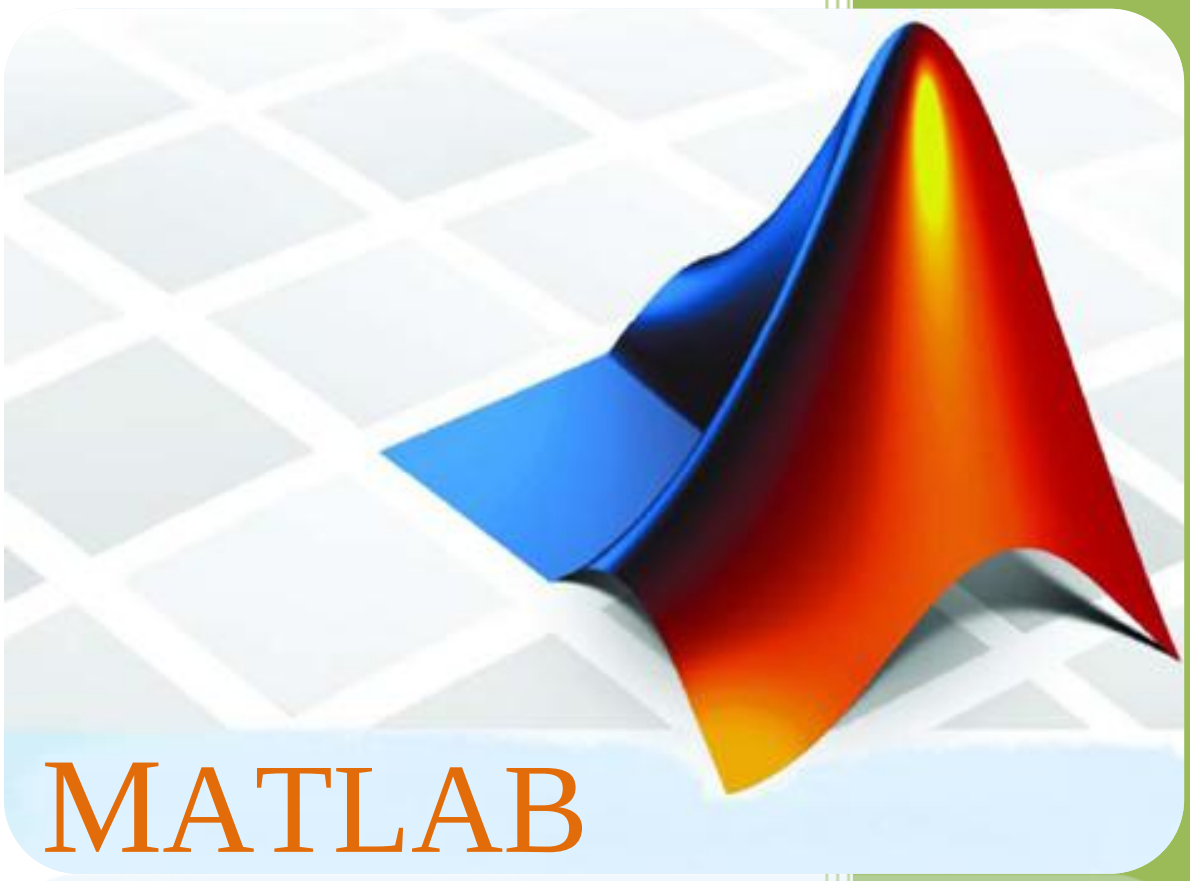
آموزش نرم افزار MATLAB



تهیه کننده:

مهندس عباس محمّدی

فصل اول:

معرفی و آشنایی با ویژگی های نرم افزار MATLAB

مقدمه

MATLAB نرم افزاری است مناسب برای کسانی که با محاسبات عددی سر و کار دارند. نام این نرم افزار از عبارت MATrix LABoratory به معنی آزمایشگاه ماتریس اقتباس شده و هدف اولیه آن قادر ساختن مهندسین و دانشمندان به حل مسائل شامل عملیات ماتریسی بدون نیاز به نوشتن برنامه در زبانهای متداول همچون C و Fortran بود. با گذشت زمان قابلیتهای بسیار بیشتری به این نرم افزار افزوده شده است بطوری که در حال حاضر MATLAB به ابزار پر قدرتی برای ترسیم داده ها، برنامه نویسی و انجام محاسبات مهندسی و پژوهشی تبدیل شده است.

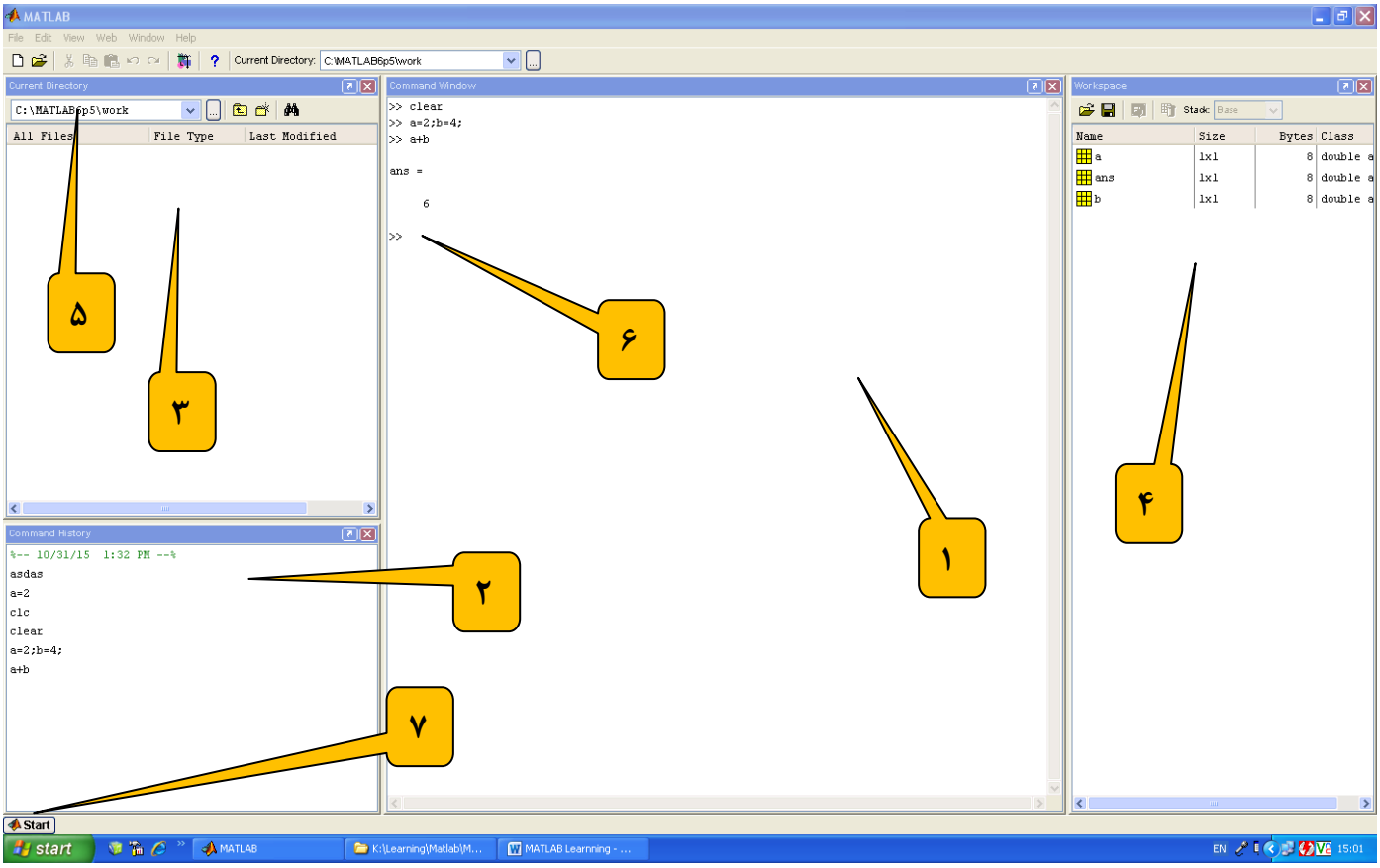
اولین نسخه از نرم افزار متلب توسط آقای مولر که رئیس دانشگاه نیومکزیکو بود در سال ۱۹۷۰ ارائه شد. ایشان قصد داشتند با ایجاد یک نرم افزار محاسباتی مبتنی بر ماتریس و آرایه به دانشجویان خود کمک نماید.

شرکت Math Work در سال ۱۹۸۴ تاسیس شد و اولین نسخه رسمی متلب توسط این شرکت به بازار ارائه شد.

۱-۱- آشنایی با محیط MATLAB

پس از نصب متلب، از مسیر زیر می توان این نرم افزار را اجرا نمود:

Start > All Programs > MATLAB > R2009a > MATLAB 2009a



۱- پنجره فرمان : Command window

۲- پنجره تاریخچه دستورات: Command History

۳- پنجره دایرکتوری جاری : Current Directory

۴- پنجره فضای کاری : Work Space

۵- دایرکتوری جاری : Current Folder

۶- خط فرمان : Command Line

۷- دکمه استارت(امکانات جانبی و پیشرفته): Start

۸- دکمه سیولینک(محیط شبیه سازی):Simulink

۱-۲- قواعد نامگذاری متغیرها

- *در نامگذاری یک متغیر بین حروف کوچک و بزرگ در متلب تفاوت وجود دارد*
- *نام متغیر با حروف الفبا باید شروع شود*
- *کاراکترهای مجاز: حروف الفبا، اعداد،زیرخط (_) و ...*
- *حداکثر طول نام: با استفاده از تابع namelengthmax در هر نسخه از MATLAB می‌تواند تعیین شود. در نسخه ۲۰۰۶، حداکثر ۶۳ کاراکتر است.*

مثال:

```
>>This_Is_a_Variable=5;
>>a=2;b=3;ali=12;str123='ali'
```

۱-۳-عملیات ریاضی پایه

مثال: محاسبه یک عبارت:

راه اول:

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

راه دوم:

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

۱-۴-عملگرهای ریاضی متلب:

تابع	عملکرد	مثال
^	توان	3^2=9
+	جمع	3+2=5
-	تفریق	3-2=1
*	ضرب	3*2=6
/	تقسیم چپ بر راست	3/2=1.5
\	تقسیم راست بر چپ	3\2=0.67

مثال:

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

نکته: ترتیب حق تقدم(از چپ به راست): - + * \ / ^

برخی از توابع پایه در متلب:

تابع	عملکرد	مثال
fix	عدد را به سمت عدد صحیح کوچکتر گرد می کند	fix(3.6)=3
round	عدد را به سمت نزدیکترین عدد صحیح گرد می کند	round(3.6)=4
ceil	عدد را به سمت عدد صحیح بزرگتر گرد می کند	ceil(3.1)=4
floor	عدد را به سمت عدد صحیح کوچکتر گرد می کند	floor(3.9)=3
log	محاسبه لگاریتم طبیعی (پایه e)	
log10	محاسبه لگاریتم در پایه 10	
exp	محایبه تابع نمایی	exp(6)
abs	تابع قدر مطلق	abs(-6)=6
angle	زاویه فاز یک عدد مختلط را محاسبه می کند	angle(3+4i)=0.9273
conj	مزدوج یک عدد مختلط را محاسبه می کند	conj(2-3i)=2+3i
sin		
cos		
tan		
cot		
nthroot	رادیکال یک عدد با فرجه دلخواه را محاسبه می کند	nthroot(27,3)=3

>>sin(30)

ans = -0.9880

```
>> sin(30*pi/180)

ans =

    0.5000
```

نکته: ورودی توابع مثلثاتی به رادیان هستند. برای تبدیل درجه به رادیان باید عدد را در $\frac{\pi}{180}$ ضرب کرد.

۱-۵-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.

نکاتی در مورد فضای کاری متلب:

۱-۵-۱-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 :

>> clear تمامی متغیرها از حافظه پاک می شوند

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

۱-۵-۲-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.

۱-۵-۳- 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 'b‘
```

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.

۱-۶-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.

۱-۶-۱- مراحل ایجاد فایلهای متنی

۱. باز کردن یک فایل جدید در ویرایشگر متلب:

File>New>m-file

۱. تایپ کردن دستورات متلب در فایل مذکور

۲. ذخیره کردن فایل با نامی مشخص:

File>Save As…

۱-۶-۲-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
```

```
c=29.3750
```

۱-۶-۳- توابع و دستورات مفید در فایلهای m

۱. 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
```

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

مثال:

```
n=input('Please enter a number')
-------------------------------------------------
Please enter a number: 10
n=      10
```

فصل دوم:

آرایه ها در MATLAB

۲-۱- ایجاد آرایه

روشهای ایجاد آرایه:

۱. با استفاده از علائم ; ، , و [ ]
۲. با استفاده از علامت :
۳. با استفاده از توابع linspace
۴. با استفاده از ترکیبی از روشهای فوق

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
```

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
```

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.

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
```

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

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(1,3,6)

x=

1   1.4   1.8   2.2   2.6   3
```

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
```

۲-۱-۴- ماتریسهای ویژه

- *[ ] : ماتریس تهی*
- *eye : یک ماتریس همانی با ابعاد داده شده ایجاد می‌کند(ماتریس قطری با درآیه های یک)*
- *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 تصادفی به ابعاد داده شده با عناصر بین صفر و یک تولید می کند*

مثال:

```
>>ones(2,3)

ans =

   1    1    1

   1    1    1

>>ones(2)

ans =

   1    1

   1    1
```

۲-۲- عملیات ریاضی بر روی آرایه‌ها

۱. *عملیات اسکالر-آرایه:* * , \ , / , ^ , + , -

۲. *عملیات عنصری:* .* , .\ , ./ , .^ , .+ , .-

۳. *عملیات ماتریسی:* * , \ , / , ^ , + , -

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
```

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
```

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
```

۲-۳- معکوس یک ماتریس:

برای محاسبه معکوس یک ماتریس از تابع inv استفاده می شود.

مثال:

$$\begin{cases} 4x+3y=5 \\ 2x-3y=7 \end{cases}$$

```
>>inv(A)*b
ans=
2
-1
```

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
```

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];
>>b=a(2:3,2:3)
b=
5       6
8       9
>>c=a(:,2)
c=
2
5
8
```

```
>>c=a(2,:)

c=

4       5       6
```

```
>>l=a(1:end,end)

l=

3

6

9
```

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
```

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

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

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

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*

مثال:

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

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

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

```
>>[r , c] = size(a)
r=2
c= 4
```

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
```

۲-۹-چند تابع برای دستکاری آرایه‌ها

- *flipud: ماتریس را حول محور افقی ۱۸۰ درجه می‌چرخاند.*
- *fliplr: ماتریس را حول محور عمودی ۱۸۰ درجه می‌چرخاند*
- *rot90 : ماتریس را در جهت مثلثاتی ۹۰ درجه می‌چرخاند*
- *diag: در صورتیکه بر روی یک ماتریس بکاربرده شود، قطر اصلی ماتریس را استخراج می‌کند. اما اگر بر روی یک بردار بکار رود، ماتریسی قطری با عناصر آن بردار می‌سازد*

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

```
>> b=flipud(a)

b =

     7     8     9
     4     5     6
     1     2     3

>> b=fliplr(a)

b =

     3     2     1
     6     5     4
     9     8     7

>> b=rot90(a)

b =

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

```
b =

   1   3

   2   4

>> diag(a)

ans =

   1

   4

>> d=[1 2 3 4 5 6]

d =

   1   2   3   4   5   6

>> diag(d)

ans =

   1   0   0   0   0   0

   0   2   0   0   0   0

   0   0   3   0   0   0

   0   0   0   4   0   0

   0   0   0   0   5   0

   0   0   0   0   0   6
```

فصل سوم:

توابع و عملیات ماتریسی

، منطقی و رابطه ای

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.

مثال:

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

```
>>a=[1   2   3          >>b=[366 ; 804 ; 351];
   4   5   6
   7   8   0];
>> x= a \ b
x=
        25
        22
        99
```

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*

عملیات منطقی و رابطه‌ای

☐ *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:

<	کوچکتر
>	بزرگتر
<=	کوچکتر مساوی
>=	بزرگتر مساوی
==	مساوی (برابر)
~=	نا مساوی
&	و
|	یا

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.

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
```

مثال: استخراج عناصری از یک ماتریس که در شرط خاصی صدق می‌کنند

```
>> a = [1 , 2 , 3 ; 4 , 2 , 2 ; 1 , 10 , 0];
a=
     1     2     3
     4     2     2
     1    10     0
>> a4 = a .* (a >= 3)
a4=
     0     0     3
     4     0     0
     0    10     0
```

۴-۲- عملگرهای منطقی

و (ترکیب عطفی)	&
یا (ترکیب فصلی)	|
یا (مانع جمع)	xor
نقیض	~

مثال:

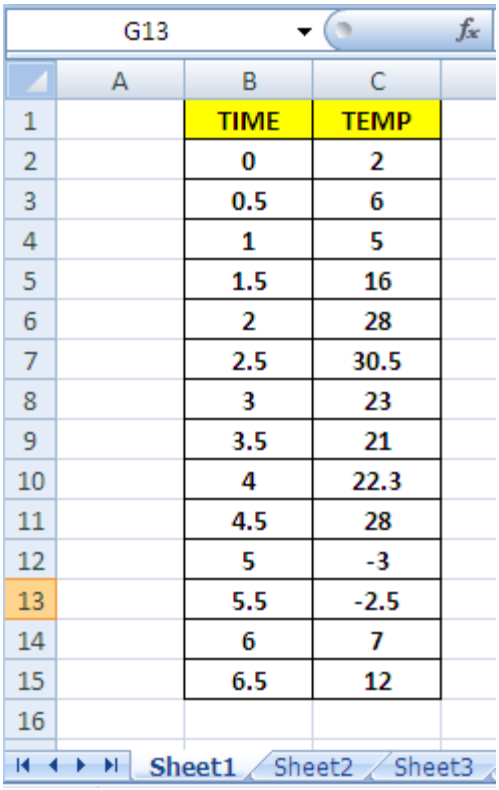
```
>> a= 1 : 9;
>> t = a > 3
     0     0     0     1     1     1     1     1     1
>> f = ~ ( a > 3)
     1     1     1     0     0     0     0     0     0
>> tf = ( a > 3) & (a <=7)
     0     0     0     1     1     1     1     0     0
```

مثال: *استاد یک درس می خواهد به دانشجویانی که نمره قبولی گرفته اند ۲ نمره اضافه کند . با استفاده از متلب کمک کنید تا نمرات نهایی زودتر آماده شود:*

```
>> marks=[17  15  9  8.5  13.5  16.45  5.75];

>> final=marks+(marks>=10)*2

final =

   19.0000   17.0000    9.0000    8.5000   15.5000   18.4500    5.7500
```

مثال: *اطلاعات یک آزمایش درون محدوده ای از فایل اکسل ثبت شده اند. ستون TIME نشان دهنده زمان است و ستون TEMP نشاندهنده دما، مشخص کنید که در چه زمانهایی دما بالای ۲۰ درجه و یا زیر صفر درجه بوده است.(این فایل را با نام test1.xlsx ایجاد نمایید)*

```
clc
s=xlsread('test1.xlsx',1,'B2:C15');
time=s(:,1)
temp=s(:,2)
n=(temp>20 | temp<0)
q=find(n==1) شماره اندیس را می دهد
time(q)
```



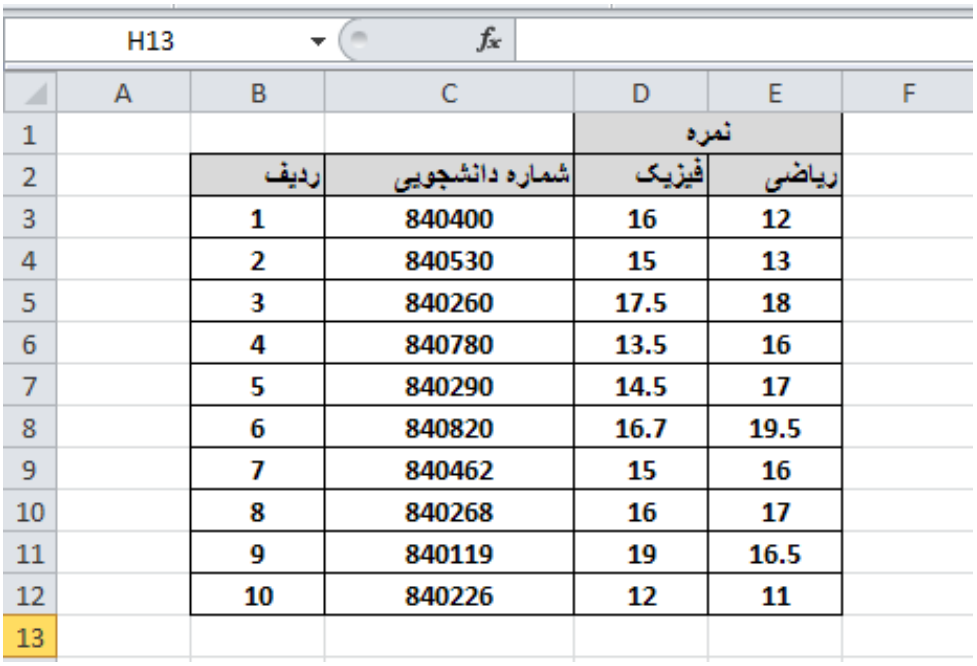
	A	B	C
1		TIME	TEMP
2		0	2
3		0.5	6
4		1	5
5		1.5	16
6		2	28
7		2.5	30.5
8		3	23
9		3.5	21
10		4	22.3
11		4.5	28
12		5	-3
13		5.5	-2.5
14		6	7
15		6.5	12
16			

مثال: طبق جدول زیر، از فهرست داده شده اطلاعات زیر را بدست آورید:

۱- شماره دانشجویانی که نمره ریاضیشان بالای ۱۶ است.

۲- شماره دانشجویانی که ریاضیشان بالای ۱۶ و فیزیکشان زیر ۱۷ است.

۳- شماره دانشجویانی که ریاضیشان یا فیزیکشان بالای ۱۵ است.



ردیف	شماره دانشجویی	نمره	
		فیزیک	ریاضی
1	840400	16	12
2	840530	15	13
3	840260	17.5	18
4	840780	13.5	16
5	840290	14.5	17
6	840820	16.7	19.5
7	840462	15	16
8	840268	16	17
9	840119	19	16.5
10	840226	12	11

```
numbers=xlsread('ex.xlsx',1,'C3:C12');
physic=xlsread('ex.xlsx',1,'D3:D12');
math=xlsread('ex.xlsx',1,'E3:E12');
a=find(math>16); شماره اندیس را می دهد
numbers(a)
b=find(math>16 & physic<17);
numbers(b)
c=find(math>15 | physic>15);
numbers(c)
```

فصل چهارم:

کار بر روی رشته ها

(متن)

۴-۱-رشته‌های کاراکتری

برای تعریف رشته‌های کاراکتری در متلب از علامت ‘ ’ استفاده می‌شود:

مثال:

```
>> s=‘This is a character string’;
>> size(s)
ans=
     1    26
```

نکته: در متلب رشته‌های کاراکتری نیز بعنوان ماتریس شناخته می‌شوند بطوریکه هر کاراکتر یک عنصر ماتریس محسوب می‌شود.

۴-۲-نمایش کد اسکی کاراکترها: تابع abs

برای نمایش کد اسکی یک رشته می‌توان از تابع abs متلب استفاده کرد:

```
>> s= ‘Hello’
>> u=abs(s)
u=
72  101   108   108  111
```

۴-۳-تبدیل کد اسکی به کاراکتر

برای تبدیل کد اسکی به کاراکتر از تابع char استفاده کنید.

```
>> s= ‘Hello’
>> u=abs(s)
```

```
u=

    72  101  108  108  111

>> sNew=char(u)

sNew=

    Hello
```

۴-۴-رفتار ماتریسی رشته‌ها

با رشته‌های کاراکتری متلب دقیقا می‌توان مانند ماتریسهای عددی رفتار کرد. مثلا می‌توان عملیات ریاضی را بر آنها اعمال کرد. در اینصورت متلب کد اسکی رشته را مورد استفاده قرار می‌دهد.

مثال: نمایش رشته از آخر به اول

```
>> s= 'Hello'

>> sInv=s( end : -1 : 1);

>>disp(sInv)

    olleH
```

۴-۷-گرفتن رشته در حین اجرای برنامه

برای گرفتن یک رشته از ورودی با استفاده از تابع input در حین اجرای برنامه دو روش را می‌توان بکار برد:

روش اول روش معمول استفاده از این تابع است. یعنی تابع مذکور را تنها با یک آرگومان ورودی بکار می‌بریم. در اینصورت در حین اجرا، باید رشته را در داخل ' ' قرار داد.

روش بهتر استفاده از تابع input با یک آرگومان دوم 's' می‌باشد که در اینصورت متلب ورودی کاربر را بعنوان رشته تلقی می‌کند حتی اگر یک عدد یا نام یک متغیر باشد.

مثال:

```
>>s=input('Please answer Yes or No: ')
```
روش اول

```
Please answer Yes or No: 'No'


s=

No

----------------------
>>s=input('Please answer Yes or No: ','s')
```
روش دوم

```
Please answer Yes or No: No

s=

No
```

۴-۸- سایر توابع کار با رشته‌ها

strcmp(s1,s2)	*درصورتیکه دو رشته یکسان باشند ۱ و درغیر اینصورت ۰ باز می‌گرداند:*
upper	*تمای حروف یک رشته را به حروف بزرگ تبدیل می‌کند:*
lower	*تمامی حروف یک رشته را به حروف کوچک تبدیل می‌کند:*
num2str	*تبدیل عدد به رشته عددی:*
str2num	*تبدیل رشته عددی به عدد:*
mat2str	*تبدیل ماتریسی از اعداد به رشته:*
eval	*اجرای فرمانی از متلب که بصورت رشته وارد شده باشد:*

مثال:

```
>> a=input('Enter <a> value= ');          enter <a> value= 12
>> disp(['You number is', num2str(a) , '  . Thank you!']);
Your number is 12 . Thank you!
```

فصل پنجم:

تصمیم گیری و کنترل روند

استفاده از حلقه‌ها و دستورات شرطی در متلب

۵-۱-حلقه for:

شکل کلی حلقه for در متلب بصورت زیر است:

```
for   نام متغیر =عدد شروع : عدد پایان
        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.

مثال:

```
A=0;
for n=1:20
        A=A+n;
End
B=A
```

در این مثال اعداد ۱ تا ۲۰ با همدیگر جمع شده و در متغیر A و B قرار می گیرد

```
A=0;
for n=1:2:20
        A=A+n;
        disp(A)
End
```

در این مثال اعداد فرد ۱ تا ۲۰ با هم جمع شده و در متغیر A قرار می گیرد و خود اعداد نیز نمایش داده می شوند

۵-۲- 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.

مثال:

```
a=1;
while  a<=10
    a=a+1;
    disp(a)
end
b=a
```

در این مثال تا زمانی که متغیر a به عدد بزرگتر از ۱۰ نرسیده باشد، حلقه تکرار می شود

۵-۳- 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;
```

مثال:

```
if a>10
X=1;
elseif a<5
X=2;
else
X=3;
end
```

در این مثال اگر متغیر a بزرگتر از عدد ۱۰ باشد مقدار x برابر ۱ و درصورتی که a کوچکتر از ۵ باشد مقدار x برابر ۲ و در غیراینصورت مقدار x برابر ۳ خواهد شد

۶-۳- ساختار switch-case

هرگاه بخواهیم به ازای مقادیر مختلف از یک متغیر عملیات مختلفی انجام شود. در این صورت استفاده از ساختار زیر مناسب است.

```
switch  نام متغیر
        Case   مقدار ۱
               دستورات
        Case   مقدار ۲
               دستورات
        Case   مقدار ۳
               دستورات
otherwise
        دستورات
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.

مثال: برنامه ای بنویسید که یک ماتریس از کاربر گرفته:

الف – کل ماتریس نمایش داده شود:

ب- اعضای مثبت را نمایش دهد:

ج- اعضای منفی را نمایش دهد:

```
clc
A=input('Please insert Matrix ?');
[m,n]=size(A);
total=m*n;
disp('-All Mtrix for show------------------')
% All Mtrix for show
for k=1:total
    str=['Num ' , num2str(k) , ':' , num2str(A(k))];
    disp(str);
end
disp('-Positive Mtrix----------------------')
% Positive Mtrix
for k=1:total
    if A(k)>0
    str2=['Num ' , num2str(k) , ':' , num2str(A(k))];
    disp(str2);
    end
end
disp('-Negative Mtrix----------------------')
% Negative Mtrix
for k=1:total
    if A(k)<0
    str3=['Num ' , num2str(k) , ':' , num2str(A(k))];
    disp(str3);
    end
end
```

خروجی برنامه:

```
Please insert Matrix ?    [2 3 4 -4 0 -8]
-All Mtrix for show------------------
Num 1:2
Num 2:3
Num 3:4
Num 4:-4
Num 5:0
Num 6:-8
-Positive Mtrix----------------------
Num 1:2
Num 2:3
Num 3:4
-Negative Mtrix----------------------
Num 4:-4
Num 6:-8
```

مثال: قسمت الف برنامه فوق را با حلقه While انجام دهید:

```
clc
A=input('Please insert Matrix ?');
[m,n]=size(A);
total=m*n;
f=1;
disp('-All Mtrix for show------------------')
% All Mtrix for show
while total>0
    str=['Num ' , num2str(f) , ':' , num2str(A(f))];
    disp(str);
    f=f+1;
    total=total-1;
end
```

مثال: برنامه ای بنویسید که یک ماتریس را گرفته و بجای اعضای غیر صفر آن صفر گذاشته و به جای صفرها به ترتیب از ۱ عددگذاری نماید.

```
clc
A=input('Please insert Matrix ?');
[m,n]=size(A);
total=m*n;
q=0;
for i=1:total
if A(i)~=0
    A(i)=0;
else
    q=q+1;
    A(i)=q;
end
end
disp(A)
```

```
Please insert Matrix ?[2 3 4 -4 0 -8]

     0     0     0     0     1     0
```

مثال: برنامه ای بنویسید که $\sum_{n=1}^{m} \frac{1}{n}$ *را با توجه به مقداری که کاربر برای* m *تعریف می کند محاسبه کرده و نمایش دهد:*

```
clc
m=input('insert final point?');
s=0;
for i=1:m
    s=s+1/i;
end
disp(s)
```

```
insert final point?  5

    2.2833
```

مثال: برنامه ای بنویسید که جدول ضرب ۱ تا ۵ را در یک ماتریس ایجاد و نمایش دهد:

```
clc
n=0;
for i=1:5
    for j=1:5
        n(i,j)=i*j;
    end
end
disp(n)
```

```
     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25
```

مثال: *برنامه ای بنویسید که زوج بودن یک عدد را بررسی نماید به گونه ای که پس از یکبار اجرا از کاربر بپرسد که آیا می خواهد عدد دیگری را امتحان کند؟ در صورت مثبت بودن پاسخ دوباره عددی از کاربر گرفته و زوج بودنش را بررسی نماید.*

```
clc
num=0;
rep='yes';
while rep =='yes'
    num=input('input 1 number>0=?');
    if mod(num,2)==0
        disp('number is even')
    else
        disp('number is odd')
    end
    rep=input('Try Again? (yes or no) ','s');
end
```

مثال: برنامه ای بنویسید که نمره داده شده را به معادل حرفی تبدیل نماید.

A	B	C	D	E	F
20-18	18-16	16-14	14-12	12-10	<10

راه حل اول:

```
n=input('number=?');


status=[n>=18&n<=20   n>=16&n<=18   n>=14&n<=16
        n>=12&n<=14   n>=10&n<=12   n<10];


q=find(status==1);
switch q
    case 1
        disp('A')
    case 2
        disp('B')
    case 3
        disp('C')
    case 4
        disp('D')
    case 5
        disp('E')
    case 6
        disp('F')
end
```

فصل ششم:

ایجاد توابع در MATLAB

(Function)

۶-۱-مزایای استفاده از توابع به جای فایلهای اسکریپت

۱. سرعت بالاتر

۲. صرفه‌جویی در حافظه کامپیوتر

۳. 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

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.

۶-۳- نحوه ایجاد توابع

تنها تفاوت ظاهری یک تابع و یک فایل m آن است که سطر اول یک تابع با کلمه کلیدی function شروع می‌شود و حتماً نیاز به ورودی دارد که شکل کلی آن بصورت زیر است:

function خروجی تابع = نام تابع (ورودی ها)

انجام محاسبات بر روی ورودی ها

خروجی تابع= پاسخ نهایی که برای نمایش محاسبه شده

۶-۴- 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.
- استفاده از (;) در انتهای تمامی دستوراتی که در تابع نوشته می شود ضروری است.
- متغیرهایی که درون تابع تعریف و یا استفاده می شوند کاملاً داخلی و درونی هستند و در محیط متلب تعریف و شناخته نمی شوند و ارتباطی به Workspace ندارد(در اصطلاح متغیرهای خصوصی یا Pritive هستند)
- در محیط نوشتن تابع از تمامی ابزارهای برنامه نویسی می توان استفاده کرد تنها چیزی که به آن نیاز ندارید input است. زیرا در همان ابتدای کار ورودی ها توسط کاربر وارد می شوند.
- پس از نوشتن تابع ، در هنگام ذخیره سازی خواهید دید که نام فایل به صورت همان نام تابع شما خواهد بود ، آن را به هیچ وجه تغییر ندهید.
- توابع می توانند بیش از یک خروجی داشته باشند.

مثال: تابعی بنویسید با نام **dist** *که مختصات دو نقطه را گرفته و فاصله بین آنها را محاسبه کند.*

```
function u=dist(x1,y1,x2,y2)
dx=x2-x1;
dy=y2-y1;
u=sqrt(dx^2+dy^2);
```

```
>> dist(1,1,3,3)

ans =

    2.8284
```

مثال: تابعی بنویسید با نام Complete که یک عدد را گرفته و مشخص کند که آن عدد کامل است یا نه؟

(راهنمایی: عددی کامل است که مجموع مقسوم علیه هایش با خودش برابر باشد،برای مثال مقسوم علیه های ۶ اعداد۱،۲و۳ هستند و از آنجا که ۶=۳+۲+۱ پس ۶ یک عدد کامل است.)

```
function d=Compelete(n)
s=0;
for i=1:n-1
    if mod(n,i)==0
        s=s+i;
    end
end
if n==s
    show='Number is Compelete!';
else
    show='Number is not Compelete!';
end
d=show;
```

```
>> Compelete(6)

ans =

Number is Compelete!
```

مثال: تابعی بنویسید با نام dist2 که مختصات دو نقطه را گرفته و فاصله و زاویه بین دو نقطه را محاسبه کند.

```
function [u,w]=dist2(x1,y1,x2,y2)
u=sqrt((x2-x1)^2+(y2-y1)^2);
w=atan((y2-y1)/(x2-x1));
```

زاویه برحسب رادیان بدست آمده

```
>> [a,b]=dist2(1,1,3,3)

a =

    2.8284

b =

    0.7854
```

فصل هفتم:

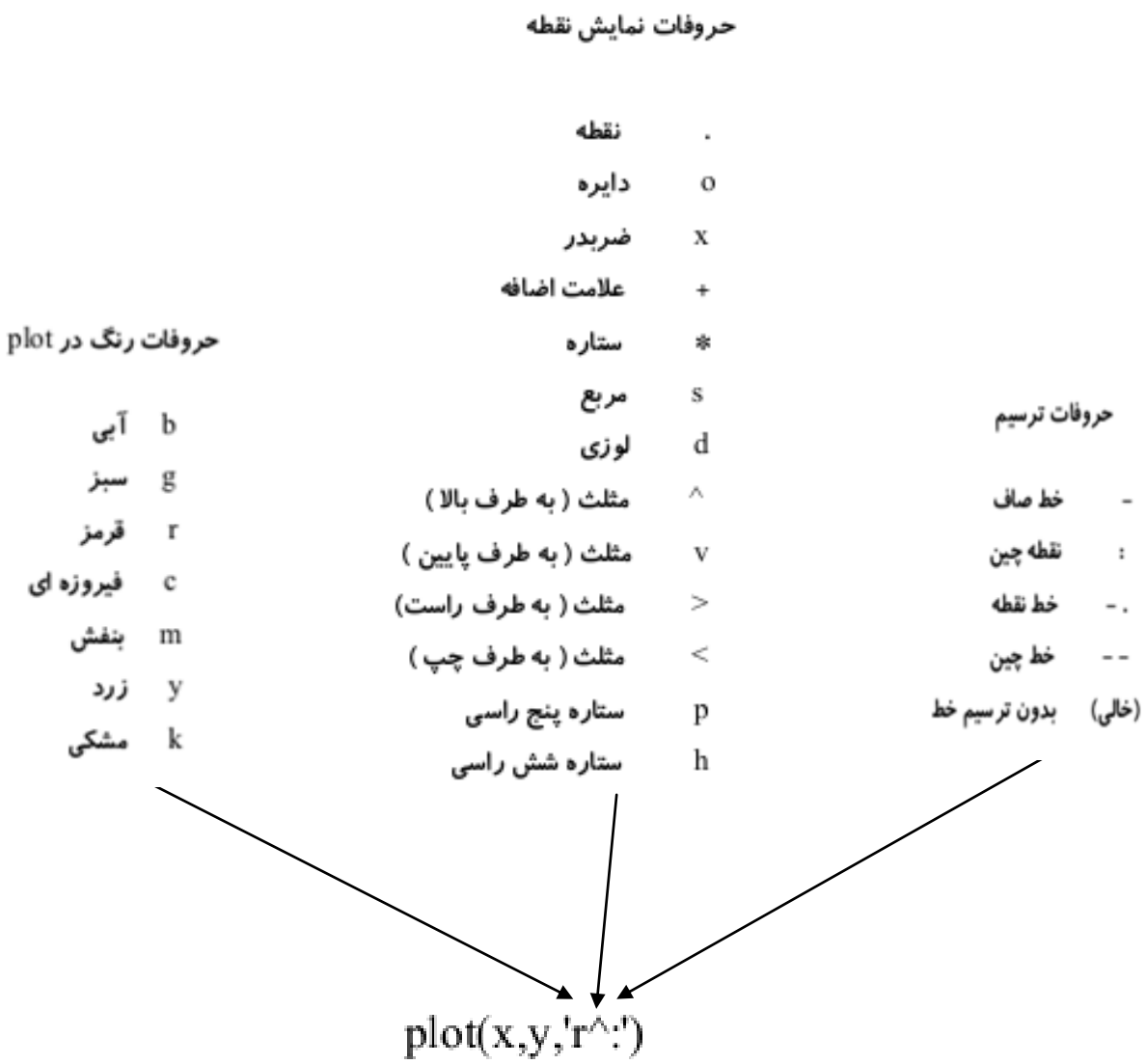
نمودارهای دوبعدی

نمودارهای دو بعدی

۷-۱-تابع plot

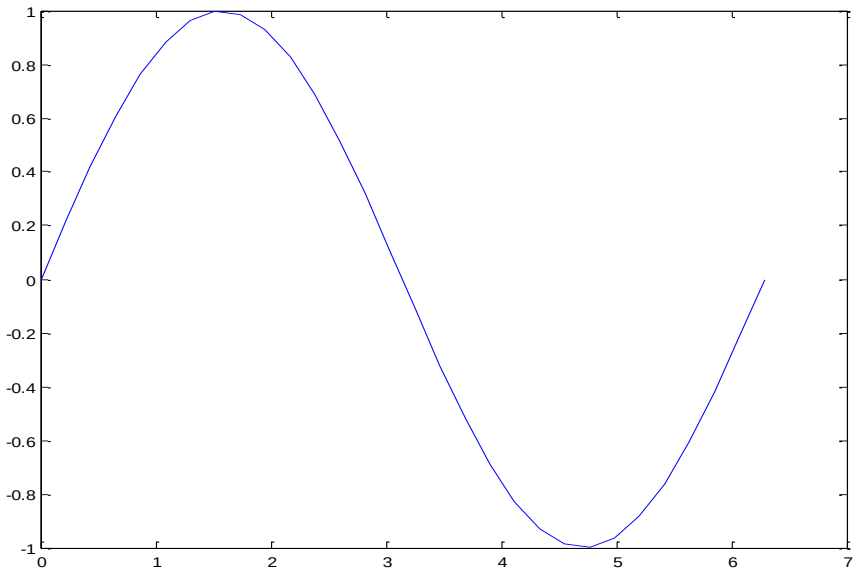
شکل کلی:

plot (x1,y1,'رنگ،نقطه،ترسیم',x2,y2, 'رنگ،نقطه،ترسیم',...)



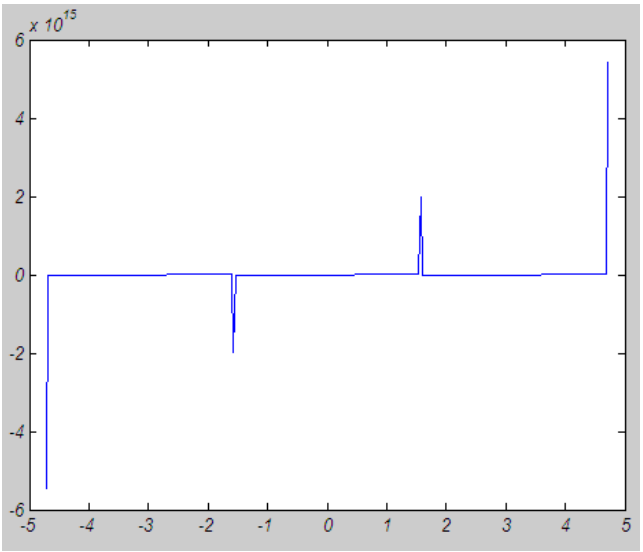
مثال:

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

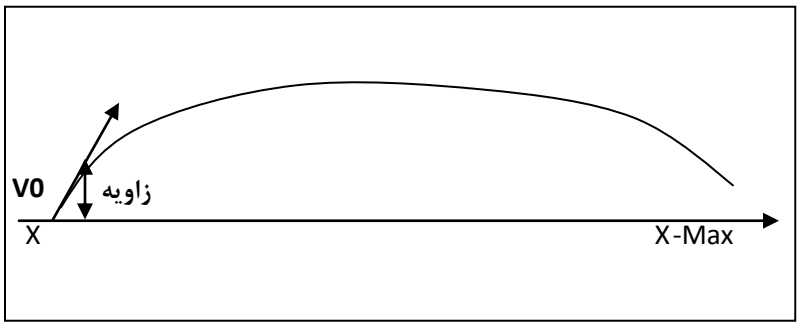


مثال: تابع تانژانت x *را در بازه* $[\frac{-3}{2}\pi, \frac{3}{2}\pi]$ *رسم نمایید:*

```
>> x=-3*pi/2 : pi/100 : 3*pi/2;
>> y=tan(x);
>> plot(x,y)
```



مثال: برنامه ای بنویسید که سرعت اولیه و زاویه پرتاب یک گلوله توپ را از کاربر گرفته و نمودار تغییر مکان آن را برحسب زمان رسم کرده و مسیر پیوده شده را محاسبه کند.



اگر پرتابه ای با سرعت v_0 و زاویه θ پرتاب شود، با فرمولهای زیر می توان حرکت آن را توضیح داد:

۱-مدت زمانی که طول می کشد تا پرتابه دوباره به زمین برگردد:

$$t_g = \frac{2v_0 \sin\theta}{g}$$

۲-مسافت پیوده شده:

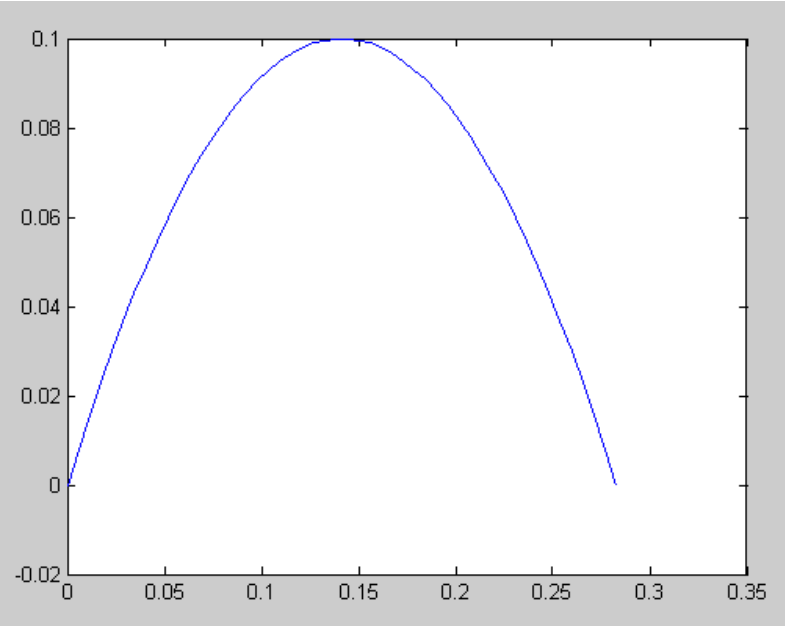
$$x_{max} = v_0 t_g \cos\theta$$

۳-فاصله گلوله از سطح زمین:

$$y(t) = v_0 t \sin\theta - \frac{1}{2} g t^2$$

```
clc
vo=input('initial velocity (Vo)=?');
theta=input('angle =? (Degrees)');
g=input('g=?');
th=theta*pi/180;
tg=2*vo*sin(th)/g;
xmax=vo*tg*cos(th);
tvec=linspace(0,tg,50);
y=vo*sin(th)*tvec-1/2*g*tvec.^2;
disp('tg=')
disp(tg)
disp('xmax is =')
disp(xmax)
plot(tvec,y)
```

```
initial velocity (Vo)=?  2
launch angle =? (Degrees)  45
g=?10
tg=
   0.2828
xmax is =
   0.4000
```



۷-۲- رسم چند نمودار مجزا در یک پنجره شکل

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

شکل کلی:

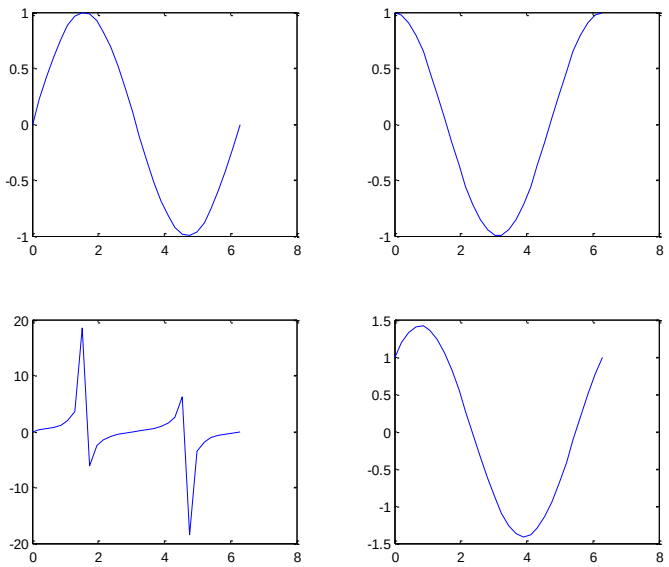
subplot(m ,n , p)

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.

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));
```



۷-۳- 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.

۷-۴- 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.

```
clc
x=[0:10:90];
y=sin(x);
z=cos(x);

figure;
plot(x,y);
title('Sin diagram');
xlabel('x');
ylabel('y');

figure;
plot(x,z);
title('Cos diagram');
xlabel('x');
ylabel('y');
```

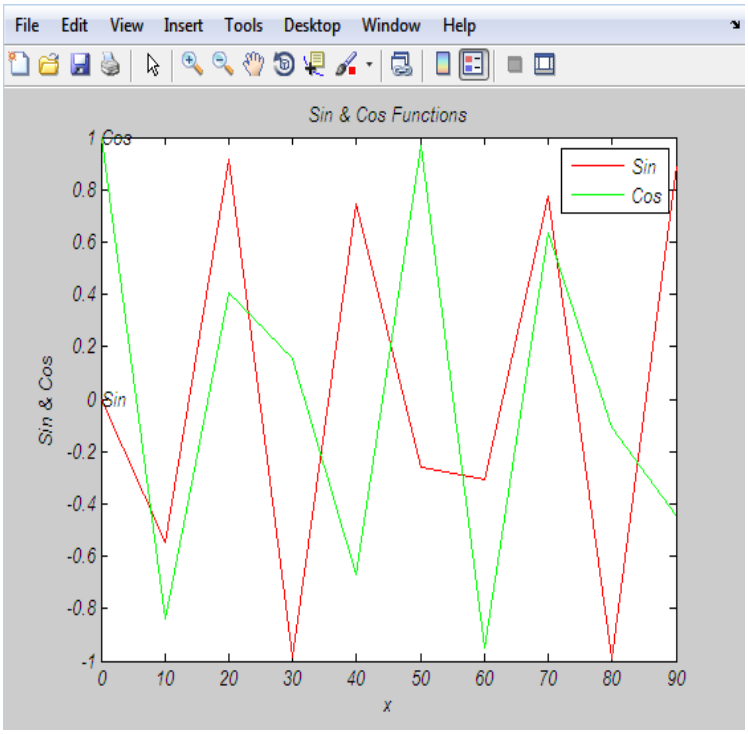
۷-۵- افزودن متن به نمودار

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

```
>> text(x,y,'رشته متنی')
```

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

```
clc
x=(0:10:90);
y=sin(x);
z=cos(x);

plot(x,y,'r',x,z,'g');

title('Sin & Cos Functions');
xlabel('x');
ylabel('Sin & Cos');
legend('Sin','Cos');
text(0,0,'Sin');
text(0,1,'Cos');
```

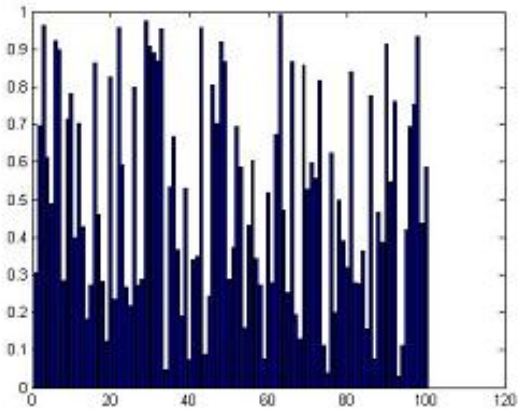


نمودارهای آماری:

Bar نمودار میله ای

این دستور نمودار میله ای یک مجموعه را رسم می کند .

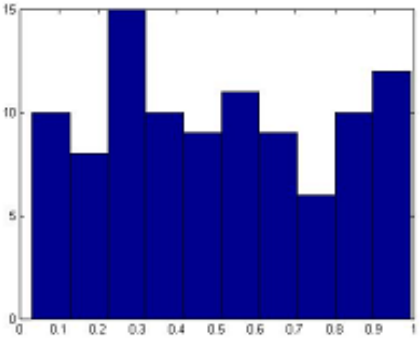
```
>>a=rand(1,100);
>>bar(a)
```



Hist نمودار فراوانی

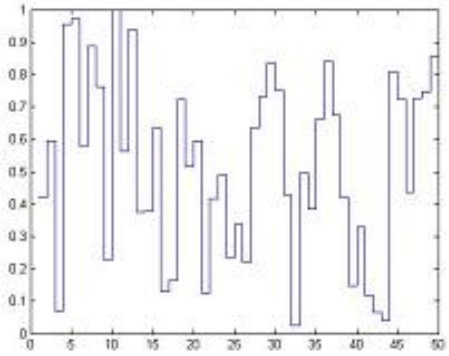
نمودار هیستوگرام مربوط به مجموعه را رسم میکند .

```
>>a=rand(1,100);
>>hist(a)
```



stairs نمودار پله ای

```
>>a=rand(50,1);
>>stairs(a)
```



فصل هشتم:

چندجمله ای ها

۸-۱-تعریف یک چندجمله‌ای در متلب

در متلب یک چند جمله‌ای توسط یک بردار سطری تعریف می‌شود.

مثال: $p = x^2 + 2x - 3$

```
>>p=[1 2 -3]
```

۸-۲- یافتن ریشه‌های چند جمله‌ای

با استفاده از تابع roots می‌توان ریشه‌های یک چند جمله‌ای را بدست آورد:

مثال:

```
>> r= roots(p)
p=
-3
1
```

۸-۳- یافتن یک چندجمله‌ای با استفاده از ریشه‌هایش

با استفاده از تابع poly می‌توان یک چند جمله‌ای را از روی ریشه‌هایش بدست آورد.

نکته: بر خلاف خود چندجمله‌ای ریشه‌های چند جمله‌ای باید بصورت یک بردار ستونی تعریف شوند.

مثال:

```
>> r=[-3,1,4]
>> p=poly(r)
p =
1   -2   -11   12
```

$p = x^4 - 2x^3 - 11x^2 + 12$

۸-۴- ضرب چند جمله‌ایها

بمنظور ضرب دو چند جمله‌ای می‌توان از تابع conv استفاده کرد.

مثال:

```
>> a= [1 2 3 4]; b= [1 4 9 16];
>> c= conv(a , b)
c=
     1  6  20  50  75  84  64
```

۸-۵- جمع و تفریق چندجمله‌ایها

برای اینکه بتوان دو بردار را با یکدیگر جمع یا تفریق کرد باید آن دو بردار هم طول باشند. لذا در صورت لزوم باید ضرایبی که تنها در یکی از چند جمله‌آیها وجود دارد را در چند دوم برابر با صفر قرار داد تا دو چند جمله‌ای هم طول شوند.

مثال:

```
>>p1= [4 5 3 2]
>>p2= [0 5 2 0]
>>p_sum=p1+p2
p_sum=
     4  10  5  2
```

۸-۶- تقسیم چند جمله‌ایها

با تابع deconv می‌توان دو چندجمله‌ای را بر یکدیگر تقسیم کرد. این تابع دو آرگومان خروجی می‌گیرد که اولی خارج قسمت و دومی باقیمانده تقسیم خواهد بود.

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

>> b=[ 2 3 4];

>> [q , r] = deconv( a , b )

q =

   0.5000    0.2500    0.1250    1.3125

r =

         0         0         0         0    0.5625    0.7500
```

۱۰-۷- مشتق چندجمله‌ای

با استفاده از تابع polyder می‌توان مشتق یک چند جمله‌ای را بدست آورد

مثال:

```
>> g = [1 6 20 48 69 72 44]

>> h= polyder(g)

h=

         6  30  80  144  138  72
```

۸-۱۰- محاسبه چندجمله‌ای

بمنظور محاسبه مقادیر چندجمله‌ای در یک یا چند نقطه از تابع polyval می‌توان استفاده کرد.

```
>>a=[1 2 3];
>>polyval(a,2)

Ans=
11
```

معادله x^2+2*x+3 به ازای x=2 محاسبه شده است .

متغیرهای پارامتری:

از ابتدای جزوه تا اینجا با توابع عددی و متغیرهای عددی سروکار داشتیم. اما بسیاری از محاسبات ریاضی بصورت پارامتری انجام می شود.

برای تعریف یک متغیر حرفی از دستور sym استفاده می شود. روش استفاده از آن برای دو متغیر x و y بصورت زیر است:

```
>> x= sym('x');
>>y= sym('y');
```

```
>> syms x y
```

۸-۱۱- حد چندجمله ای های پارامتری

برای محاسبه چند جمله ای از تابع limit استفاده می شود:

Limit(عبارت پارامتری , متغیر حدگیری , نقطه محاسبه حد)

مثال: $\lim_{x\to 0}\frac{\sin x}{x}$ *را محاسبه نمایید:*

```
>>limit (sin(x)/x,x,0)

ans=1
```

نکته: اگر حد دارای صورتهای مبهم باشد متلب خود مراحل رفع ابهام را انجام می دهد:

```
>> x=sym('x');
>>limit(1/x,x,+inf)
ans=0
```

۱۲-۸- مشتق چندجمله ای های پارامتری

diff(عبارت پارامتری , متغیر مشتق گیری , مرتبه مشتق)

```
>> x=sym('x');
r=1*x^3+6*x^2+20*x+48;
>> diff(r,x,1)
 ans =
 3*x^2 + 12*x + 20
 >> diff(r,x,2)
 ans =
 6*x + 12
```

۱۳-۹- انتگرال چندجمله ای های پارامتری

با استفاده از تابع polyint می‌توان انتگرال یک چند جمله‌ای را بدست آورد

int(عبارت پارامتری , متغیر انتگرال گیری , حدپایین , حدبالا)

مثال:

```
>> x=sym('x');
>> int(x^2,x)
ans =
x^3/3
```

```
>> syms x;
>> b=int(sin(x),x,pi/2,pi)
b =
1
```

```
>> syms w;
>> int(exp(w),w,0,2)
ans =
exp(2) - 1
```

۱۴-۹-تابع solve:

از این تابع پرکاربرد برای حل معادلات استفاده می شود یعنی این تابع عبارت پارامتری را برابر صفر قرار داده و آن را برحسب متغیر داده شده حل می کند.

Solve(نام متغیر , عبارت پارامتری)

مثال:

```
>>solve (2*x+8,x)

ans=-4

>>solve(x^2*y-5*y+6,y)

ans=

-6 / (x^2 – 5)
```

فصل نهم:

برازش منحنی و درونیابی

۹-۱-برازش منحنی- تابع polyfit

با استفاده از تابع polyfit می‌توان بهترین منحنی گذرنده از چند نقطه را بدست آورد. این تابع چند جمله‌ای معرف منحنی فوق را بعنوان آرگومان خروجی باز می‌گرداند. شکل کلی استفاده از این تابع بصورت زیر است:

```
P= polyfit(x , y , n)
```

که در این رابطه، x و y نقاط معلوم و n درجه چندجمله‌ای مطلوب است.

مثال:

```
>> x= [ 1 2 5 7]; y=[10 22 48 75];

>> p= polyfit(x,y,1)

p=      10.45   -0.4396
```

مثال:

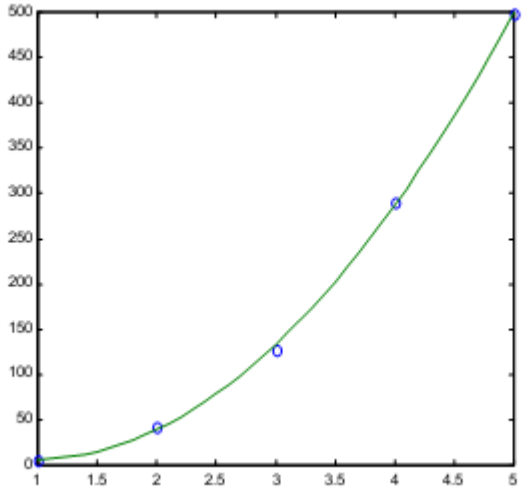
```
» x=[1  2  3  4  5];
» y=[5.5  43.1  128  290.7  498.4];
```

دستور زیر ضرایب بهترین چند جمله ای درجه سوم را محاسبه می کند که از بین نقاط فوق می گذرد:

```
» p=polyfit(x,y,3)
p =
  -0.1917   31.5821  -60.3262   35.3400
```

حال می توانید برای مقایسه منحنی محاسبه شده و داده های اولیه را در یک نمودار رسم کنید:

```
» x2=1:.1:5;
» y2=polyval(p,x2);
» plot(x,y,'o',x2,y2)
```



۹-۲- درونیابی یک بعدی: تابع interp1

تفاوت درونیابی با برازش آن است که در برازش منحنی لزوما خود نقاط اولیه بر روی منحنی برازش شده قرار ندارند اما در درونیابی، نقاط اولیه جزیی از منحنی مورد استفاده برای درونیابی می‌باشند. شکل کلی استفاده از تابع interp1 بصورت زیر است:

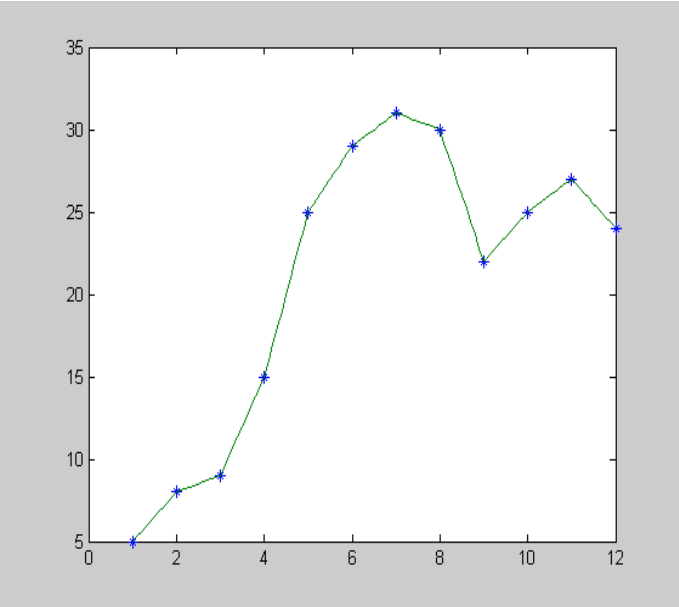
```
y_new = interp1 (x , y, x_new , ['method'])
```

که در این رابطه x,y نقاط اولیه، x_new مقادیری از x است که باید مقادیر y ان درونیابی شوند و y_new مقادیر درونیابی شده می‌باشند. method می‌تواند یکی از مقادیر زیر باشد:

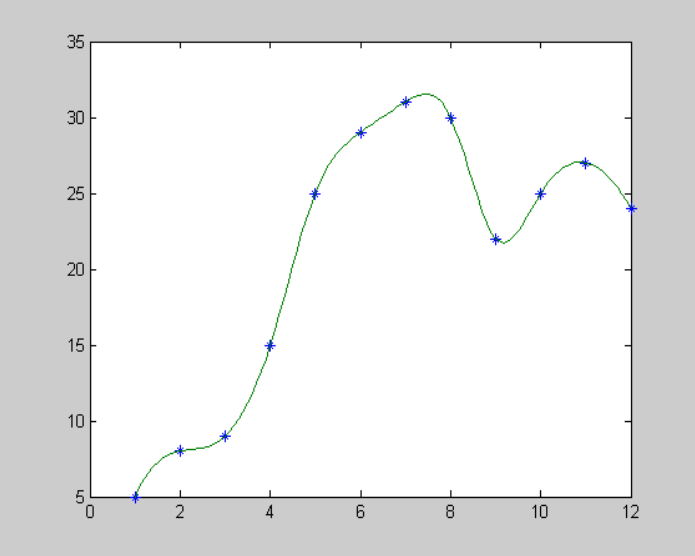
'nearest' , ' linear', ' spline', ' pchip', ' cubic', 'cubic5v'

مثال:

```
>> h = 1:12;
>> temps = [5 8 9 15 25 29 31 30 22 25 27 24];
>> plot ( h, temps); % عملا درونیابی خطی بکار برده می‌شود
>> h_new=1.5;
>> t_new = interp1(h , temps , h_new)
t_new=
        6.5;
>> h_new2 = 1: 0.1 : 12;
>> t_new2 = interp1(h , temps , h_new2);
>> plot( h, temps , '*' , h_new2 , t_new2);
```




```
>> t_spline = interp1(h , temps , h_new2 , 'spline');

>> plot(h , temps , '*' , h_new2 , t_spline);
```



۹-۳- درونیابی دو بعدی: تابع interp2

شکل کلی استفاد از تابع:

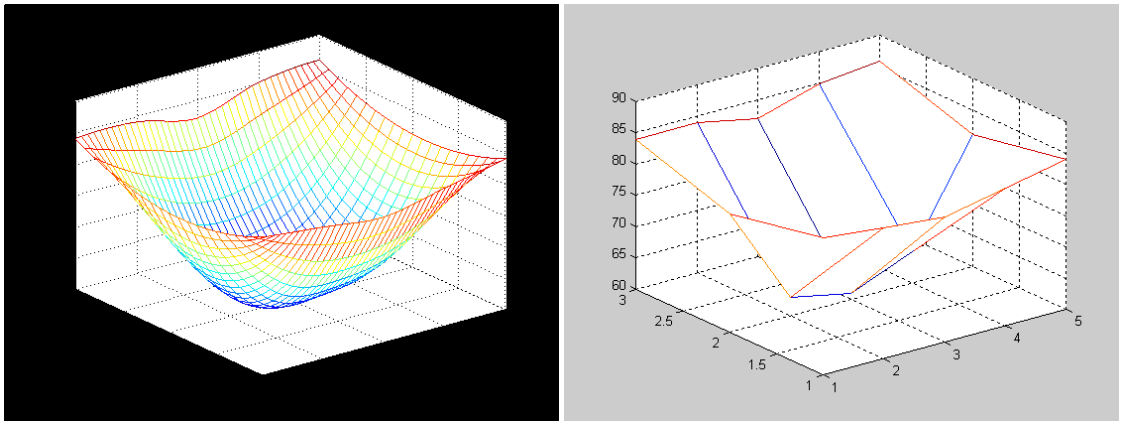
```
z_new = interp2(x, y, z, x_new, y_new, ['method'])
```

method می‌تواند یکی از مقادیر زیر باشد:

'nearest' , 'linear', 'spline' , 'cubic'

مثال:

```
>> w=1:5; d=1:3;
>> t = [82 81 80 82 84
        79 63 61 65 81
       84 84 82 85 86];
>> w_new =1:0.1:5;
>> d_new  =1:0.1:3;
>> t_new = interp2(w, d' , t , w_new , d_new' ,'cubic');
>> mesh(w,d,t);
>> figure;mesh(w_new,d_new,t_new)
```



فصل دهم:

نمودارهای سه بعدی

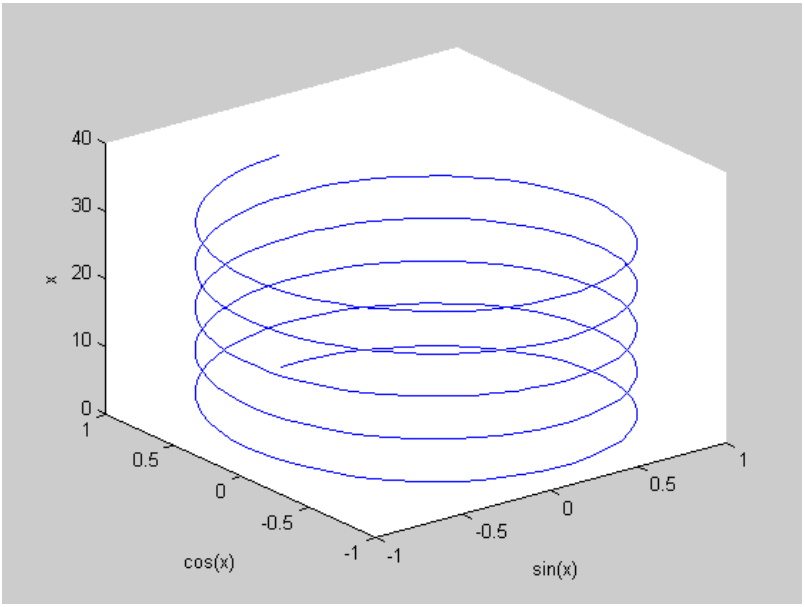
۱۰-۱- خمهای فضایی- تابع plot3

با استفاده از تابع plot3 در متلب می‌توان یک منحنی را در فضای سه‌بعدی ترسیم کرد. روش استفاده از این تابع بسیار شبیه تابع plot است. جز اینکه بازای هر منحنی به سه بردار هم طول نیاز است.

مثال: رسم یک فنر با شعاع برابر با یک:

$$\begin{cases} x=t \\ y=\sin(t) \qquad t \in R \\ z=\cos(t) \end{cases}$$

```
>>t=0: pi/50:10*pi;
>>plot3(sin(t) , cos(t) , t); xlabel('sin(x)'); ylabel('cos(x)'); zlabel('x')
```



۱۰-۲-تغییر زاویه دید

با استفاده از دکمه فشاری Rotate 3D بر روی هر پنجره شکل متلب و یا با استفاده از دستور view می توان زاویه دید را تغییر داد. همچنین در منوی Tools با استفاده از زیرمنوهای Camera Motion می توان در فضای سه‌بعدی حرکت کرد.

شکل کلی استفاده از دستور view بصورت زیر است:

```
view([az , el])
```

که در این رابطه az و el بترتیب زاویه دوربین نسبت به صفحه xy و بخش منفی محور y است.

۱۰-۳-نمودارهای شبکه‌ای: توابع mesh, meshc, meshz

با استفاده از این توابع می‌توان سطوح شبکه‌ای(یا توری) ایجاد کرد.شکل کلی استفاده از تابع mesh بصورت زیراست:

```
mesh(x,y,z)
```

که در این رابطه z تابعی دو متغیره از متغیرهای x و y می‌باشد. بنابراین لازم است که z یک ماتریس دو بعدی باشد که تعداد سطرهای آن برابر با تعداد عناصر y و تعداد ستونهایش برابر با تعداد عناصر x باشد. x و y باید بردار باشند اگرچه می توانند ماتریسهایی هم‌بعد نیز باشند بدینصورت که بردار x به تعداد عناصر بردار y بصورت سطری تکرار شود و بردار y به تعداد عناصر x بصورت ستونی تکرار گردد. که در اینصورت دو ماتریس هم‌بعد خواهیم داشت. تابع meshgrid می تواند این عمل را انجام دهد:

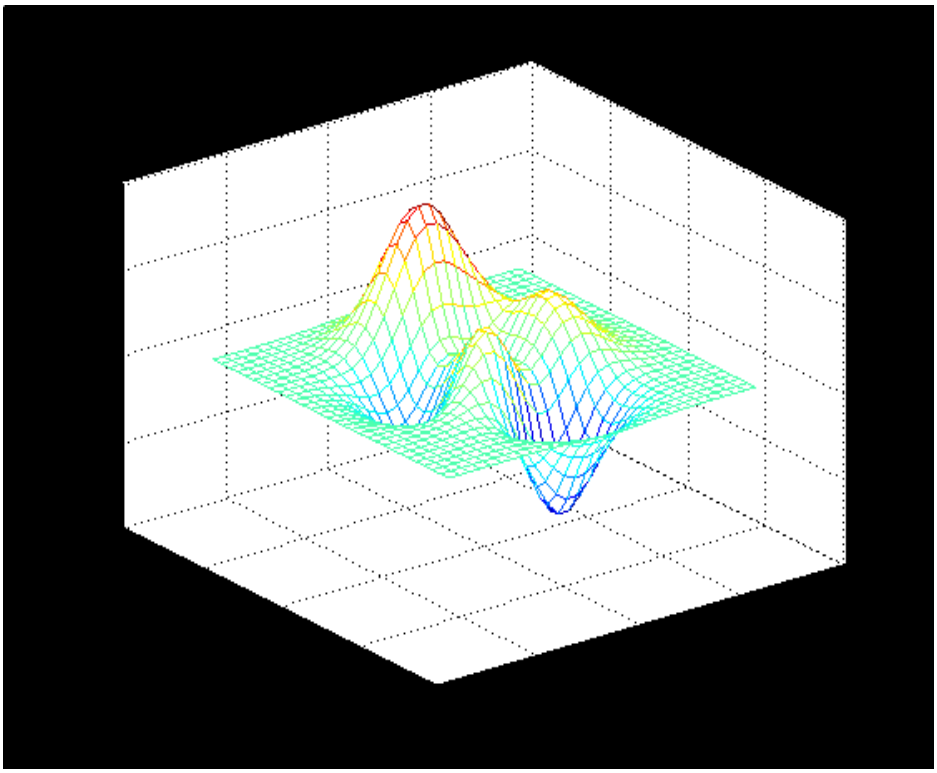
```
[x_new,y_new]=meshgrid(x,y);
```

تابع meshc علاوه بر نمودار شبکه‌ای، نمودارهای تراز را نیز رسم می‌کند.

تابع meshz دیواره‌هایی را در پایین نمودار به سمت صفحه x-y رسم می‌کند.

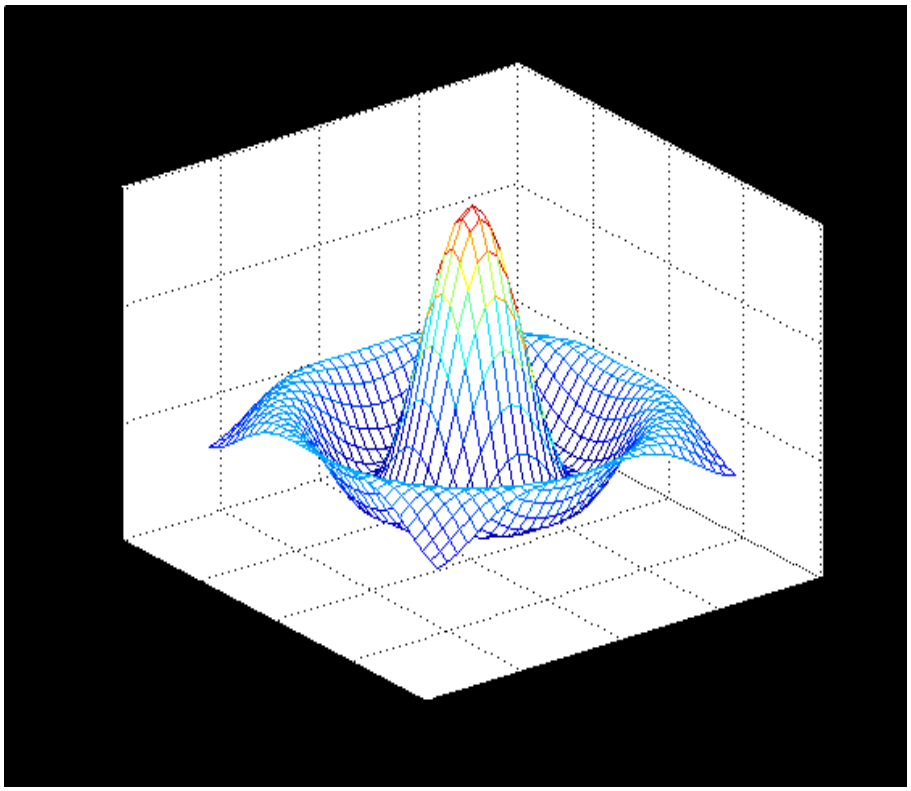
مثال : تابع peaks یکی از توابع متلب است که یک مدل ریاضی از پیش‌تعریف شده را ایجاد می‌کند:

```
>> [x ,y,z]= peaks(30);

>> mesh(x,y,z);
```



مثال : رسم یک تابع دو بعدی به فرمول $z=\frac{\sin(r)}{r}$ *که* $r=\sqrt{X^2 + Y^2}$

```
>>x=-7.5: 0.5: 7.5; y=x;

>> [x_new,y_new]=meshgrid(x,y);

>> r = sqrt(x_new.^2 + y_new.^2) + eps;

>>z=sin(r) ./ r; mesh(x_new , y_new , z) یا: mesh(x,y,z);
```



موفق و پیروز باشید

AbbasMohammadiSite.Mihanblog.com

oxinauto@gmail.com

۰۹۱۶۳۰۷۷۸۶۵