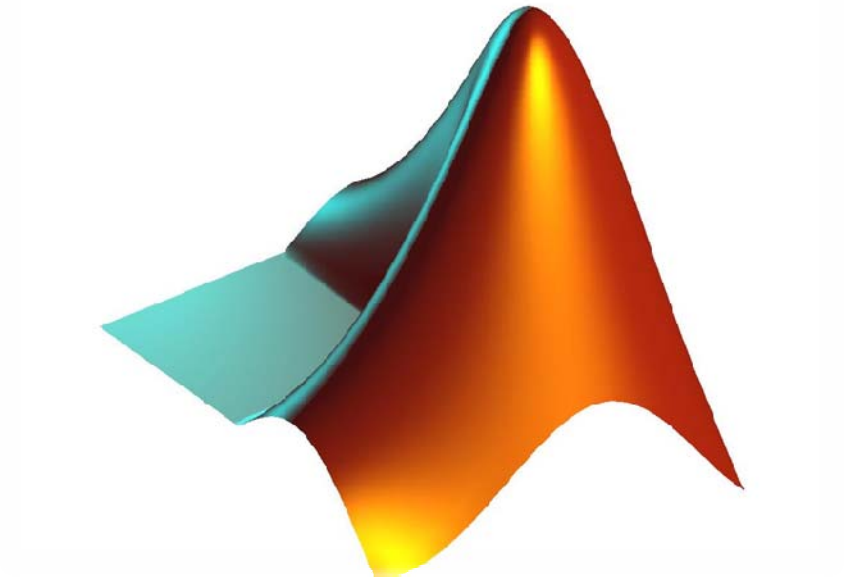
مقدمه ای کوتاه بر نحوه استفاده از نرم افزار MATLAB برای دانشجویان دوره کارشناسی علوم و مهندسی

شروع کار با MATLAB



اسماعیل خالقی

عضو هیأت علمی گروه مکانیک

مقدمه

نرم افزار Matlab در بین نرم افزارهای محاسباتی از ویژگیهای بیشتری برخوردار است و با قابلیت های برنامه نویسی که دارد توانسته است به عنوان یک نرم افزار و زبان برنامه نویسی برای انجام محاسبات علوم و مهندسی مورد استفاده قرار گیرد .

در این خلاصه سعی شده است تا برای کسانی که به تازگی با Matlab آشنا شده اند روش کار توضیح داده شود و بتوانند برخی از مسائل محاسباتی که در سطح کارشناسی آموزش می بینند را توسط این نرم افزار پیاده سازی کنند.

تعریف شناسه ها

همانند بسیاری از زبان های برنامه نویسی نحوه تعریف شناسه ها در Matlab از اهمیت برخوردار است . منظور از شناسه نام یک متغییر ، نام یک آرایه ، نام تابع ، نام یک M-file و ... می باشد . در نامگذاری بایستی موارد زیر مورد لحاظ قرار گیرد .

۱- نام یک شناسه بایستی با یک حرف شروع شود (حرف اول یک نام عدد یا کاراکتر علامت نباشد)
۲- بین کاراکترهای یک نام فضای خالی (Space) نباشد .
۳- برای نامگذاری از اعداد و کاراکتر زیر خط (_) می توان استفاده کرد .
۴- طول نام یک شناسه بایستی محدود باشد . (۶۴ کاراکتر)

مقدار دهی متغیر ها

متغیر آدرسی از حافظه است که در آن داده ذخیره می شود و با نامگذاری این آدرس حافظه طبق قاعده بالا می توان به مقادیر ذخیره شده در این محل از حافظه دسترسی پیدا کرد .

در Matlab برای مقدار دهی یک متغیر از علامت (=) استفاده می شود و برای آزاد سازی متغیر از دستور clear می توان استفاده کرد .

مثال : متغیری به نام gu را مقدار ۵ می دهیم .

```
>> gu=5
>> gu+3
    8
>> clear gu
```

دستور اخیر مقدار وابسته شده به gu را حذف می کند . توجه نمایید که اگر از دستور clear all استفاده کنید ، تمامی متغیرهای مقداردهی شده آزاد می شود .

وقتی برنامه یک کار محاسباتی را در دست دارید بایستی بدانید که چه متغیرهایی را تعریف کرده و مقداردهی نموده اید . با دستور who می توانید لیست متغیرهای مقداردهی شده را مشاهده کنید .

مثال : متغیرهای a,b,c,d را تعریف و مقدار دهی کرده ایم . لذا هر موقع دستور who را اجرا کنید متغیرهای مقداردهی شده را نمایش خواهد داد .

```
>> a=3;
>> b=5;
>> d=0.0003;
>> c=20000000101222;
>> who

Your variables are:
a  b  c  d
```

اگر به جای دستور who دستور whos را استفاده کنید می توانید مشخصات بیشتری از متغیرهای مقدار دهی شده را مشاهده کنید .

```
>>whos
  Name      Size                    Bytes  Class

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

Grand total is 4 elements using 32 bytes
```

توجه : دقت نمایید که همانگونه که در نتیجه دستور whos نیز مشاهده می کنید . برای تمامی داده ها بدون توجه به اینکه مقدار آن یک عدد کوچک مثل d یا یک عدد معمولی مثل a,b یا یک عدد بزرگ مثل c باشد همگی ۸ بایت فضا از حافظه اشغال می کند . بنابراین طول کلمه حافظه تعریف شده در matlab ، ۸ بایت می باشد و بزرگترین یا کوچکترین عددی که در Matlab قابل تعریف است ، مقداری است که در این فضا جا شود .

تعریف آرایه ها

در Matlab آرایه ها یکی از عناصر مهم و پرکاربرد است و می توان یک آرایه را به صورتهای مختلفی تعریف نمود

۱) می توان عناصر آرایه را داخل یک کروشه قرار داد و با ((,)) یا یک فاصله خالی ((Space)) جدا نمود .

```
>> A=[2,3,4,5,0,1]
```

برای دسترسی به عناصر آرایه کافیست نام آرایه را نوشته و در مقابل آن در داخل پرانتز اندیس عضو را بنویسیم . به طور مثال اگر نام آرایه test باشد test(i) یعنی i امین عضو آرایه .

۲) برای تعریف آرایه های منظم روشهای مختلفی وجود دارد .

با دستور

```
A=   عضو ابتدا  :  عضو انتها
A= a:b
```

می توان آرایه منظمی تعریف کرد که از a شروع می شود و یک واحد یک واحد افزایش می یابد تا به b برسد .

```
>> c=3:8
c =
     3     4     5     6     7     8
```

در این نوع آرایه منظم می توان به جای گام یک واحدی از گام n واحدی استفاده نمود .

```
A=   عضو ابتدا  :  گام  :  عضو انتها
A= a:n:b
```

```
>> b=2:3:11
b =
     2     5     8    11
```

توضیح : اگر با گام مورد نظر آخرین عنصر آرایه از عضو انتها بیشتر شود ، مقدار اضافی تر از عضو انتها در نظر گرفته نمی شود .

همچنین گام می تواند عدد منفی یا عدد اعشاری باشد .

نوع دیگر از آرایه منظم با دستور linspace تولید می شود . که به شکل زیر نوشته می شود .

```
A=linspace(  عضو ابتدا  ,  عضو انتها , تعداد عناصر  )
A= linspace(a,b,n)
```

با این دستور می توان آرایه ای تولید کرد که عضو ابتدای آن a و عضو انتهای آن b باشد و آرایه شامل n عضو با تقسیمات مساوی باشد .

```
>> b=linspace(1,2,5)
b =
    1.0000    1.2500    1.5000    1.7500    2.0000
```

آرایه های دوبعدی

آرایه های دوبعدی را می توان به شکلهای مختلف تعریف کرد .

۱) می توان عناصر آرایه را در داخل کروشه نوشته و با کاما یا فاصله خالی جدا کرد و هر سطر را نیز با سمی کالن ((;)) یا enter جدا نمود .

مثال :

```
>> A=[2,3,4;6,0,9;1,3,8]
A =
     2     3     4
     6     0     9
     1     3     8
```

یا می توان سطرهای یک آرایه دوبعدی را به صورت جداگانه به شکل یک آرایه تعریف کرد و بعد نام آنها را داخل کروشه نوشته و با سمی کالن جدا کنید .

مثال

```
>> A=[3,4,6]
>> B=[0,9,8]
>> C=[-1,2,3]
>> T=[A;B;C]
>> z=[A;B;C;T]
z =
     3     4     6
     0     9     8
    -1     2     3
     3     4     6
     0     9     8
    -1     2     3
```

اعمال محاسباتی بر آرایه ها

– برای ضرب یا جمع عناصر یک آرایه و یک اسکالر از اعمال معمولی (- / * +) استفاده می شود .

مثال:

```
>> a=[2,3,6];
>> a+2
ans =
     4     5     8

>> 4-a
ans =
     2     1    -2

>> a*3
ans =
     6     9    18

>> a/3
ans =
    0.6667    1.0000    2.0000
```

توجه : اگر بخواهیم عناصر آرایه را تک به تک بر یک اسکالر تقسیم کنیم از عملگر معمولی / استفاده می شود ولی اگر بخواهم اسکالری بر تک تک عناصر آرایه تقسیم گردد از عملکر **./** استفاده می شود .

```
>> 3./a
ans =
    1.5000    1.0000    0.5000
```

– اگر بخواهیم عناصر آرایه را به توان یک اسکالر برسانیم و یا یک اسکالر به توان عناصر یک آرایه برسد از عملگر .^ استفاده می شود .

```
>> a.^2
ans =
     4     9    36
>> 3.^a
ans =
     9    27   729
```

- اگر دو آرایه هم اندازه داشته باشیم و بخواهیم عناصر نظیر به نظیر با هم جمع یا تفریق شود از عملگر معمولی + و – استفاده می شود .
- ولی اگر بخواهیم عناصر نظیر به نظیر در هم ضرب یا بر هم تقسیم شود و یا به توان یکدیگر برسد از .* یا ./ یا .^ استفاده می کنیم .

برخی از آرایه های خاص

دستوراتی وجود دارد که می توان برخی آرایه های خاص ایجاد نمود .

با دستور rand(n,m) می توان یک آرایه تصادفی با n سطر و m ستون ایجاد نمود که عناصر آن بین صفر و یک است و با توزیع یکنواخت محاسبه و تولید می شود .

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

با دستور zeros(n,m) می توان یک آرایه با n سطر و m ستون ایجاد نمود که عناصر آن همگی صفر باشد .

```
>> zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

با دستور diag(a) می توان ماتریسی قطری تولید کرد که آرایه a روی قطر اصلی آن باشد .

```
>> c=1:5;
>> diag(c)
ans =
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
      0     0     0     0     5
```

دسترسی به عناصر یک آرایه دوبعدی

فرض کنید آرایه دوبعدی A با n سطر و m ستون مقدار دهی شده باشد .

- دسترسی به عنصر i و j با دستور A(i,j)
- دسترسی به سطر i تا k و ستون j تا h با دستور A(i:k,j:h)

```
A =
     2     3     4     5     1
     5     0     9    11     1
     2     3     1     0     9
    10     7     8     1     2

>> A(1:3,2:4)
ans =
     3     4     5
     0     9    11
     3     1     0
```

تفاوت محاسبات کامپیوتری با محاسبات ریاضی

در کامپیوتر برخی از اعمال محاسباتی با آنچه در ریاضی داریم متفاوت است .

به طور مثال در ریاضی عدد صفر تنها عدد خنثی جمعی است یعنی فقط صفر است که با هر عدد جمع شود حاصل خود آن عدد می شود و اعداد دیگری این خاصیت را ندارد M+0 =M

اما در محاسبات کامپیوتری ممکن است اعداد دیگری غیر از صفر باشند که خاصیت خنثی جمعی داشته باشد و این به دلیل خطای گرد کردن و خطای تبدیل مبنا بر اثر محدودیت طول کلمه حافظه می باشد .

در MAtlab با دستور eps می توان بزرگترین عددی را که خاصیت خنثی جمعی دارد را نمایش داد .

```
>> eps
ans =
  2.2204e-016
```

این عدد غیر صفر ولی کوچک را با هر عددی جمع کنید خود آن عدد خواهد شد .

```
>> eps+25
ans =
    25
```

مسلماً اعداد کمتر از eps نیز همین خاصیت را دارد .

در ریاضی صفر تنها عددی است که بی اثر جمعی است و این عدد تهی ساز نیز هست یعنی در هر عددی ضرب شود حاصل صفر خواهد شد ولی دقت شود که eps بی اثر جمعی است ولی تهی ساز نیست .

```
>> eps*15
ans =
  3.3307e-015
```

با دستور realmin و realmax می توان بزرگترین و کوچکترین عدد مثبت که توسط Matlab قابل تعریف است را مشاهده یا استفاده نمود.

```
>> realmin
ans =
  2.2251e-308

>> realmax
ans =
  1.7977e+308
```

لذا باید توجه داشت که در محاسبات realmin را بایستی صفر درنظر گرفته و realmax را بی نهایت محسوب نماییم .

نکته دیگری که بایستی به آن توجه داشت این است که ترتیب اعمال به ویژه در مورد اعداد مثبت و منفی و یا محاسبات اعداد گنگ ، ممکن است نتایج با آنچه در ریاضی با آن مواجه هستیم متفاوت باشد .

مثال : سه عدد a=0.08 , b=-0.5 , c=0.42 را در نظر بگیرید . از نظر ریاضی این سه عدد را با هر ترتیبی با هم جمع کنیم حاصل صفر خواهد شد . ببینیم در محاسبات کامپیوتری نیز وضع همینگونه است ؟

```
>> a=0.08;
>> b=-0.5;
>> c=0.42;

>> a+b+c
      ans =0


>> a+c+b
      ans =0

>> b+c+a
      ans =-1.3878e-017
```

در مورد اعداد گنگ نیز نتیجه محاسبات با خطا مواجه خواهد شد ، چون اعداد گنگ دارای بی نهایت اعشار است و به دلیل کمبود حافظه بخشی از اعشار عدد گنگ در محاسبات حذف می شود .

مثال : فرض کنید a ریشه دوم ۲ باشد . به خطای ایجاد شده نگاه کنید .

```
>> a=sqrt(2);
>> a^2-2
ans =
  4.4409e-016
```

یا به طور مثال $\sin(\pi)$ را ملاحظه کنید .

```
>> sin(pi)
ans =
  1.2246e-016
```

رسم نمودارها

برای رسم یک نمودار دو بعدی دو حالت ممکن است داشته باشیم

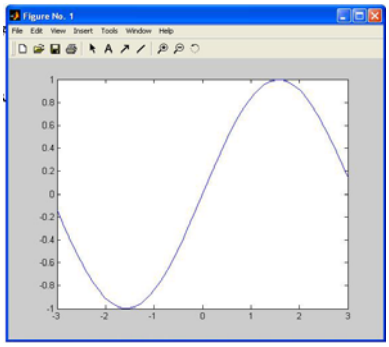
الف) ممکن است نقاط (x,y) زیادی در دسترس باشد و بخواهیم برای این نقاط نموداری را رسم کنیم .
ب) ممکن است تابع y=f(x) مشخص باشد و بخواهیم این تابع را در بازه [a,b] رسم کنیم .

در هر دو حالت نقاط x ها را در یک آرایه و y ها را در آرایه ای دیگر ذخیره می نماییم و با دستور **plot** نمودار مورد نظر رسم می شود .

مثال : y=sin(x) را در بازه [-3,3] رسم کنید .
برای اینکار بازه [-3,3] را به تعدادی نقاط تقسیم می کنیم و در یک آرایه نگهداری می نماییم . به ازای نقاط بدست آمده مقادیر y متناظر را بدست آورده و با دستور plot رسم می کنیم .

```
>> x=linspace(-3,3,50);
>> y=sin(x);
>> plot(x,y)
```

وقتی نموداری رسم می شود این نمودار در یک پنجره به نام **figure** رسم می گردد و هر گاه دستور رسم جدیدی صادر شود محتویات این پنجره پاک شده و نمودار جدید جایگزین می گردد .



با دستور **hold on** می توان محتویات این پنجره را حفظ نموده و نمودار جدید را روی نمودار قبلی رسم کرد .

اگر ضابطه تابع مشخص باشد می توان بدون تقسیم بندی محور x ها با دستور **ezplot** تابع مورد نظر را رسم کرد . برای اینکار بایستی عبارت f را داخل کوتیشن قرار داده و به صورت زیر اجرا کرد . **ezplot('f ')**

```
>> ezplot('sin(x)')
```

اگر بخواهیم نمودار در بازه مشخصی رسم شود می نویسیم **ezplot(g,[a,b])**
چند مثال زیر را اجرا کنید .

```
>> g='sin(1/x)'
>> ezplot(g,[-1,1])
----------------------------
>> k='t^3-3*t+1'
>> ezplot(k,[-3,3])
----------------------------
>>ezplot('a^2-3*exp(a)')
```

لازم به ذکر است که اگر بازه برای رسم نمودار مشخص نشود matlab به صورت پیش فرض در بازه [-2π,2π] عبارت مورد نظر را رسم می کند.

وقتی از دستور plot استفاده می کنید می توان نمودار منحنی ها و توابع را با علامت جدول زیر به صورت های مختلف رسم نمود .

کاراکتر	رنگ
B	آبی
G	سبز
R	قرمز
C	فیروزه ای
M	ارغوانی
Y	زرد
K	سیاه
W	سفید

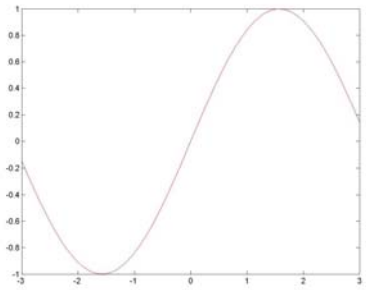
برای اعمال هر یک از رنگ ها یا علایم مختلف در رسم نمودارها کافیست کاراکترهای معرفی شده در جدول بالا را در دستورات plot داخل ' ' قرار دهیم اگر از چند کاراکتر بخواهیم استفاده کنیم می توان کاراکترها را داخل ' ' و با یک فاصله قرار داد

کاراکتر	علامت
O	دایره ای
+	به شکل +
*	به شکل *
S	به صورت مربع
D	به صورت لوزی
^	به صورت مثلثی
>	به صورت مثلث رو به چپ
<	به صورت مثلث روبه راست
P	ستاره پنج پر
H	ستاره شش پر
-	به صورت خط توپر
:	نقطه چین
-.	خط نقطه
--	خط چین

مثال :

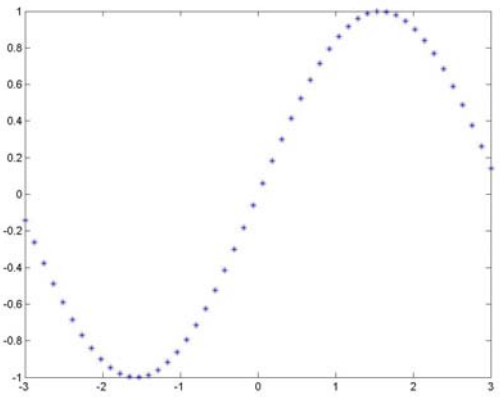
رسم نمودار sin(x) به صورت قرمز رنگ

```
>> x=linspace(-3,3,50);
>> y=sin(x);
>> plot(x,y,'r')
```



یا رسم همین نمودار به صورت ستاره وار آبی رنگ

```
>> plot(x,y,'* b')
```



تنظیمات محور مختصات

وقتی نموداری رسم شده باشد و پنجره figure باز باشد می توان با دستور axis محورهای مختصات را محدود کرد . برای اینکار می نویسیم

```
axis([a,b,c,d])
```

که در این صورت در پنجره figure نمودار رسم شده در محور x ها در بازه [a,b] و در محور y ها در بازه [c,d] محدود می شود .

تقسیم بندی پنجره figure

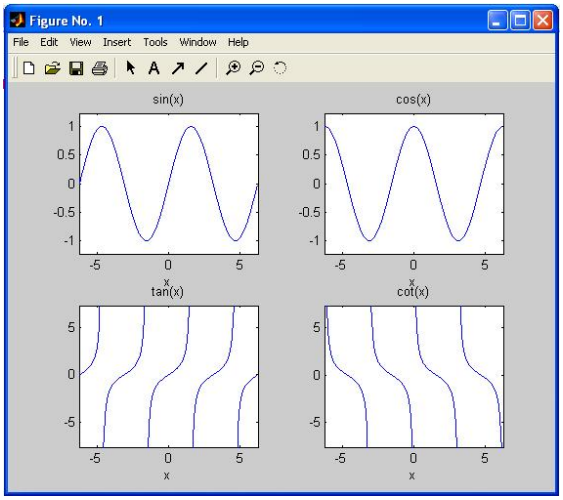
قبلا دیدیم که وقتی نموداری رسم می شود سطح پنجره figure از نمودار قبلی پاک شده و نمودار جدید جایگزین می شود . مگر اینکه دستور hold on را اجرا کنیم .

با دستور subplot می توان پنجره figure را به صورت دلخواه تقسیم بندی کرد و در هر قسمت نمودار مورد نظر را رسم نمود .

با استفاده از این دستور اگر بنویسیم **subplot(n , m , k)** پنجره figure به n سطر و m ستون تقسیم می شود و ناحیه k ام فعال است و اگر دستور رسم نمودار اجرا شود در این ناحیه فعال رسم خواهد شد .

مثال : می خواهیم توابع sin(x) , cos(x) , tan(x) , cot(x) در پنجره figure در دو سطر و دو ستون نمایش داده شود.

```
>> subplot(2,2,1)
>> ezplot('sin(x)')
>> subplot(2,2,2)
>> ezplot('cos(x)')
>> subplot(2,2,3)
>> ezplot('tan(x)')
>> ezplot('cot(x)')
```



در هر مرحله که دستور subplot(2,2,k) اجرا می شود در پنجره figure که به دو سطر و دو ستون تقسیم بندی شده است قسمت kام فعال می شود و اگر دستور رسم نمودار اجرا شود در این ناحیه رسم خواهد شد . شایان ذکر است که شماره بندی نواحی از سمت چپ از بالا به سمت راست و پایین به ترتیب خواهد بود.

تذکر : اگر به خواهیم پنجره figure را به حالت اولیه و نرمال برگردانیم کافیست دستور subplot(1,1,1) را اجرا نمایید .

برخی مواقع نیاز است تا در پنجره جدید و متمایزی نموداری را رسم کنیم بدون آنکه پنجره قبلی بسته شود .

برای اینکار از منوی file و در فرمان new گزینه figure را کلیک نمایید . پنجره figure جدیدی باز می شود که در نوار تیتر آن figure NO. k نوشته شده است که k شماره پنجره figure است .

بنابراین اگر در هر بخش از برنامه دستور figure(n) را اجرا کنید پنجره شماره n فعال خواهد بود .

برخی دستورات دیگر در رسم نمودار

با دستور bar(x,y) می توان نقاط آرایه های x,y را به صورت نوارهای مستطیل شکل نمایش داد و با دستور bar3(x,y) می توانید به صورت نمودار ستونی نمایش دهید .

با دستور pie(a) و pie3(a) می توانید عناصر یک آرایه را به صورت دیاگرام دایره ای دوبعدی یا سه بعدی نمایش دهید .

با دستور stairs(x,y) می توان نقاط آرایه های x و y را به صورت نموداری پلکانی نمایش داده .

از این دستورات برای رسم داده های آماری می توان استفاده کرد .

با دستور ezpolar و polar می توان نمودارها را در مختصات قطبی رسم نمود .

اگر بخواهیم در بالای یک نمودار عنوان دلخواهی را درج نماییم می توان با دستور title اینکار را انجام داد .

```
title(' عبارت مورد نظر ' )
```

البته بایستی دقت داشت که دستور فوق بعد از اجرای دستور رسم نمودار باشد .

دستور ginput

یکی از قابلیتهای matlab این است که می توان مختصات یک نقطه بر نمودار را با کلیک کردن بدست آورد و اینکار با دستور
`[x,y]=ginput(n)`
امکان پذیر می باشد

تذکر : پس از اجرای دستور رسم نمودار دستور ginput را اجرا می کنیم و ماوس را روی شکل نمودار در پنجره figure نگه می داریم . سپس با زدن Enter مختصات نقطه مورد نظر در متغیرهای x,y ذخیره می شود .
اگر مقدار n را ((1)) قرار دهیم یک نقطه و اگر از (n) استفاده نشود به صورت پیش فرض دو نقطه تقریبی محاسبه و تولید می شود .

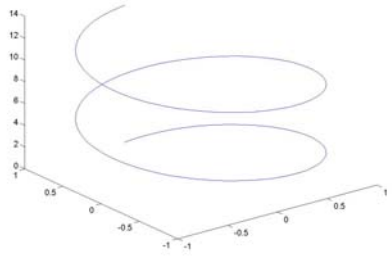
رسم نمودار های سه بعدی

در فضای سه بعدی ممکن است بخواهیم یک **منحنی فضایی** رسم کنیم یا یک **رویه**

برای رسم یک منحنی در فضای سه بعدی معادله منحنی که به صورت پارامتری می باشد را با دستور plot3 رسم می کنیم .

مثال : منحنی r(t)= (sin(t),cos(t),t) را در بازه 0 تا 4pi رسم کنید .
استفاده از دستور plot3 مشابه دستور plot است و لذا باید ابتدا t را تقسیم بندی کرده و به ازای آن مقادیر دیگر را محاسبه و در آرایه ذخیره نمود .

```
>> t=linspace(0,4*pi,100);
>> x=sin(t);
>> y=cos(t);
>> plot3(x,y,t)
```



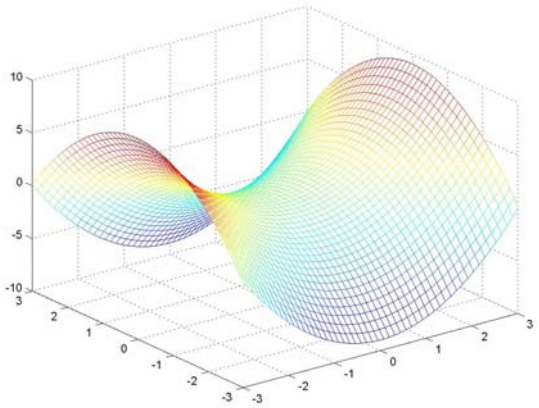
نکته : وقتی نموداری به صورت سه بعدی رسم می شود در پنجره figure دکمه ای به نام Roate3D را می توان فعال کرد و با فعال سازی این دکمه اگر کلیک موس را روی نمودار نگه دارید می توان شکل را در زوایای مختلف دوران داد .

رسم رویه ها در فضای سه بعدی

می دانیم که یک رویه به صورت z=f(x,y) بیان می شود . که در آن x و y متغیرهای مستقل و z متغیر وابسته می باشد . بنابراین بایستی x,y را مقدار دهی نمود و به ازای آن مقادیر مختلف z را به دست آورد .
تفاوتی که در رسم رویه با نمودار دو بعدی وجود دارد این است که وقتی محور x ها در بازه مورد نظر تقسیم بندی می شود و محور y ها را نیز تقسیم بندی می کنیم محل تقاطع این تقسیم بندی که به آن node یا گره می گویند را باید مشخص نمود .
سپس مقادیر z را به ازای این گره ها بدست می آوریم .
در Matlab با مشخص شدن مختصات گره ها که با دستور **meshgrid** انجام می شود می توان با دستور **mesh** یا دستور **surf** رویه مورد نظر را رسم نمود .

مثال : تابع $z = x^2 - y^2$ را رسم نمایید . x,y در بازه بین ۳ و ۳- تغییر می کند .

```
>> x=linspace(-3,3,50);
>> y=linspace(-3,3,50);
>> [a,b]=meshgrid(x,y);
>> z=a.^2-b.^2;
>> mesh(x,y,z)
```



توجه :

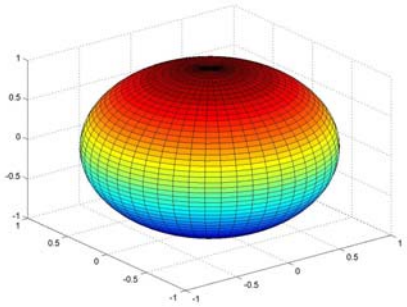
اگر به جای دستور mesh دستور surf استفاده کنیم رویه به صورت یک صفحه توپر نمایش داده خواهد شد و اگر از دستور contour استفاده کنید منحنی های کانتور یا به عبارتی منحنی های تراز رویه قابل رسم است

اگر دستور تابع دو متغیره مشخص باشد شبیه رسم نمودارهای دو بعدی می توان از دستورات ezmesh یا ezsurf استفاده نمود و نیازی به meshgrid ندارد .

با دستور **view** می توان زاویه جهت و زاویه آستانه نمودار سه بعدی را تعیین نمود .
وقتی نموداری در فضای سه بعدی رسم شده باشد و پنجره figure باز باشد، با این دستور که به صورت view([a,b]) نوشته می شود a زاویه آستانه یعنی زاویه نمایش نمودار حول محور قائم است و b زاویه جهت یعنی زاویه نمایش نمودار حول محور افق می باشد .

با دستور sphere می توان کره به صورت سه بعدی رسم کرد . برای این کار می نویسیم sphere(n) لذا کره ای به مرکز مبدا و متشکل از n صفحه در هر ردیف تشکیل می شود .

```
>> sphere(50);
```



البته می توان با دستور زیر کره ای به شعاع و مرکز دلخواه رسم کرد .

```
>>[x,y,z]=sphere(40)
>>x=3*x+5
>>y=3*y+6
>>z=3*x+2
>>surf(x,y,z)
```

با این دستورات کره ای به مرکز (5,6,2) و شعاع 3 با 40 صفحه تولید و رسم می شود .

با دستور cylinder می توان یک استوانه به مرکز مبدا رسم نمود . برای این کار می نویسیم cylinder(n) که استوانه ای به مرکز مبدا و متشکل از n صفحه و طول ۱ رسم خواهد شد .

تمرین :

۱- مکعبی به ضلع ۵ و مرکز مبداء مختصات به صورت سه بعدی رسم کنید . برای این کار شش صفحه را با هم در یک محور مختصات (به صورت یک گراف) به گونه ای رسم کنید که تشکیل یک مکعب بدهد .

۲- استوانه ای رسم کنید که در منحنی فضایی (cos(6t),sin(6t),t) محاط شده است .

دستورات انتگرال گیری و مشتق گیری

با دستور diff می توان از یک تابع تعریف شده مشتق گرفت .برای این کار ابتدا می توان ضابطه تابع را تعریف و داخل ' ' قرار داد . سپس با دستور diff مشتق می گیریم . مثال:

```
>> f='x^3-3*x+1';
>> diff(f)
      ans =3*x^2-3
```

اگر تابع چند متغیره ای تعریف شده باشد و بخواهیم مشتق جزئی بگیریم لازم است متغییر مورد نظر برای مشتق گیری در داخل دستور diff نوشته شود . مثال:

```
>> f='y*x^3-3*x*y^2';
>> diff(f,'x')
      ans =3*y*x^2-3*y^2
```

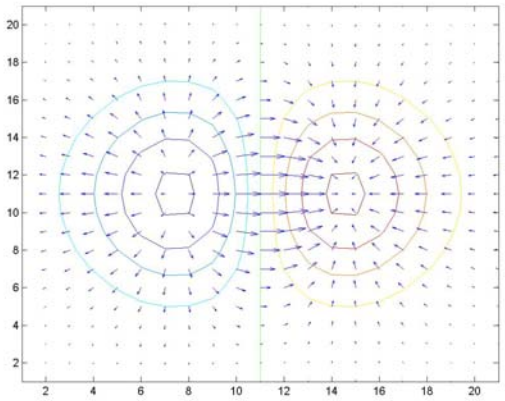
برای انتگرال گیری از دستور int استفاده می شود و شبیه به دستور diff عمل می کند . در صورتی که بخواهیم انتگرال معین از a تا b بگیریم کافیست بنویسید int(f,'x',a,b) مثال:

```
>> g='x^3-3*x^2';
>> int(g)
      ans =1/4*x^4-x^3

>> int(g,0,1)
      ans =-3/4

>> g='a*x^3-3*x^2';
>> int(g,'x')
      ans =1/4*a*x^4-x^3

>> int(g,'x',1,3)
      ans =20*a-26
```

با دستور gradient و دستور quiver می توان میدان های پتانسیل را رسم کرد .

مثال :

```
>>[x,y] = meshgrid(-2:.2:2, -2:.2:2);
>>z = x .* exp(-x.^2 - y.^2);
>>[px,py] = gradient(z,.2,.2);
>> contour(z)
>>hold on
>>quiver(px,py
```



M-file در Matlab

بیشتر دستورات و کدهای برنامه نویسی را می توان در پنجره ای به اسم M-File نوشته و با یک نام که از قاعده نامگذاری متغیرها تبعیت می کند نامگذاری کرد و هر در هر زمان که لازم باشد آن را اجرا نمود . این برنامه ها و دستورات که به صورت فایلهایی با پسوند m. ذخیره می شود را M-File می گویند .
از منوی file در فرمان new بر گزینه M-file کلیک کنید تا محیط ویرایشی M-file در دسترس قرار گیرد . در این محیط دستورالعملها را نوشته و طبق قاعده نامگذاری شناسه ها ذخیره نمایید . به صورت پیش فرض در داخل شاخه Matlab و شاخه Work ذخیره می شود . شما می توانید این شاخه اجرایی را تغییر دهید .

نحوه پیاده سازی ساختارهای تصمیم در Matlab

می توان در یک محیط M-File دستورات شرطی را به سادگی به صورت زیر نوشت

شرط ساده

```
if   گزاره
دستورات
end
```

شرط دوگانه

```
if    گزاره
دستورات
else
دستورات
end
```

شرط چند گانه

```
if    گزاره ۱
دستورات ۱
elseif    گزاره ۲
دستورات ۲
…
elseif    گزاره
دستوارت
end
```

مثال :
برنامه بنویسید که اگر 0.02^{20} از eps کمتر شد کلمه Yes نمایش داده شود .

```
if 0.02^(20)<eps
    'Yes'
end
```

مثال : در برنامه قبلی اگر 0.02^5 از eps کمتر شد کلمه Yes در غیر اینصورت No نمایش داده شود .

```
if 0.02^(20)<eps
      'Yes'
else
      'No'
end
```

دستور input

با این دستور می توان داده ای را از کاربر درخواست نمود و این داده را به داخل M-file برده و بر اساس آن یک سری پردازشها صورت پذیرد .
این دستور به شکل زیر نوشت می شود .

```
n=input ( ' عبارت ' )
```

لذا وقتی پردازشگر به خط برنامه بالا که دستور input درج شده برسد عبارت داخل پرانتز را نمایش می دهد تا توضیحاتی به کاربر در مورد نوع داده ای که باید وارد کند بدهد . سپس پردازشگر منتظر می ماند تا کاربر مقدار را وارد کند و دکمه Enter را فشار دهد . در این صورت مقدار وارد شده با نام n به داخل M-file منتقل می شود .

مثال : برنامه ای بنویسید که از کاربر مقدار k درخواست شود و به ازای این مقد ار وارد شده اگر 0.02^k از eps کمتر شد کلمه Yes در غیر اینصورت No نمایش داده شود .

```
K=input('Insert Your Number : ');
a=0.02^k;
if k<eps
   'Yes'
else
   'No'
end
```

مثال : برنامه ای بنویسید که عدد a را از کاربر بگیرد و اگر این عدد کمتر از صفر باشد پیغام خطایی ظاهر و در غیر اینصورت تابع f=sin(1/x) در بازه [0,a] رسم شود .

```
clc
a=input('a: ')
if a<=0
    error('"a" Could not be less than 0')
end
ezplot('sin(x)',[0,a])
```

دستورات تکراری در MAtlab

می توان دستورات تکراری و حلقه های تکرار را به سادگی به صورت یک M- File نوشته و اجرا نمود .

حلقه شمارشی For

در Matlab حلقه شمارشی For به صورت زیر نوشته می شود .

```
for    i  =  a:n
دستورات
end
```

در این حلقه به i که طبق قاعده نامگذاری متغیرها می تواند تعریف شود ((شمارنده حلقه)) می گویند . که این شمارنده ابتدا مقدار a را می گیرد و به ازای هر بار تکرار دستورات داخل حلقه به مقدار a یک واحد اضافه می شود تا مقدار آن به n برسد که در این صورت حلقه تمام می شود .

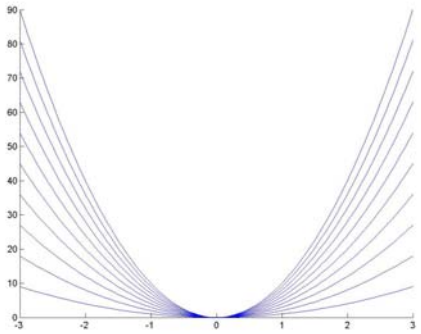
مثال : توسط یک حلقه مقادیر $2,4,8,16,\ldots,2^{13}$ را به عنوان عناصر یک آرایه تولید کنید .

```
clc
for i=1:13
    a(i)=2^i;
end
a
```

مثال : برنامه ای بنویسید که به ازای m دلخواه و صحیح و مثبت که کاربر وارد می کند توابع

X^2 ، X^4 ، X^6 ، ... X^{2m} در یک محور مختصات رسم شود .

```
m=input('Insert Integer Number :');
x=linspace(-3,3,100);
hold on
for i=1:2*m
    y=x.^2*i;
end
```



حلقه های تو در تو

دستورات داخل یک حلقه می تواند خود یک حلقه باشد . که در این صورت به نوع حلقه ها حلقه های تو در تو گویند . همچنین می توان از دستورات شرطی در داخل حلقه استفاده کرد .

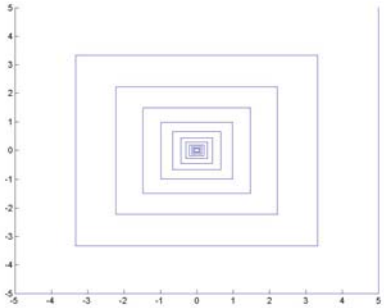
مثال : برنامه بنویسید که توسط آن ماتریسی با مرتبه ۶ تولید شود . روی قطر اصلی آن عناصر $1/6,\ldots,1/3$, $1/2$, 1 باشد و مابقی عناصر صفر است .

```
for i=1:6
        if i==j
            A(i,j)=1/i
        else
            A(i,j)=0
        end
    end
end
```

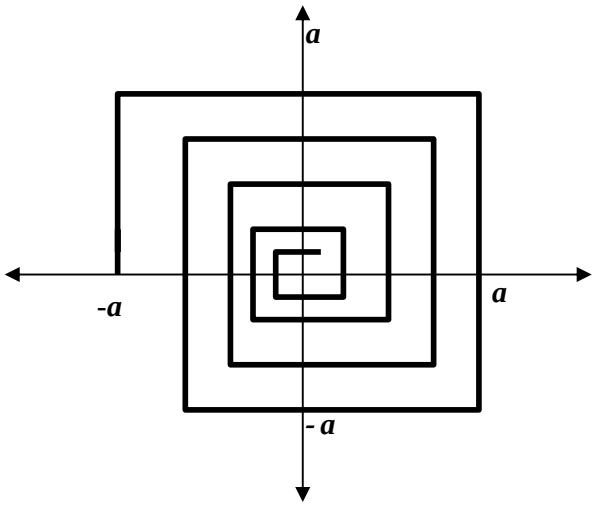
```
A =
   1.0000         0         0         0         0         0
        0    0.5000         0         0         0         0
        0         0    0.3333         0         0         0
        0         0         0    0.2500         0         0
        0         0         0         0    0.2000         0
        0         0         0         0         0    0.1667
```

مثال : برنامه ای بنویسید که مربعات هم مرکز به مبدا مختصات تولید شود . به طوریکه مربع خارجی ضلع برابر ۱۰ داشته باشد و ضلع هر مربع داخلی ۲/۳ مربع خارجی باشد . تعداد اینگونه مربعات ۱۲ است .

```
a=5;
hold on
for i=1:12
x=[-a,-a,a,a,-a];
y=[-a,a,a,-a,-a];
plot(x,y)
a=(2/3)*a;
end
```



تمرین : برنامه ای بنویسید که از طریق آن بتوان مربعات ارشمیدس به صورت زیر به گونه ای رسم که از ضلع 2a=10 و مرکز مبدأ شروع و ضلع هر مربع داخلی ۰/۸ مربع خارجی باشد . و این کار ۲۰ بار تکرار شود .



حلقه تکرار شرطی while

در Matlab این نوع حلقه به صورت زیر نوشته می شود .

```
while    گزاره شرطی
دستورات
End
```

مثال : برنامه ای بنویسید که مربعات هم مرکز به مبدا مختصات تولید شود . به طوریکه مربع خارجی ضلع برابر ۱۰ داشته باشد و ضلع هر مربع داخلی ۲/۳ مربع خارجی باشد . رسم مربعات تا آنجا ادامه یابد که ضلع مربع داخلی به کمتر از ۰/۰۱ برسد.

```
a=5;
hold on
while a>0.01
   x=[-a,-a,a,a,-a];
   y=[-a,a,a,-a,-a];
   plot(x,y)
   a=(2/3)*a;
end
```

تذکر : در حلقه while تعداد تکرار دستورات مشخص نیست . بنابراین اگر نیاز داشته باشیم که بدانیم دستورات چند بار تکرار شده است بایستی متغیری که ابتدا مقدار یک به آن می دهیم در داخل حلقه گذاشته تا به ازای هر بار تکرار دستورات به مقدار این متغییر یک واحد اضافه شود . به عنوان مثال اگر در مثال قبلی بخواهیم بدانیم چند مربع رسم شده معلوم نیست ولی با اضافه کردن یک متغیر به صورت زیر می توان به هدف رسید.

```
a=5;
n=0;
hold on
while a>0.01
   x=[-a,-a,a,a,-a];
   y=[-a,a,a,-a,-a];
   plot(x,y)
   a=(2/3)*a;
   n=n+1;
end
```

با هر بار اجرای حلقه مقدار متغیر n یک واحد اضافه می شود .بنابراین با اتمام حلقه تعداد تکرار حلقه برابر مقدار عددی n است .

تمرین : برنامه ای بنوسیید که توسط آن مربعات ارشمیدس با نقطه شروع ۵- به گونه ای باشد که ضلع داخلی ۰/۷ ضلع بیرونی باشد و رسم این مربعات تا آنجا ادامه یابد که فاصله تا مبدا کمتر از ۰/۰۱ شود .

دستور getframe و نحوه تولید تصاویر متحرک

در بسیاری از مواقع لازم است تا بتوانیم نمودارهای متحرکی بر صفحه نمایش تولید کنیم مانند نمودارهای مربوط به معادلات موج یا ارتعاشات و یا شبیه سازی فرایندهای فیزیکی
برای این کار نمودارهای مختلف را با دستورات رسم نمودار که قبلا گفته شد تولید می کنیم و با دستور **getframe** اطلاعات این نمودارها را در حافظه و به صورت یک آرایه ذخیره می کنیم . سپس با دستور **movie** می توان این داده های ذخیره شده مربوط به فریم ها را پشت سر هم نمایش داد .

مثال : موج سینوسی `sin(2xt)` را وقتی `x` در بازه `[-pi,pi]` و `t` از ۰ تا ۵ تغییر می کند رسم کنید .

```
x=linspace(-pi,pi,100);
t=linspace(0,5,200);
for i=1:200
    y=sin(2*x*t(i));
    plot(x,y)
    m(i)=getframe;
end
movie(m)
```

مثال : یک رویه به معادله $x^2y-xy+3x$ را که ۳۶۰ درجه حول محور قائم دوران می کند به صورت یک تصویر متحرک نمایش دهید .
برای تولید تصویر متحرک می توانیم بازه بین صفر و ۳۶۰ درجه را به چند قسمت تقسیم نماییم . سپس رویه را رسم نموده و با استفاده از تقسیمات تعیین شده از زوایای مختلف رویه را نمایش و فریم آن را در آرایه m ذخیره نماییم . سپس با دستور movie این فریم ها را پشت سر هم نمایش دهیم تا به صورت تصویر متحرک ایجاد شود .

```
t=linspace(0,360,120);
f='x^2*y-x*y+3*x';
ezsurf(f)
for k=1:120
view([t(k),15])
m(k)=getframe;
end
movie(m)
```

مثال : متحرکی بر مسیر منحنی مارپیچ `(cos(t),sin(t),t)` در حال حرکت است . حرکت این متحرک را نمایش دهید .

```
t=linspace(0,6*pi,120);
x=sin(t);
y=cos(t);
for i=1:120
hold on
plot3(x,y,t,'r :')
plot3(x(i),y(i),t(i),'o')
view([30,15])
m(i)=getframe;
clf
end
movie(m)
```

توابع در Matlab (Functions)

یکی از ساختارهای مهم در تمام زبانهای برنامه نویسی توابع می باشند . با استفاده از توابع می توان چندین دستورالعمل یا یک قطعه برنامه را نامگذاری کرد و در مواقع مورد نیاز اجرا نمود .

در Matlab توابع به صورت زیر نوشته می شود .

```
function  name ( آرگومان)
دستورات
end
```

name که نام تابع و دلخواه است از قاعده نامگذاری شناسه ها تبعیت می کند و بایستی تابع را به صورت یک M-file نوشته و با نامی دقیقاً مشابه name ذخیره نمود . ممکن است در یک تابع آرگومان مورد نیاز باشد یا می توان در صورت عدم نیاز از نوشتن آن صرف نظر گرد .

مثال : تابعی بنویسید که با نام tm(n) ماتریسی با n سطر و n ستون تولید شود و عناصر روی قطر اصلی آن 1,2,3,…,n باشد و مابقی عناصر صفر شود .

```
function tm(n)
for i=1:n
    for j=1:n
        if i==j
            A(i,j)=i;
        else
            A(i,j)=0;
        end
    end
end
A
end
```

کدهای بالا را به صورت M-file نوشته و با اسم تابع یعنی tm ذخیره می کنیم . پس از این هرگاه tm(n) را به ازای n مورد نظر اجرا کنید ، ماتریس مورد نظر تولید خواهد شد . به عنوان مثال اگر tm(5) را اجرا کنیم ، ماتریس زیر را خواهیم داشت .

```
A =
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
```

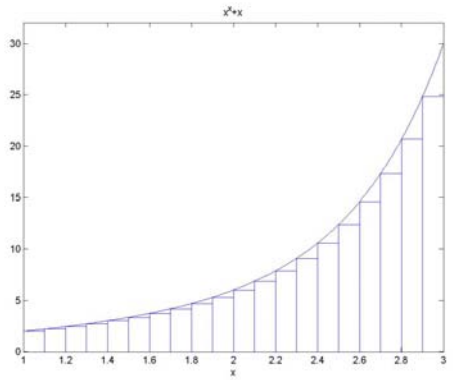
مثال : تابعی به نام kore بنویسید که با kore(r,a,b,c) بتوان کره ای به مرکز (a,b,c) و شعاع r رسم نمود .

```
function kore(r,a,b,c)
[x,y,z]=sphere(50);
x=r*x+a;
y=r*y+b;
z=r*z+c;
surf(x,y,z)
end
```

تمرین : تابعی به نام dmatrix(n) بنویسید که توسط آن ماتریسی پایین مثلثی تولید شود . روی قطر اصلی آن اعداد 1,2,…,n باشد و زیر قطر اصلی همه عناصر برابر i+j و بالای قطر اصلی صفر باشد .

مثال : تابعی بنویسید به نام myint که توسط myint(a,b,n) انتگرال تابع f=x^x+x از a تا b با استفاده از جمع پایین ریمان با n تقسیم محاسبه شود .

```
function myint(a,b,n)
h=(b-a)/n;
s=0;
r=a;
for i=1:n
    k=r^r+r;
    s=s+k*h;
    r=r+h;
end
s
```



تمرین: مثال قبلی را به گونه ای بنویسید که نمودار تابع به همراه مستطیل های ریمان رسم شود . و مقدار انتگرال به عنوان تیتر نمودار درج شود .

تمرین : مثال قبل را با استفاده از جمع بالای ریمان بنویسید .

تمرین : تابع قبلی را به گونه ای بنوسید که بتوان از طریق آن مقدار انتگرال را به گونه ای حساب کرد که جمع بالا و پایین ریمان از eps کمتر شود .

راهنمایی : می توان اول n=10 گرفت و مقدار جمع بالا و پایین را حساب کرد اگر از eps کمتر نشد دوباره $n=n^2$ قرار داده و این کار را آنقدر ادامه می هیم تا به نتیجه برسد .

مثال : حرکت نوسانی یک توپ کره ای شکل که در راستای قائم و طبق |sin(t)/t| نوسان می کند را به صورت تصویر متحرک نمایش دهید

```
clear all
clc
t=linspace(0,30,100);
h=abs(3*sin(t)./t);
for i=1:100
    kore(0.8,0,0,h(i))
    axis([-2,2,-2,2,0,3])
    n(i)=getframe
end
movie(n)
```

توضیح : با توجه به اینکه در مثالهای قبلی تابعی به نام kore نوشته بودیم که می توانست کره به مرکز و شعاع مورد نظر را رسم کند . این تابع را در داخل این حلقه تکرار استفاده کرده ایم .

تمرین : برنامه ای بنویسید که حرکت کره ای به شعاع 0.3 را که بر مسیر sin(t)/t حرکت می کند را در بازه [0,50] x به صورت تصویر متحرک نمایش دهد

راهنمایی

برای سادگی کار می توان قبلا کره را به صورت یک تابع تعریف کرد و آن را به مرکزهای مختلفی که از sin(t)/t به دست می آید نمایش داد و getframe کرد .

نمایش مقادیر متغیر ها در داخل یک عبارت fprintf

در بسیاری از مواقع نیاز است تا بتوانیم مقادیری که به صورت یک متغیر ذخیره شده است را در داخل یک عبارت قرار دهیم . این کار را می توان با دستور fprintf انجام داد . که به صورت زیر نوشته می شود .

```
fprintf( ' عبارت %g عبارت %g ' , a,b)
```

طبق این دستور مقادیری که در متغیرهای a,b ذخیره شده است به ترتیب در محلهایی که با %g مشخص شده است قرار می گیرد .

مثال : برنامه ای بنویسید که توسط دستور input دو مقدار را بگیرد و جمع این دو را با پیغامی نمایش دهد .

```
clc
a=input('insert a :');
b=input('insert b :');
c=a+b;
fprintf('Add %g and %g is :%g',a,b,c)
```

توجه : اگر در داخل عبارت توضیحی که در fprintf نوشته می شود شما \n قرار دهید ادامه آن عبارت در سطر جدید قرار می گیرد .

یافتن ریشه های معادلات غیر خطی

در بسیاری از مسائل نیاز به آن داریم تا جوابی برای یک معادله بدست آوریم یا به عبارتی ریشه های یک معادله خطی یا غیر خطی را به دست آوریم برای این کار روشهای مختلفی وجود دارد که به تفصیل در محاسبات عددی بحث می شود . در این بخش می خواهیم فقط برنامه کامپیوتری مربوط به این روشها را توضیح دهیم .

یکی از روشهای متداول برای یافتن جواب روش نیوتون است . با استفاده از این روش از یک جواب اولیه و غیر دقیق شروع می کنیم و با رابطه

$$X(n+1) = X(n) - f(x(n)) / f'(x(n))$$

مثال : ریشه معادله $f= e^x -x^3-3x^2-x$ را با جواب اولیه x=7 تا تقریب مناسبی به دست آورید .

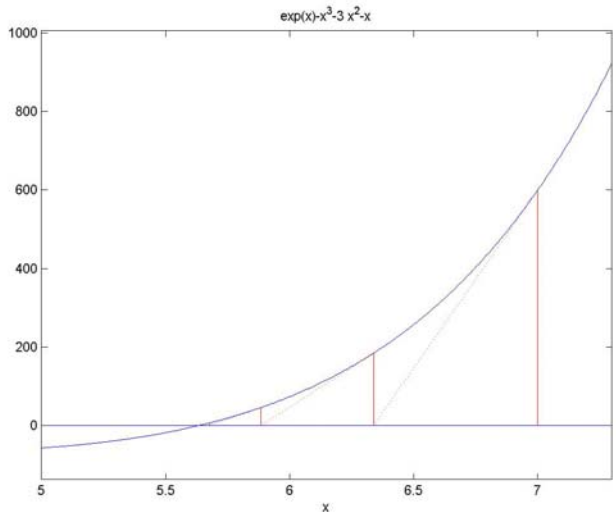
```
clc
x=7
for i=1:10
a=exp(x)-x^3-3*x*2-x;
b=exp(x)-3*x^2-6*x-1;
x=x-a/b
end
```

برای مشاهده نتایج و نحوه همگرایی می توانیم به صورت زیر نیز بنویسیم .

```
clc
x=7
for i=1:20
a=exp(x)-x^3-3*x*2-x;
b=exp(x)-3*x^2-6*x-1;
x=x-a/b;
fprintf('(%g):x=%g   f=%g \n',i,x,a)
end
```

```
(1):x=6.2228   f=704.633
(2):x=5.59471   f=219.587
(3):x=5.20534   f=54.7172
(4):x=5.13602   f=4.7636
(5):x=5.15965   f=-1.3958
(6):x=5.1496   f=0.625448
(7):x=5.1536   f=-0.243454
(8):x=5.15196   f=0.100646
(9):x=5.15263   f=-0.0406253
(10):x=5.15236   f=0.0165598
(11):x=5.15247   f=-0.00672341
(12):x=5.15242   f=0.00273417
(13):x=5.15244   f=-0.00111116
(14):x=5.15243   f=0.000451694
(15):x=5.15243   f=-0.000183596
```

تمرین : مثال قبلی را به گونه ای بنویسید که به صورت هندسی روش نیوتون را نمایش دهد . (شکل زیر)



تمرین : مثال قبلی را با استفاده از حلقه `while` به گونه ای بنویسید که خطا را کمتر از 10^{-6} نماید .

تمرین : تابعی بنویسید به نام `ns(e,x)` که با استفاده از آن `sin(1/x)` با نقطه اولیه `x` و تا دقت `e` ریشه بهبود یابد .

تمرین : برنامه ای بنویسید که توسط آن بتوان جواب یک دستگاه معادلات `AX=b` را که ماتریس ضرایب آن یعنی ماتریس `A` یک ماتریس سه قطری است را حساب کند .

تمرین : گلوله ای کره ای شکل به شعاع ۰/۵ که در فنر محاط شده است توسط فنر طبق رابطه `sin(t)/t` نوسان می کند حرکت نوسانی آن را در ۳۰ ثانیه اول نمایش دهید .