4
6
8


1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8

فصل نهم: نمودار هاي دوبعدي

۹-۳- برچسب گذاری محورهای افقی و عمودی و عنوان

بمنظور برچسب‌گذاری محورها و ایجاد عنوان برای نمودار می‌توان از توابع xlabel, ylabel , title استفاده کرد.

```
>> xlabel(‘یک رشته متنی ’);
>> ylabel(‘یک رشته متنی ’);
>> title(‘یک رشته متنی ’);
```

این دستورات بر روی آخرین نمودار ترسیم شده اعمال میشوند بنابراین بعد از هر دستور plot یا دستور ترسیمی دیگر بلافاصله باید از این دستورات استفاده گردد.

فصل نهم: نمودار هاي دوبعدي

۹-۴- رسم خطوط شبکه‌ای بر روی نمودار

بمنظور ایجاد خطوط شبکه‌ای(چهارخانه‌های نقطه‌چین) بر روی یک نمودار، می‌توان از دستور grid استفاده کرد. شکل کلی استفاده از دستور grid بصورتهای زیر است:

```
>> grid on
```
حالت شبکه‌ای را فعال می‌کند

```
>> grid off
```
حالت شبکه‌ای را غیر فعال میکند

```
>> grid
```
حالت شبکه‌ای را از فعال به غیرفعال و از غیر فعال به فعال تغییر می‌دهد

فصل نهم: نمودار هاي دوبعدي

۹-۵- ایجاد پنجره شکل جدید

بصورت پیش‌فرض در متلب هر نمودار جدید جایگزین نمودار قبلی در همان پنجره شکل میگردد. در صورتیکه بخواهیم چند نمودار در پنجره‌های شکل جداگانه ترسیم شوند از دستور figure استفاده می‌کنیم

```
>> figure;
```

این دستور باعث می‌شود که یک پنجره شکل جدید باز شده و نمودار بعدی در آن پنجره ترسیم گردد.

فصل نهم: نمودار هاي دوبعدي

۹-۶- افزودن متن به نمودار

با استفاده از توابع text و gtext می‌توان متنی را به نمودار اضافه کرد:

```
>> text(x,y,'رشته متنی')
>> gtext('رشته متنی')
```

دستور اخیر اجازه می‌دهد که ناحیه قرار گیری رشته متنی را بتوان با ماوس انتخاب کرد.

فصل نهم: نمودار هاي دوبعدي

۹-۷- افزودن راهنمای علائم: دستور legend

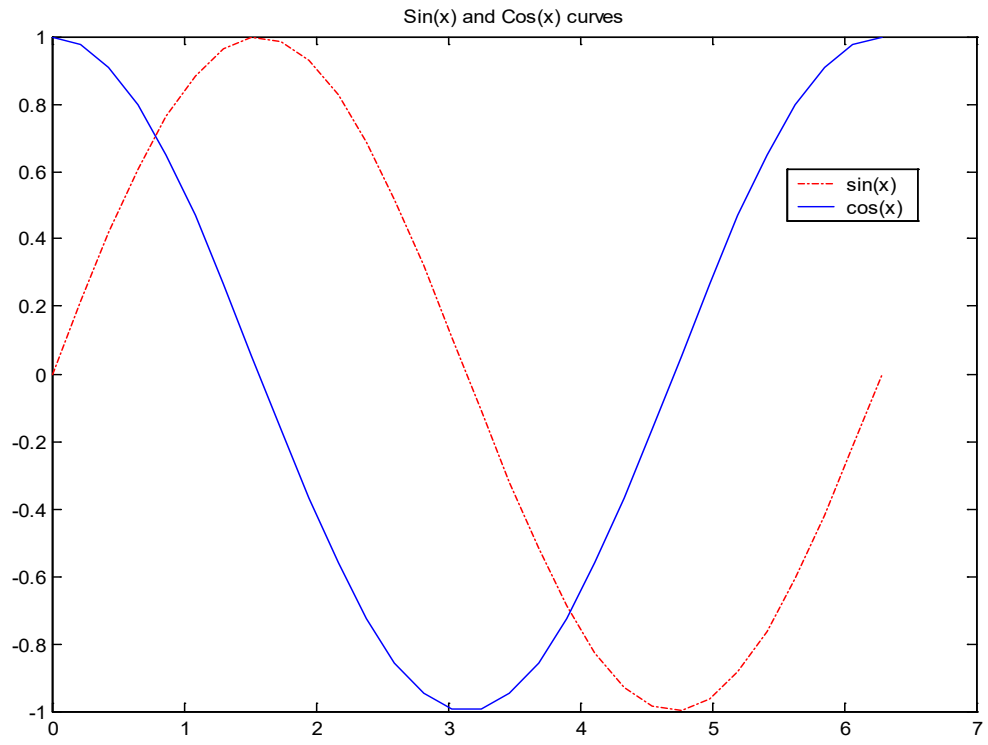
مثال:

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
plot(x,y,'g-.','b-');
legend('sin(x)','cos(x)');
title('Sin(x) and Cos(x) curves);
```

فصل نهم: نمودار هاي دوبعدي

۹-۸- افزودن راهنمای علائم: دستور legend

مثال-ادامه:



فصل نهم: نمودار هاي دوبعدي

۹-۹-دستور axis

با استفاده از این دستور می‌توان دامنه ترسیم را تغییر داد:

```
axis([xmin,xmax,ymin,ymax,zmin,zmax])
```

نمودار در دامنه xmin تا xmax ، ymin تا ymax و... ترسیم می‌گردد.

---

axis off	محورهای مختصات را حذف می‌کند
axis on	محورهای مختصات را ترسیم می‌کند

فصل نهم: نمودار هاي دوبعدي

۹-۱۰- ثابت نگهداشتن نمودار: دستور hold

بصورت پیش‌فرض متلب هر نمودار جدید را جایگزین نمودار قبلی میکند، اگر بخواهیم بدون پاک شدن نمودار فعلی نمودار جدیدی اضافه کنیم باید از دستور hold استفاده نماییم:

hold on	فعال
hold off	غیر فعال
hold	تغییر حالت

فصل نهم: نمودار هاي دوبعدي

۹-۱۱- سایر دستورات

clf: محتویات پنجره شکل جاری را پاک می‌کند

cla: محتویات نمودار جاری را پاک میکند

zoom: حالت زوم را فعال یا غیر فعال می‌کند

ginput: برای گرفتن مختصات یک یا چند نقطه از نمودار با استفاده از ماوس

فصل نهم: نمودارهاي دوبعدي

۹-۱۲- سایر نمودارهای دوبعدی

علاوه بر plot دستورات ترسیم نمودارهای دوبعدی دیگری نیز در متلب وجود دارد که عبارتند از:

polar: ترسیم نمودار در مختصات قطبی

fill: ترسیم نواحی بسته دو بعدی(چندضلعی‌ها)

semilogx, semilogy, loglog:

ترسیم نمودار در مختصات لگاریتمی

stairs: ترسیم نموار پله‌ای

hist: ترسیم نمودار فراوانی

bar: ترسیم نمودار میله‌ای