بسم الله الرحمن الرحیم

درس نرم افزارهای ریاضی، آشنایی با نرم‌افزار متلب و لاتک

مدرس: نجمه حسینی منجزی

دانشگاه اصفهان، دانشکده ریاضی و آمار، گروه ریاضیات کاربردی و علوم کامپیوتر

بخش ۷

بهمن ۱۴۰۱

فهرست مطالب

۱ معرفی چند تابع کاربردی

گفتیم MATLAB انواع مختلفی از متغیرها را پشتیبانی می‌کند. حال در اینجا می‌خواهیم چند نکته دیگر را اضافه کنیم.

نرم افزار MATLAB از انواع مختلف متغیر پشتیبانی می‌کند از جمله می‌توان به موارد زیر اشاره کرد

Data Type	Description
int8	8-bit signed integer
uint8	8-bit unsigned integer
int16	16-bit signed integer
uint16	16-bit unsigned integer
int32	32-bit signed integer
uint32	32-bit unsigned integer
int64	64-bit signed integer

و

uint64	64-bit unsigned integer
single	single precision numerical data
double	double precision numerical data
logical	logical values of 1 or 0, represent true and false respectively
char	character data (strings are stored as vector of characters)

برای مثال می‌توان به صورت زیر تعریف کرد

Command Window

```
>> str = 'Hello World!'

str =

Hello World!

>> n = 2345

n =

        2345

>> d = double(n)

d =

        2345

>> un = uint32(789.50)

un =

         790
```

و

Command Window

```
>> rn = 5678.92347

rn =

   5.6789e+03

>> c = int32(rn)

c =

        5679

>> un=uint8(-326.259)

un =

    0

>> c1=int8(326.25)

c1 =

  127
```

دقت کنید این توابع دارای جزئیاتی می‌باشند که هنگام استفاده از آنها باید به این موارد دقت کرد

Command Window

```
>> c1=int8(126.25)

c1 =

  126

>> c1=int8(126.95)

c1 =

  127

>> c1=int8(126.50)

c1 =

  127

>> c1=int8(126.49)

c1 =

  126
```

و

```
Command Window
>> c1=int8(326.25)

c1 =

  127

>> c1=int16(326.25)

c1 =

    326

>> 2^8

ans =

   256

>> 2^16

ans =

       65536
```

توابع مختلفی وجود دارند که با استفاده از آنها می‌توانیم انواع مختلف این متغیرها را به هم تبدیل کنیم. به عنوان مثال داریم

int2str	Convert integer data to string
mat2str	Convert matrix to string
num2str	Convert number to string
str2double	Convert string to double-precision value

str2num	Convert string to number

از طرف دیگر توابع مختلفی وجود دارند که توسط آنها می‌توانیم مشخص کنیم یک متغیر دلخواه به کدام دسته تعلق دارد. برای مثال توابع

زیر را داریم

$$>> \quad x$$

$$>> \quad isreal(x)$$

$$>> \quad isinteger(x)$$

$$>> \quad isfloat(x)$$

$$>> \quad isvector(x)$$

$$>> \quad ischar(x)$$

$$>> \quad islogical(x)$$

$$>> \quad isnumeric(x)$$

$$>> \quad isscalar(x)$$

$$>> \quad isstr(x)$$

همچنین چند تابع از این گونه متناظر با ماتریس‌ها داریم

iscolumn	Determines whether input is column vector
isempty	Determines whether array is empty
ismatrix	Determines whether input is matrix
isrow	Determines whether input is row vector
isscalar	Determines whether input is scalar
isvector	Determines whether input is vector

در این باره چند مثال را مشاهده خواهیم کرد.

مثال ۱)

Command Window

```
>> x = 3

x =

     3

>> a1=isinteger(x);
>> a2=isfloat(x);
>> a3=isvector(x);
>> a4=isscalar(x);
>> a5=isnumeric(x);
>> a6=isreal(x);
>> [a1 a2 a3 a4 a5 a6]

ans =

     0     1     1     1     1     1

fx >>
```

مثال ۲)

Command Window

```
>> x = 12.036

x =

   12.0360

>> a1=isinteger(x);
>> a2=isfloat(x);
>> a3=isvector(x);
>> a4=isscalar(x);
>> a5=isnumeric(x);
>> a6=isreal(x);
>> [a1 a2 a3 a4 a5 a6]

ans =

     0     1     1     1     1     1

fx >>
```

مثال۳)

Command Window

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

x =

     1     2     3     4

>> a1=isinteger(x);
>> a2=isfloat(x);
>> a3=isvector(x);
>> a4=isscalar(x);
>> a5=isnumeric(x);
>> a6=isreal(x);
>> [a1 a2 a3 a4 a5 a6]

ans =

     0     1     1     0     1     1

fx >>
```

مثال ۴)

Command Window

```
>> x = 'Hello'

x =

Hello

>> a1=isinteger(x);
>> a2=isfloat(x);
>> a3=isvector(x);
>> a4=isscalar(x);
>> a5=isnumeric(x);
>> a6=isreal(x);
>> [a1 a2 a3 a4 a5 a6]

ans =

     0     0     1     0     0     1

fx >>
```

مثال۵)

Command Window

```
>> x = 3+4i

x =

   3.0000 + 4.0000i

>> a1=isinteger(x);
>> a2=isfloat(x);
>> a3=isvector(x);
>> a4=isscalar(x);
>> a5=isnumeric(x);
>> a6=isreal(x);
>> [a1 a2 a3 a4 a5 a6]

ans =

     0     1     1     1     1     0

fx >>
```

متغیرهای از نوع منطقی را قبلا تعریف کردیم. این متغیر می‌تواند مقادیر صفر و یک را دریافت کند. این نوع متغیر را با استفاده از کلمه logical تعریف می‌کنیم. این متغیر می‌تواند به صورت بردار یا ماتریس نیز تعریف شود. از طرفی متغیرها از نوع‌های دیگر را می‌توان به متغیر منطقی تبدیل کرد. برای مثال

Command Window

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

a =

     1     2     3
     0     1    -3

>> a1=logical(a)

a1 =

     1     1     1
     0     1     1

>> a1(2,2)

ans =

     1

>> a1(2,2)=5

a1 =

     1     1     1
     0     1     1
```

به علاوه داریم

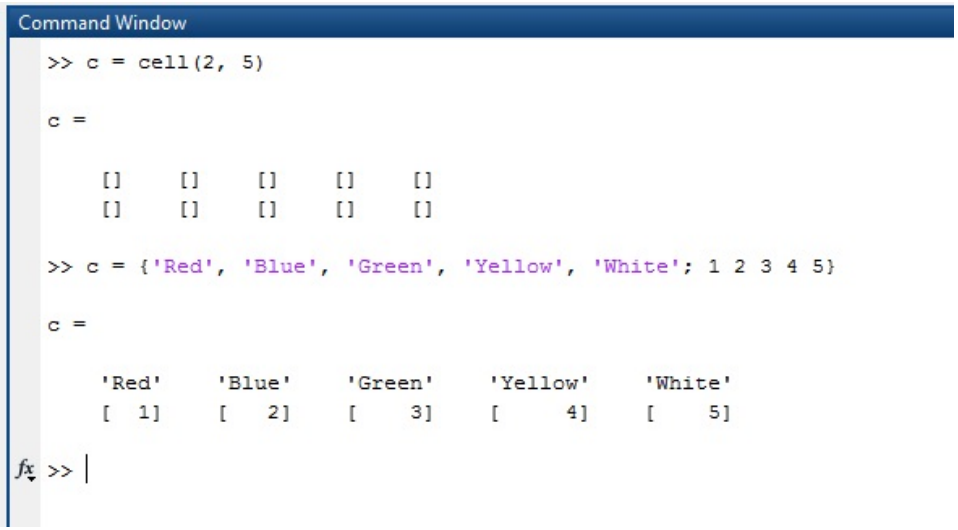

```
Command Window
>> b1=true

b1 =

     1

>> b2=true(3)

b2 =

     1     1     1
     1     1     1
     1     1     1

>> b3=true(3,2)

b3 =

     1     1
     1     1
     1     1

fx >>
```

و

Command Window

```
>> b1=false

b1 =

     0

>> b2=false(3)

b2 =

     0     0     0
     0     0     0
     0     0     0

>> b3=false(3,2)

b3 =

     0     0
     0     0
     0     0

fx >>
```

با استفاده از دستور cell می‌توانیم ارایه‌ای از سلول‌ها تعریف کنیم. برای مثال این دستور به صورت زیر استفاده می‌شود



و به صورت زیر می‌توان به اطلاعات ذخیره شده در ارایه دسترسی داشت

Command Window

```
c =

    'Red'    'Blue'    'Green'    'Yellow'    'White'
    [  1]    [   2]    [    3]    [     4]    [    5]

>> c(1,2)

ans =

    'Blue'

>> c(4)

ans =

    [2]

>> c(1:2,2:3)

ans =

    'Blue'    'Green'
    [   2]    [    3]
```

با استفاده از دستور num۲cell می‌توانیم یک ماتریس را به صورت ارایه‌ای تبدیل کنیم

Command Window

```
>> A=magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> B=num2cell(A)

B =

    [8]    [1]    [6]
    [3]    [5]    [7]
    [4]    [9]    [2]

>> B(1,2)+B(1,3)
Undefined operator '+' for input arguments of type 'cell'.

>> A(1,2)+A(1,3)

ans =

     7
```

انواع اعداد قابل تعریف در MATLAB به صورت زیر می‌باشند

Function	Purpose
Double	Converts to double precision number
Single	Converts to single precision number
int8	Converts to 8-bit signed integer
int16	Converts to 16-bit signed integer
int32	Converts to 32-bit signed integer
int64	Converts to 64-bit signed integer
uint8	Converts to 8-bit unsigned integer
uint16	Converts to 16-bit unsigned integer
uint32	Converts to 32-bit unsigned integer
uint64	Converts to 64-bit unsigned integer

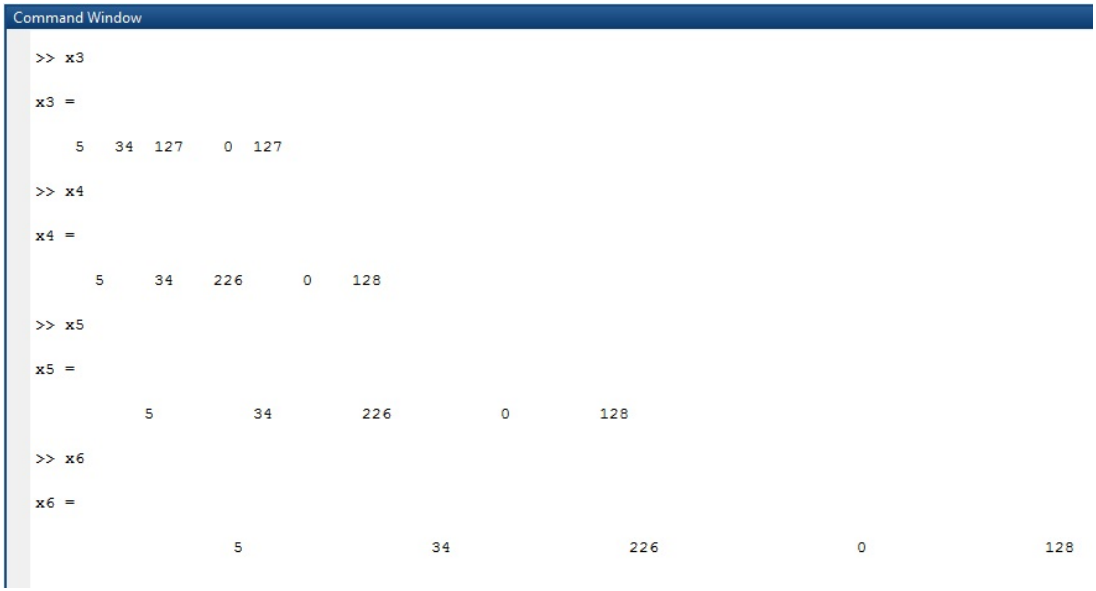
که حتی نحوه‌ی نمایش این اعداد متفاوت می‌باشد. برای مثال


```
Command Window
>> a=[5.32 34.47 226.28 0.0001 128.329]

a =

    5.3200   34.4700  226.2800    0.0001  128.3290

>> x1 = single(a);
>> x2 = double(a);
>> x3 = int8(a);
>> x4 = int16(a);
>> x5 = int32(a);
>> x6 = int64(a);
fx >>
```

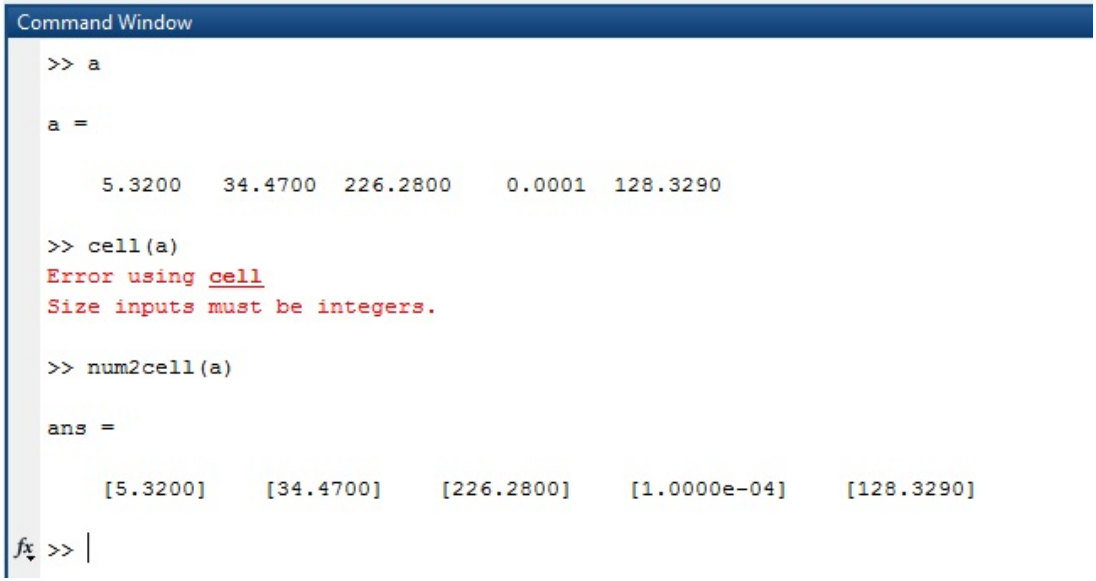
و

Command Window

```
>> a

a =

    5.3200   34.4700  226.2800    0.0001  128.3290

>> x1

x1 =

    5.3200   34.4700  226.2800    0.0001  128.3290

>> x2

x2 =

    5.3200   34.4700  226.2800    0.0001  128.3290
```



حال اگر بخواهیم این بردار را به صورت آرایه ای از سلول ها تبدیل کنیم می نویسیم



دقت کنید در پیام خطای بالا نوشته ورودی تابع cell باسد عدد طبیعی باشد و این امر به معنای زیر می باشد

Command Window

```
>> x0=[1 2 3]

x0 =

     1     2     3

>> cell(x0)

ans(:,:,1) =

    []    []


ans(:,:,2) =

    []    []


ans(:,:,3) =

    []    []

fx >>
```

۲ تعریف تابع با @ و توابع ترسیم متناظر

قبلا نحوه‌ی محاسبه یک تابع با استفاده از function را معرفی کردیم حال می‌خواهیم با استفاده از @ یک تابع تعریف کنیم. که به صورت زیر عمل می‌کنیم

$$f = @(arglist)\ expression$$

برای مثال

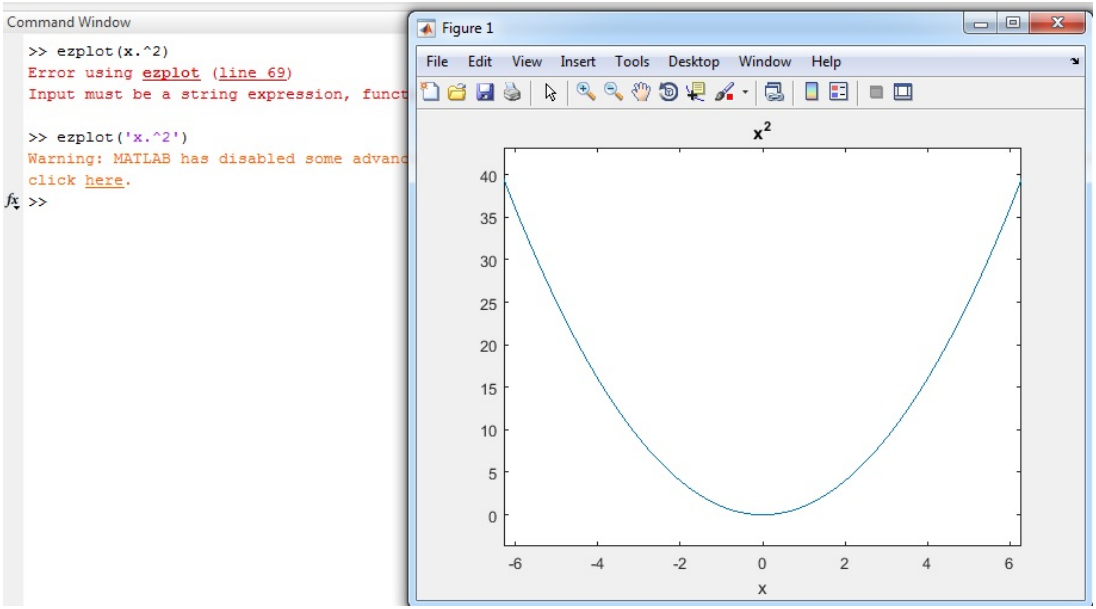
Command Window

```
>> power1 = @(x, n) x.^n;
>> result1 = power1(7, 3);
>> result2 = power1(49, 0.5);
>> result3 = power1(10, -10);
>> result4 = power1 (4.5, 1.5);
>>
>>
```

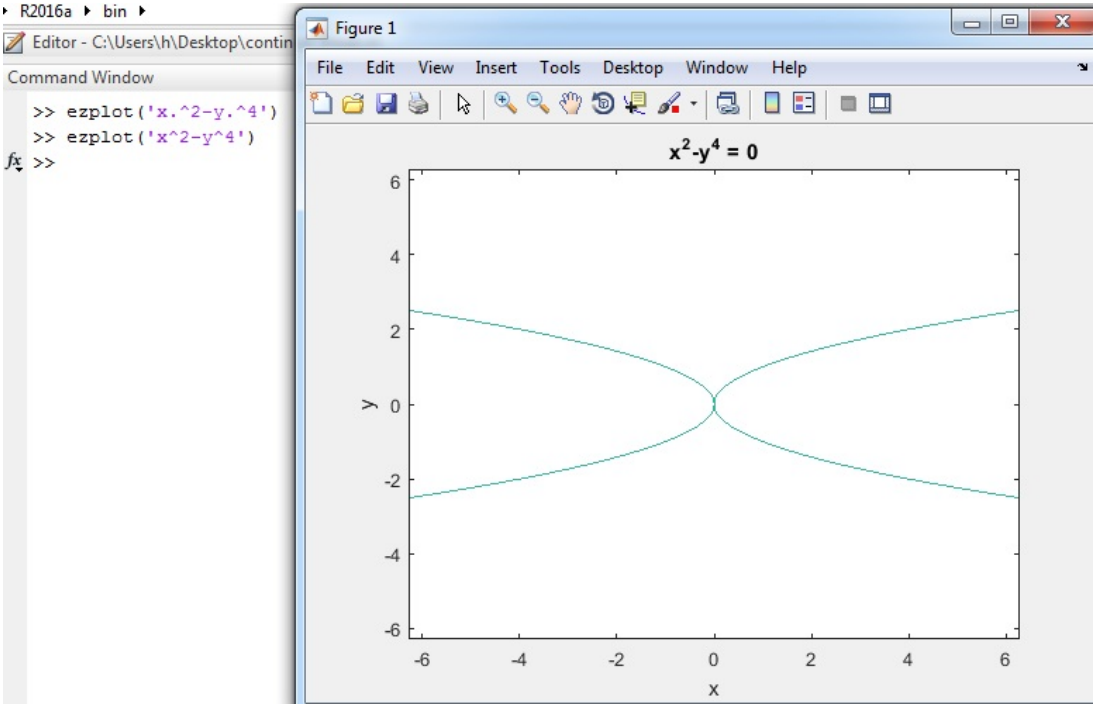
و


```
Command Window
>> result1

result1 =

   343

>> result2

result2 =

     7

>> result3

result3 =

   1.0000e-10

>> result4

result4 =

    9.5459
```

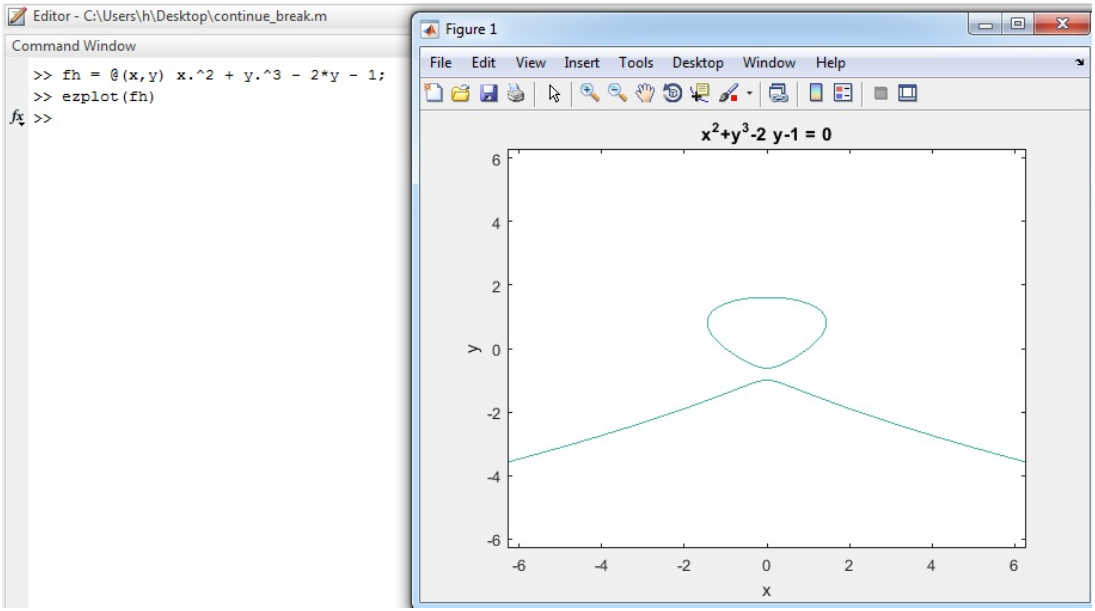
تابع ezplot برای رسم توابع بصورت ساده استفاده می‌شود برای مثال



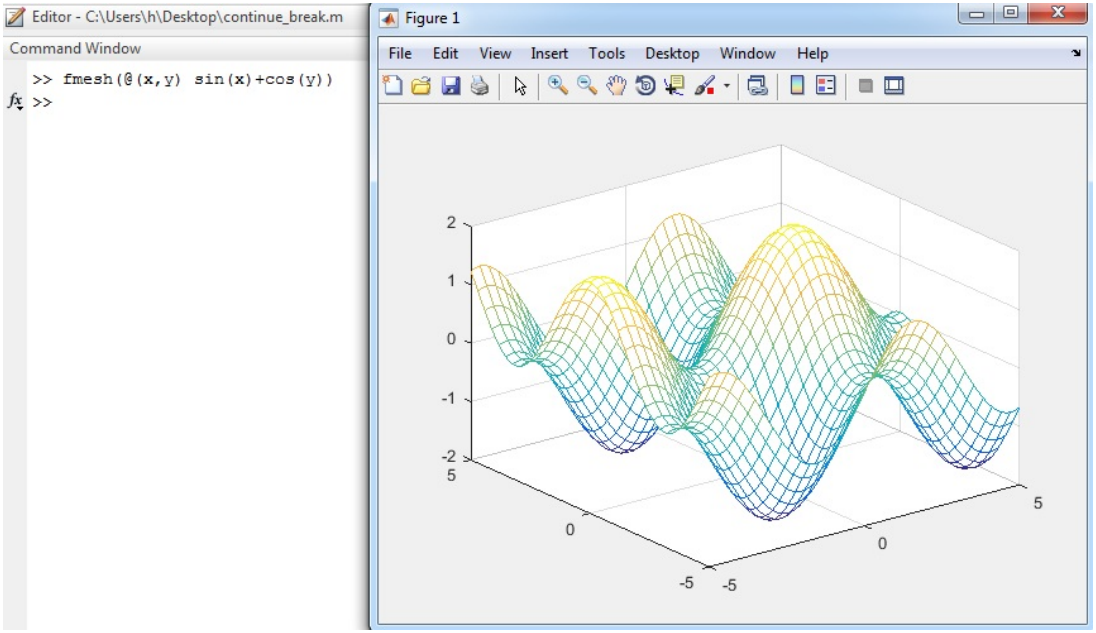
و



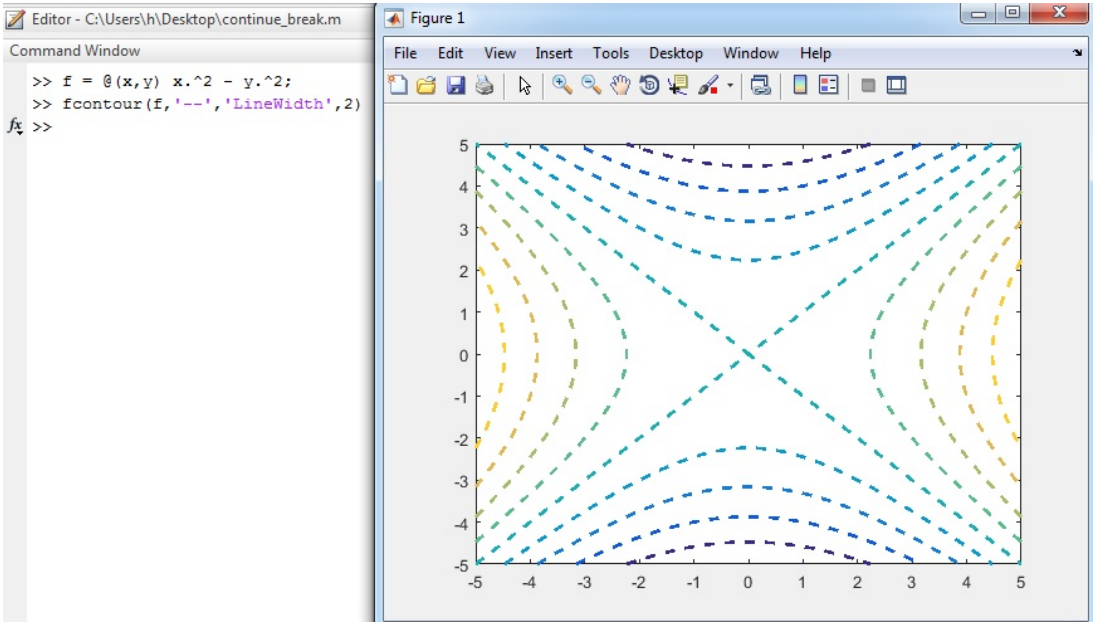
و داریم



برای رسم های سه بعدی ساده داریم



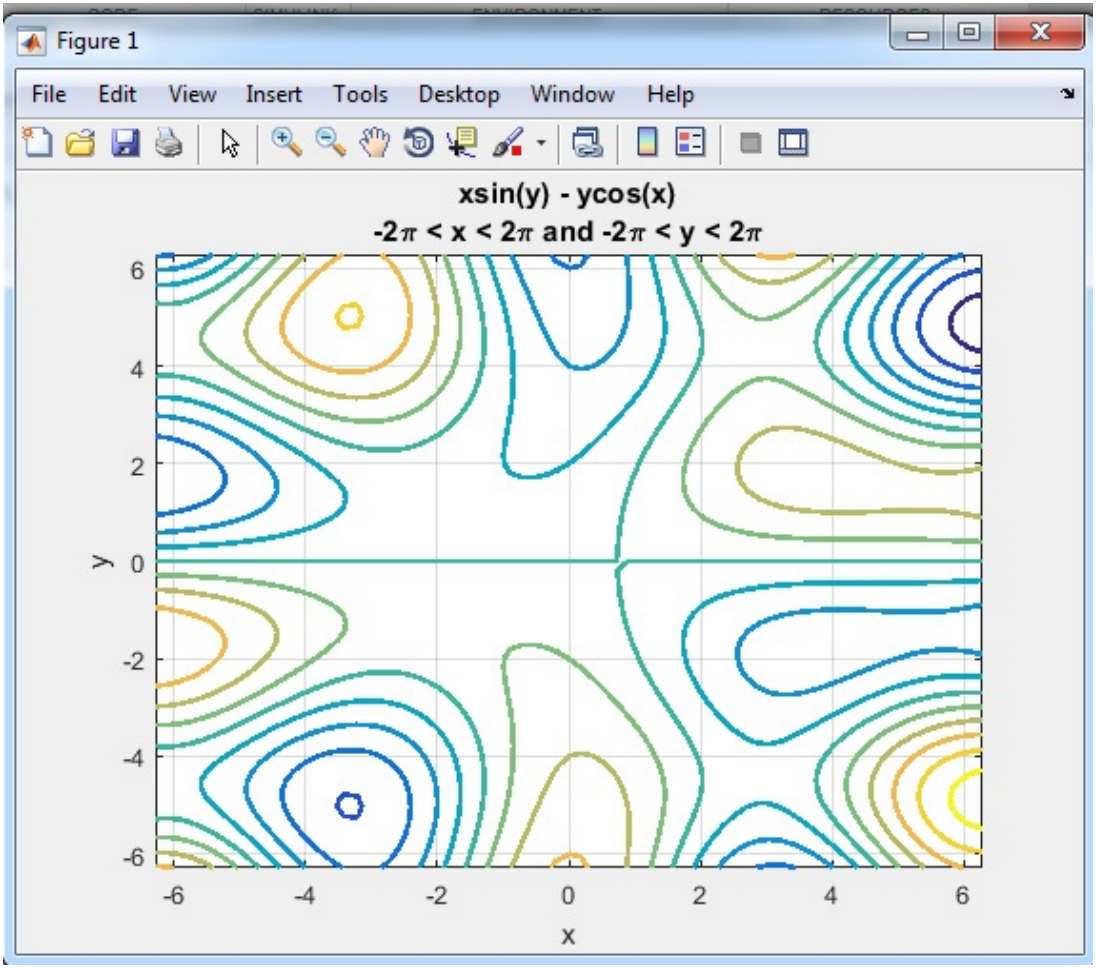
و برای رسم کانتورها به صورت ساده داریم



برای مثال

```
Command Window
>> fcontour(@(x,y) x.*sin(y) - y.*cos(x), [-2*pi 2*pi], 'LineWidth', 2);
>> grid on
>> title({'xsin(y) - ycos(x)','-2\pi < x < 2\pi and -2\pi < y < 2\pi'})
>> xlabel('x')
>> ylabel('y')
>>
```

که تصویر به صورت زیر می باشد



مثال) تابع یر را رسم کنید.

$$x = r\cos(s)\sin(t)$$

$$y = r\sin(s)\sin(t)$$

$$z = r\cos(t)$$

$$r = ۲ + sin(۷s + ۵t)$$

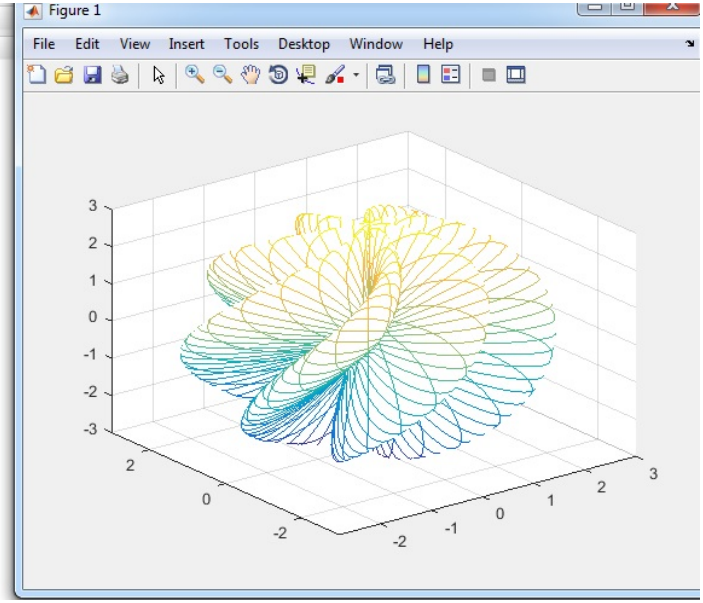
$$۰ < s < ۲\pi,\ ۰ < t < \pi$$

می نویسیم

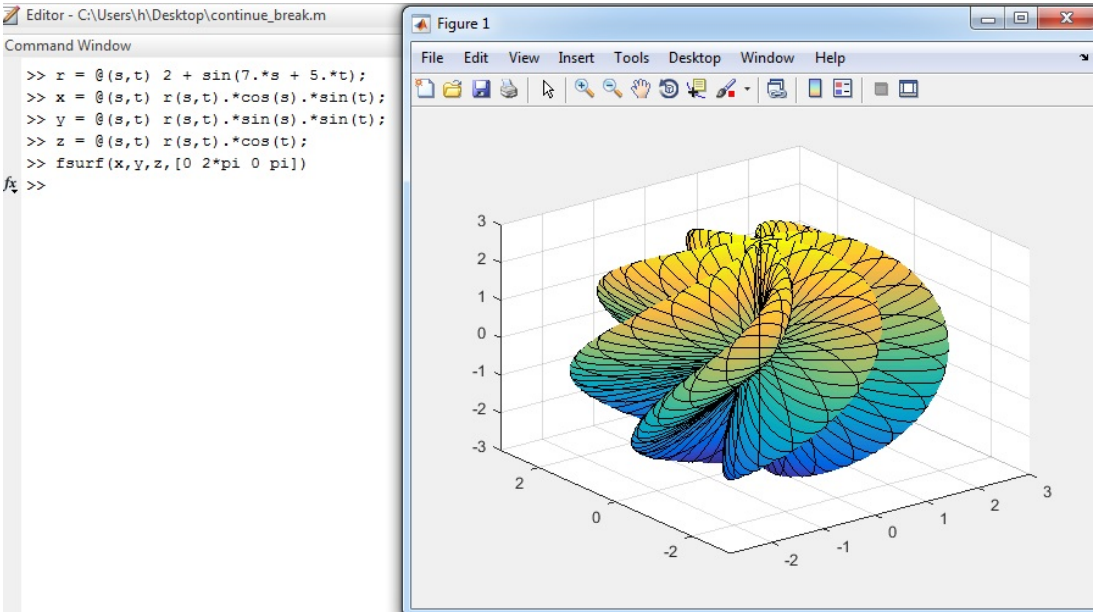
Editor - C:\Users\h\Desktop\continue_break.m

Command Window

```
>> r = @(s,t) 2 + sin(7.*s + 5.*t);
>> x = @(s,t) r(s,t).*cos(s).*sin(t);
>> y = @(s,t) r(s,t).*sin(s).*sin(t);
>> z = @(s,t) r(s,t).*cos(t);
>> fmesh(x,y,z,[0 2*pi 0 pi])
>>
```



و

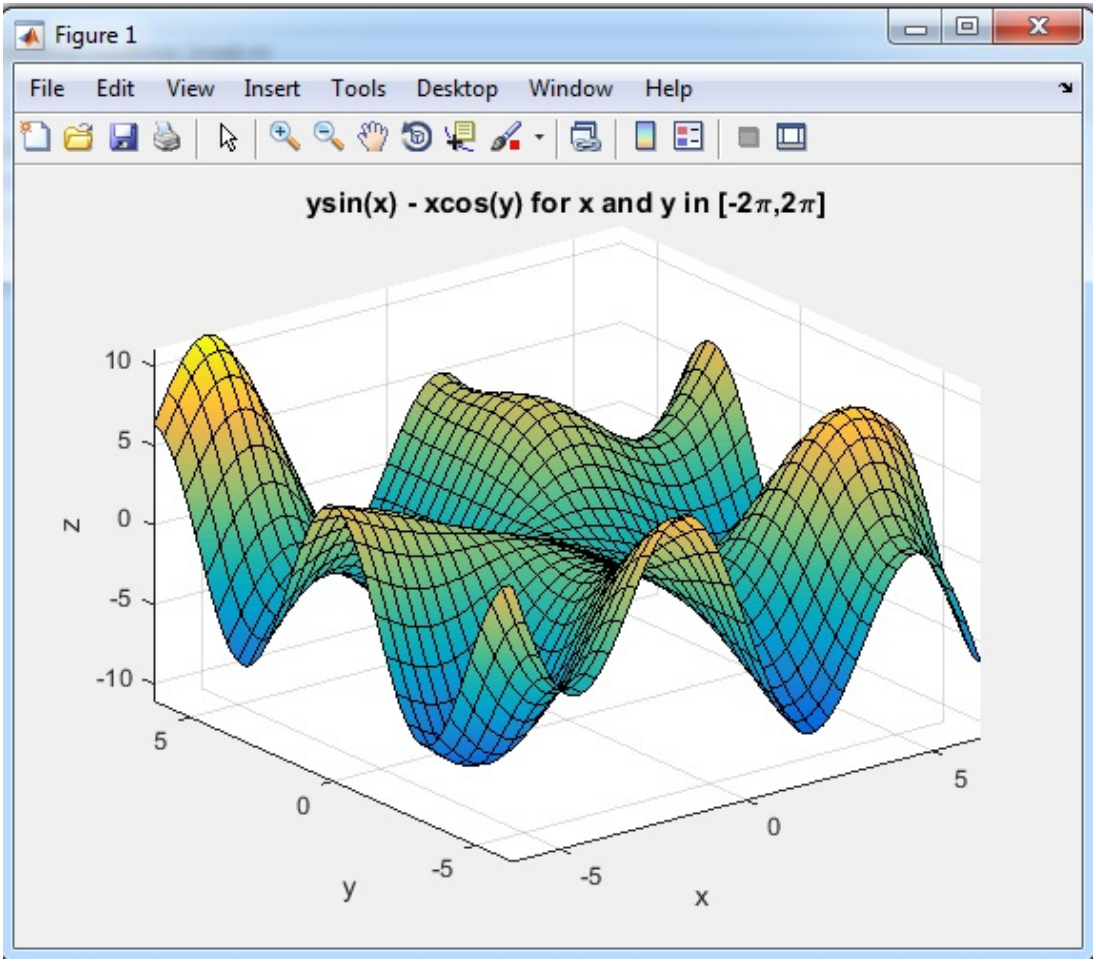


مثال)اگر دستورات زیر را بنویسیم

Command Window

```
>> fsurf(@(x,y) y.*sin(x)-x.*cos(y),[-2*pi 2*pi])
>> title('ysin(x) - xcos(y) for x and y in [-2\pi,2\pi]')
>> xlabel('x');
>> ylabel('y');
>> zlabel('z');
>>
```

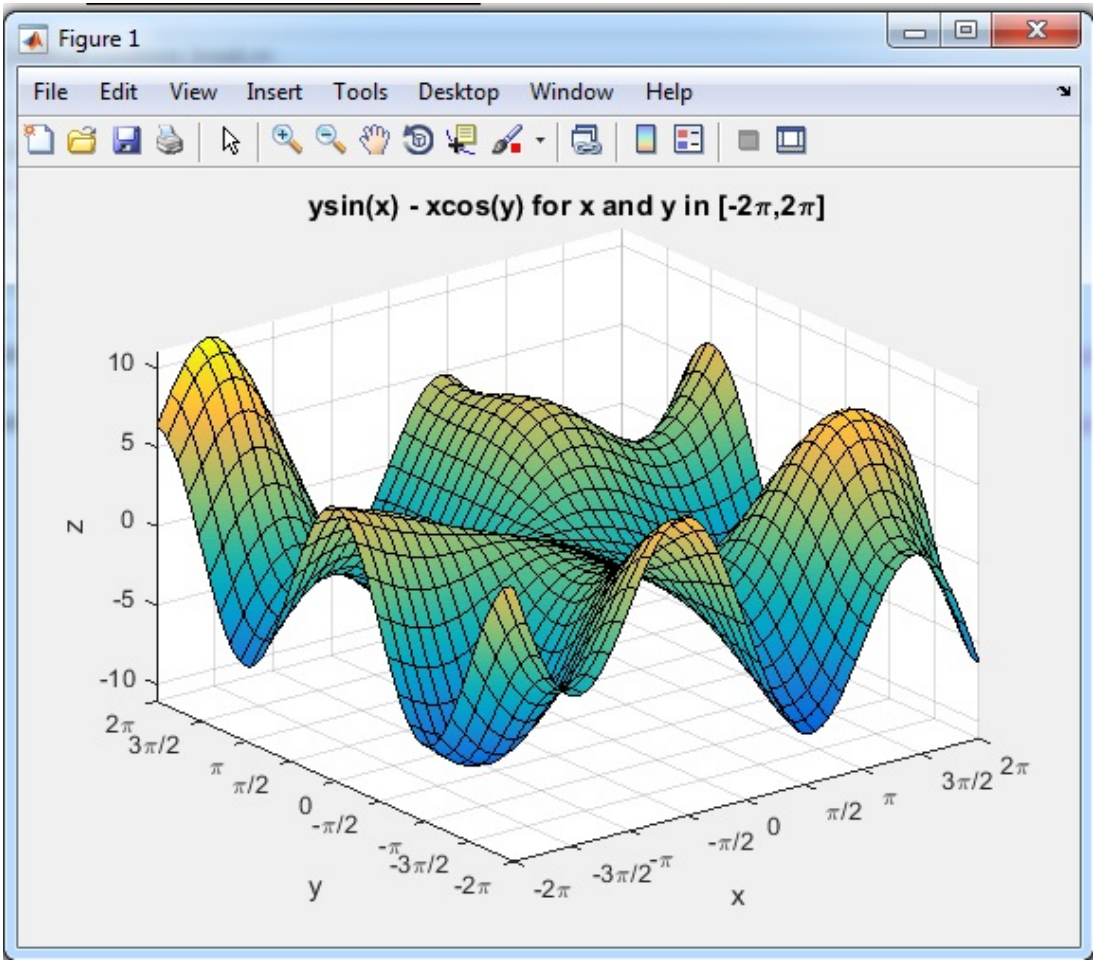
شکل به صورت زیر رسم می شود



حال اگر دستورات زیر را اضافه کنیم

```
Command Window
>> fsurf(@(x,y) y.*sin(x)-x.*cos(y),[-2*pi 2*pi])
>> title('ysin(x) - xcos(y) for x and y in [-2\pi,2\pi]')
>> xlabel('x');
>> ylabel('y');
>> zlabel('z');
>> ax.XTick = -2*pi:pi/2:2*pi;
>> ax.XTickLabel = {'-2\pi','-3\pi/2','-\pi','-\pi/2','0','\pi/2','\pi','3\pi/2','2\pi'};
>> ax.YTick = -2*pi:pi/2:2*pi;
>> ax.YTickLabel = {'-2\pi','-3\pi/2','-\pi','-\pi/2','0','\pi/2','\pi','3\pi/2','2\pi'};
fx >>
```

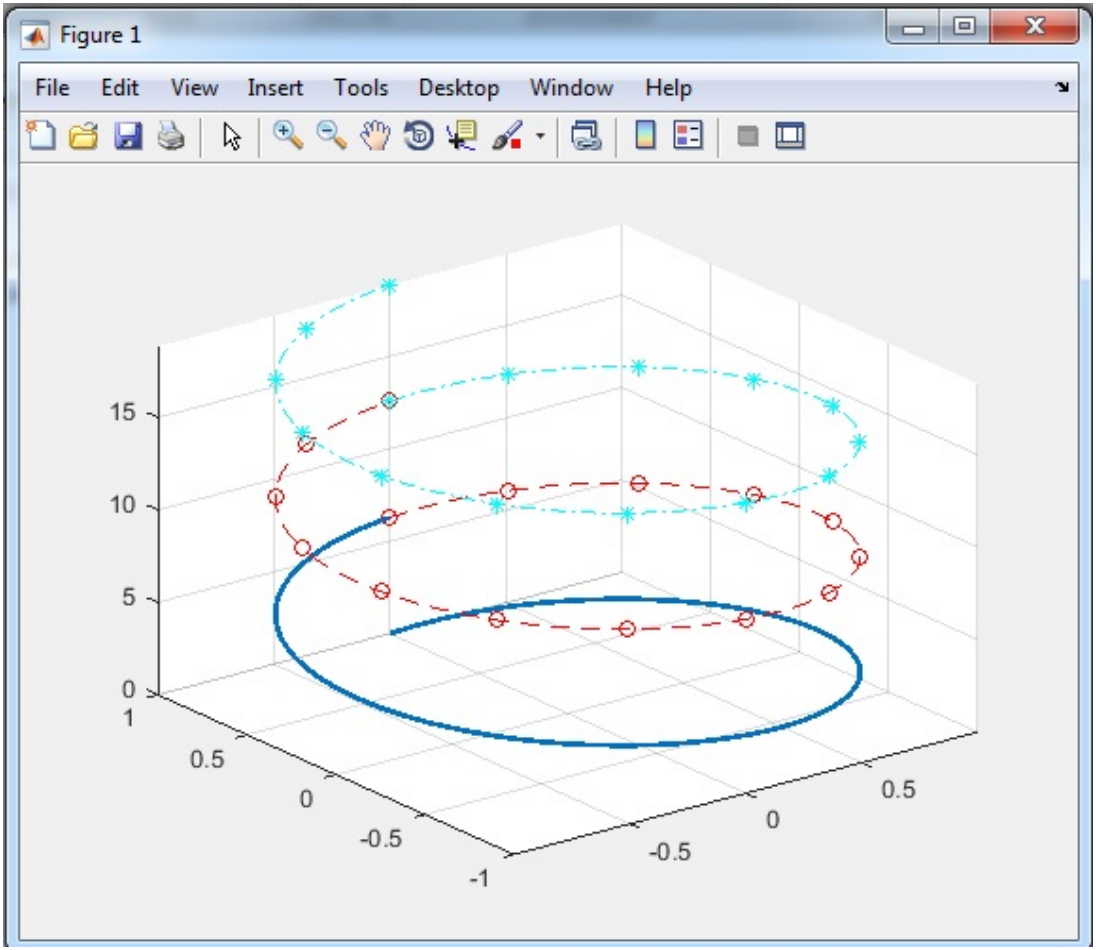
داریم



مثال) اگر دستورات زیر را وارد کنیم

```
Command Window
>> fplot3(@(t)sin(t), @(t)cos(t), @(t)t, [0 2*pi], 'LineWidth', 2)
hold on
fplot3(@(t)sin(t), @(t)cos(t), @(t)t, [2*pi 4*pi], '--or')
fplot3(@(t)sin(t), @(t)cos(t), @(t)t, [4*pi 6*pi], '-.*c')
hold off
>>
```

داریم



مثال) برای fplot داریم