به نام یزدان پاک

فهرست مطالب

گرد کردن اعداد اعشاری در متلب:

بردارها و ماتریس ها در متلب:

ترسیم گرافیکی توابع در متلب:

معرفی نرم افزار متلب:

متلب یک سیستم جامع نرم افزاری برای محاسبات ریاضی و محاسبات تکنیکی می باشد . نرم افزار متلب از زبان برنامه نویسی سطح بالا استفاده می کند که شامل صدها دستور برای محاسبات ریاضی می باشد اما نباید نگران تعداد زیاد دستورها در متلب باشید زیرا معمولا برای پروژه خود تنها به تعداد اندکی از آنها نیاز دارید و همچنین با Help قوی نرم افزار می توانید دستوراتی را که لازم دارید ، بیابید .
کاربردهای نرم افزار متلب بسیار وسیع می باشد . شما با متلب حتی می توانید صدا و یا انیمیشن بسازید .
شرکت توسعه دهنده نرم افزار متلب ، شرکت MathWorks می باشد.

آشنایی با محیط اصلی نرم افزار متلب:

هنگامی که وارد محیط نرم افزار متلب بشوید ، پنجره اصلی نرم افزار ، خود شامل پنجره های زیر می باشد :

پنجره : Command

در پنجره Command می توانیم دستورات خود را نوشته و سپس با فشار دادن کلید enter از کیبورد ، نتایج اجرای دستورات را ، در همین پنجره ، مشاهده کنیم .

پنجره : Workspace

در پنجره Workspace ، لیستی از متغیرهایی که به وسیله دستورات در متلب تعریف شده است ، نمایش داده می شود .

پنجره : Current Folder

در پنجره Current Folder می توانیم پوشه ای که در آن فایل های متلب مورد نظرمان وجود دارد را مشاهده کنیم و به آسانی پوشه و یا فایل های مورد نظرمان را بیابیم .

پنجره : Command History

در پنجره Command History ، لیستی از دستوراتی که در متلب اجرا کرده ایم ، نمایش داده می شود.

نحوه اجرای دستورات در متلب:

بهترین روش برای یادگیری متلب این است که از دستورات ساده شروع کنید و نتایج آن را مشاهده کنید ، هرگاه دستوری را در متلب اشتباه وارد کنید ، متلب پیغام خطایی در پنجره Command نمایش می دهد که نوع خطا و همچنین محل خطا را برای شما مشخص می کند . پس با دستورات ساده شروع کنید ، نتایج و یا پیام های خطا را مشاهده کرده و به تدریج به سراغ دستورات پیچیده تر بروید.

نوشتن دستورات در پنجره: Command

ابتدا باید در پنجره Command کلیک کنید تا فعال شود ، سپس می توانید دستورات مورد نظر خود را وارد کرده و با فشار دادن کلید enter از کیبورد ، نتیجه اجرای دستورات را در همان پنجره ببینید.

مثال :

دستور ساده زیر را وارد می کنیم:

```
1+3
```

نتیجه به صورت زیر در پنجره Command نمایش داده می شود:

نتیجه :

```
ans =

     4
```

ansنمایشگر ابتدای کلمه answer می باشد . هنگامی که در متلب برای نتیجه یک محاسبه ، نامی انتخاب نشده باشد ، خود نرم افزار متلب ، نام ans را برای آن انتخاب می کند ، یعنی این که متغیری به نام ans با مقدار 2 ایجاد می کند . چنانچه برای محاسبات بعدی به این عدد احتیاج داشته باشید باید حتما نام دیگری برای آن انتخاب کنید (مثلا a=1+3) زیرا نرم افزار متلب به دستور بعدی که نام متغیری برای آن در نظر گرفته نشده باشد ، دوباره نام ans را اختصاص می دهد و عملا مقدار قبلی آن پاک می شود.

نکته :

چنانچه تمایل داشته باشید که دستورات قبلی و نتایج آنها که در پنجره Command نمایش داده شده اند پاک شوند تنها کافی است که بر روی قسمتی از پنجره Command کلیک سمت راست کرده و گزینه Clear Command Window را انتخاب نمایید . باید دقت داشته باشید که با این کار ، تنها دستورات و نتایج نشان داده شده در پنجره Command پاک می شوند اما متغیرهایی که توسط این دستورات در متلب تعریف شده اند ، همچنان وجود دارند و می توان از آنها استفاده نمود . دستوری که متغیرها را به طور کامل از متلب پاک می کند ، دستور clear می باشد که در دروس بعدی در مورد آن صحبت خواهیم کرد . همچنین چنانچه برنامه متلب را ببندید ، تمامی متغیرهای تعریف شده در متلب پاک می شوند و دفعه بعد که متلب را باز کنید هیچ متغیری در آن تعریف نشده است.

نکته :

چنانچه بخواهید نتیجه اجرای خطی از دستورات در پنجره Command نمایش داده نشود تنها کافی است که در پایان آن خط از دستورات ، علامت ((;)) را بنویسید . با به کار بردن این علامت در پایان هر خط ، متلب دستورات آن خط را اجرا کرده و نتیجه محاسبات را در متغیرها ذخیره می کند اما نتیجه محاسبات را در پنجره Command نمایش نخواهد داد.

انواع متغیرها و مقداردهی به متغیرها در متلب:

یکی از ویژگی های متلب این است که احتیاجی نیست که حتما نوع متغیر را در همان ابتدای برنامه مشخص کنیم و با مقادیری که در طول برنامه به متغیر نسبت داده می شود ، نوع متغیر به صورت خود به خود تعیین می شود .

در تعریف نام متغیرها باید دقت داشته باشید که متلب نسبت به کوچک یا بزرگ بودن حروف حساس می باشد.

در نرم افزار متلب ، انواع مختلفی از متغیرها وجود دارد که عبارتند از:

متغیرهای عددی:

این متغیرها می توانند دارای مقادیر عددی باشند . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم به متغیر A مقدار 2 را نسبت بدهیم . باید بنویسیم:

```
A=2
```

نتیجه :

```
A =

     2
```

متغیر های رشته ای (string)

چنانچه متغیری را بخواهیم به صورت یک رشته از حروف تعریف کنیم باید از علامت ' استفاده کنیم.

مثال :

```
s='this is a string'
```

نتیجه :

```
s =

this is a string
```

نکته :

دقت شود که استفاده از علامت ' برای ایجاد رشته ها ضروری است و چناچه از این علامت استفاده نشود ، با پیغام خطا مواجه می شویم . این موضوع را در مثال زیر نشان داده ایم:

مثال :

```
s=this is a string
```

نتیجه :

```
??? s=this is a string
           |
Error: Unexpected MATLAB expression.
```

متغیرهای سمبلیک:

گاهی نیاز است که متغیر تنها به صورت سمبلیک (مثلا با حرف x) تعریف شود تا با آن معادلاتی را به صورت نمادین حل کنیم . در مورد متغیرهای سمبلیک در مباحث دیگر به صورت مفصل صحبت خواهیم کرد.

چند دستور ساده:

دستور whos :

چنانچه تعداد متغیرهایی که در متلب تعریف کرده اید از حدی بیشتر شود ، به سختی می توانید نام آنها را به یاد آورید . برای آنکه بتوانید تمامی متغیرهایی که در متلب تعریف کرده اید را به صورت فهرست وار ببینید تنها کافی است که دستور whos را اجرا کنید . لیست تمامی متغیرهای تعریف شده در متلب در خروجی نمایش داده می شود و نوع هر متغیر ، فضای اختصاص داده شده به آن و اندازه آن را می توانید مشاهده کنید.

مثال :

```
whos
```

نتیجه :

```
Name        Size              Bytes  Class     Attributes

  A         1x1                   112  sym
  B         1x1                   112  sym
  C         1x1                   112  sym
  ans       1x1                   112  sym
  x         1x1                   112  sym
  y         1x1                   112  sym
```

اما باید دقت کنید که در فهرست فوق ، مقادیر متغیرها نمایش داده نشده است . برای مشاهده مقدار هر متغیر تنها کافی است که نام متغیر را در پنجره command نوشته و سپس کلید enter از کیبورد را فشار دهید ، مقدار متغیر نمایش داده خواهد شد .

علاوه براین Workspace به صورت گرافیکی لیستی از تمامی متغیرهای تعریف شده در متلب را نمایش می دهد.

دستور clear :

این دستور برای پاک کردن متغیرهای تعریف شده در متلب به کار می رود . این دستور را می توان به شیوه های زیر به کار برد:

clear	تمامی متغیرهای تعریف شده در متلب را پاک می کند
clear all	تمامی متغیرهای تعریف شده در متلب را پاک می کند
clear x y	تنها متغیرهای x و y را پاک می کند

نکته بسیار مهم :

یک برنامه نویس متلب معمولا اولین دستوری که در برنامه خود به کار می برد ، دستور clear all می باشد زیرا امکان این که متغیرهایی که قبلا در متلب توسط برنامه های قبلی تعریف شده اند در

برنامه جدید اختلال ایجاد کنند ، زیاد است . این نکته در اجرای برنامه های طولانی و پیچیده از اهمیت زیادی برخوردار است.

دستورclc:

دستور clc در متلب برای پاک کردن اطلاعات نمایش داده شده در پنجره command به کار می رود . یعنی هر زمان که خواستید تمامی اطلاعات نمایش داده شده در پنجره) command البته تا این لحظه و نه برای اطلاعاتی که در آینده نمایش داده می شوند) ، پاک شوند تنها کافی است که دستور زیر را بنویسید:

```
clc
```

البته روش دیگری نیز برای پاک کردن اطلاعات نمایش داده در پنجره command نیز وجود دارد که به صورت دستی است ، برای این منظور ابتدا باید بر روی پنجره command کلیک سمت راست نموده و سپس گزینه Clear Command Window را انتخاب کنید.

عملگرهای ریاضی در متلب:

در متلب برای استفاده از عملگرهای ریاضی ، دو شیوه به کار رفته است . شیوه اول برای عملگرهای بسیار رایج می باشد مانند جمع ، تفریق ، ضرب ، تقسیم و ... که متلب برای هر کدام از آنها یک نماد را در نظر گرفته است یعنی شما در هنگام نوشتن دستورات تنها کافی است که از کیبورد نماد مربوط به آن عملگر را در خط دستورات وارد کنید (مثلا زدن دکمه مربوط به نماد * از کیبورد برای عملگر ضرب).
شیوه دوم ، استفاده از تابع است . متلب برای عملگرهای ریاضی که میزان استفاده از آنها زیاد نیست ، دستوراتی را ساخته است و شما باید نام آن دستورات و نحوه استفاده از آنها را بدانید (مثلا برای عملگر رادیکال 2 ، دستور sqrt در نظر گرفته شده است).
در ادامه عملگرهای مربوط به هر دو شیوه را شرح خواهیم داد.

عملگرهای ریاضی مشخص شده با نماد در متلب:

همان طور که گفتیم برای هر یک از این عملگرهای ریاضی ، در متلب یک نماد خاص اختصاص یافته است . در جدول زیر نمادهای مربوط به عملگرهای ریاضی در متلب آمده است:

نماد در متلب	عملگر
+	جمع
-	تفریق
*	ضرب
/	تقسیم
^	به توان

مثال :

```
3*2-(3*4)/2+4^2
```

نتیجه :

```
ans =

    16
```

مثال :

برای به توان 2 رساندن عدد 3 این گونه می نویسیم:

```
3^2
```

نتیجه :

```
ans =
```

عملگرهای ریاضی مشخص شده با دستور در متلب:

در متلب برای هر کدام از این عملگرهای ریاضی یک دستور در نظر گرفته شده است . برای نمونه تعدادی از این عملگرها را در جدول زیر آورده ایم:

دستور در متلب	عملگر
sqrt	رادیکال 2
log	Ln (لگاریتم طبیعی)
log10	Log (لگاریتم بر مبنای 10)

مثال :

با دستور sqrt ، رادیکال 2 عدد 4 را محاسبه می کنیم:

```
A=sqrt(4)
```

نتیجه :

```
A =

     2
```

مثال :

با دستورlog10 ، لگاریتم بر مبنای 10 یا log2 لگاریتم بر مبنای 2 و...

لگاریتم بر مبنای 10 عدد 100 را محاسبه می کنیم:

```
A=log10(100)
```

نتیجه :

```
A =

     2
```

نمادهای پرکاربرد در متلب:

در نرم افزار متلب برای برخی مقادیر که معمولا پرکاربرد می باشند ، نمادهایی منحصر به فرد در نظر گرفته شده است . به عنوان مثال برای عدد مشهور ((پی)) که در ریاضیات بسیار به کار می رود ، نماد pi انتخاب شده است . به مثال زیر توجه کنید:

مثال :

```
A=pi
```

نتیجه :

```
A =

    3.1416
```

برخی از نمادهای پرکاربرد در نرم افزار متلب را در جدول زیر نمایش داده ایم:

pi	عدد مشهور پی (3.1415)
Inf	بینهایت
i	موهومی
eps	کوچکترین عدد ممکن (اپسیلون) (2.2204e-016)
NaN	به صورت یک عدد تعریف نشده است (مبهم) (در حالت 0/0 رخ می دهد)
realmax	بزرگترین عدد حقیقی مثبت (1.7977e+308)
realmin	کوچکترین عدد حقیقی مثبت (2.2251e-308)

استفاده از Help متلب:

استفاده از Help متلب می تواند در موارد مختلف به شما کمک کند . در اینجا می خواهم نکته ای را به شما بگویم که بسیار مهم است ((Help)) : متلب در برخی مواقع بهترین انتخاب نیست((! بله درست خواندید . گاهی اوقات Help متلب می تواند بسیار آزاردهنده باشد و آن زمانی است که شما می خواهید دستوری را برای یک عمل خاص بیابید اما نام آن دستور را نمی دانید و مجبورید با کلماتی که آن عمل خاص را توصیف می کنند به جستجو بپردازید . به دلیل حجم زیاد دستورها و ابزارهای مختلف متلب باید خوش شانس باشید که کلمه درست را بیابید و با آن کلمه نام دستور مورد نظرتان را پیدا کنید . در این گونه موارد اگر از جستجو در Help متلب خسته شدید ، در اینترنت به جستجو بپردازید و در کتاب های آموزش متلب در فصل هایی جستجو کنید که حدس می زنید آن دستور در آنها کاربرد دارد.
اما زمانی که نام دستور را بدانید ، Helpمتلب مرجع بسیار مناسبی است زیرا نحوه دقیق استفاده از آن دستور و همچنین دستورهای با عملکرد مشابه آن دستور را به شما نشان خواهد داد.

باز کردن پنجره Help متلب:

در پنجره اصلی نرم افزار متلب از منوی Help ، گزینه Product Help را انتخاب کنید . پنجره Help متلب باز می شود . در پنجره Help قسمتی برای جستجو وجود دارد که می توانید با آن کلمه یا کلمات مورد نظر خود را وارد کرده و به جستجو در صفحات Help متلب بپردازید.

دستور format:

به صورت پیش فرض در متلب اعداد تا 4 رقم اعشار نمایش داده می شوند . اگر بخواهیم به فرمت طولانی تر تبدیل کنیم، که تعداد ارقام بیشتری را نشان دهد:

Format long

تا 15 رقم اعشار نمایش میدهد.

مثال :

```
format long
pi
```

نتیجه :

```
ans =

   3.141592653589793
```

برای نمایش اعداد تا 2 رقم اعشار :

Format bank

اگر بخواهیم به فرمت قبل برگردد: Format short

هر بار که برنامه ی متلب را ببندیم و باز کنیم به فرمت قبل خود که همان 4 رقم اعشار است بر می گردد .

نکته :

موارد دیگر format را میتوانید به راحتی در help مشاهده کنید.

ایجاد نقاط با فواصل خطی:

```
linspace(a,b)
```

که a عدد ابتدا و b عدد انتهاست.

پیش فرض این است که 100 نقطه تعریف شده است.

مثال :

```
Linspace(1,5,5)
```

نتیجه :

```
ans =

     1    2    3    4    5
```

5 نقطه می دهد، با فواصل یکسان.

ایجاد نقاط با فواصل لگاریتمی:

دستور logspace اعداد را به صورت لگاریتمی به صورت های زیر نمایش میدهد.

```
logspace (a,b)
```

دستور فوق بین 10^a و 10^b پنجاه عدد انتخاب میکند

```
logspace (a,b,n)
```

این دستور مانند دستور قبل عمل میکند با این تفاوت که عدد آخر(n) تعداد نقاط را برای ما مشخص میکند.

مثال :

```
logspace (1,2,5)
```

نتیجه :

```
ans =

   10.0000   17.7828   31.6228   56.2341  100.0000
```

گرد کردن اعداد اعشاری در متلب:

گاهی تعدادی عدد داریم که دارای بخش اعشاری می باشند اما می خواهیم آنها را به یک عدد صحیح ، تبدیل کنیم (گرد کنیم) . در متلب برای این منظور ، چند تابع در نظر گرفته شده است که انتخاب از میان آنها به این بستگی دارد که عمل گرد کردن را به چه صورت بخواهیم انجام دهیم.
در جدول زیر این دستورات و تفاوت کاربرد آنها را به طور مختصر شرح داده ایم:

دستور round	گرد کردن به سمت نزدیکترین عدد صحیح
دستور fix	گرد کردن به سمت صفر
دستور ceil	گرد کردن به سمت مثبت بینهایت
دستور floor	گرد کردن به سمت منفی بینهایت

دستور round:

چنانچه از دستور round برای گرد کردن عدد اعشاری مورد نظرمان استفاده کنیم ، آنگاه آن عدد اعشاری به نزدیکترین عدد صحیح تبدیل خواهد شد . به مثال زیر توجه کنید:

مثال :

```
round(1.9)
```

نتیجه :

```
ans =

     2
```

دستور fix:

فرض کنید از دستور fix برای گرد کردن یک عدد اعشاری استفاده کنیم ، چون آن عدد اعشاری بین دو عدد صحیح قرار گرفته است باید یکی از آن دو به عنوان گرد شده آن عدد اعشاری انتخاب شود . وقتی می گوییم گرد کردن به سمت صفر یعنی اینکه از بین آن دو عدد صحیح ، عددی انتخاب می شود که به صفر نزدیکتر باشد . به مثال زیر توجه کنید:

مثال :

```
fix(1.9)
```

نتیجه :

```
ans =

     1
```

دستور ceil :

اگر از دستور ceil برای گرد کردن یک عدد اعشاری استفاده کنیم ، از میان دو عدد صحیحی که در دو طرف عدد اعشاری قرار گرفته اند ، عددی انتخاب می شود که به مثبت بینهایت نزدیکتر باشد . به مثال زیر توجه کنید:

مثال :

```
ceil(1.2)
```

نتیجه :

```
ans =

     2
```

دستور floor :

اگر از دستور floor برای گرد کردن یک عدد اعشاری استفاده کنیم ، از میان دو عدد صحیحی که در دو طرف عدد اعشاری قرار گرفته اند ، عددی انتخاب می شود که به منفی بینهایت نزدیکتر باشد . به مثال زیر توجه کنید:

مثال :

```
floor(1.9)
```

نتیجه :

```
ans =

     1
```

بردارها و ماتریس ها در متلب:

نرم افزار متلب بر اساس کار کردن با بردارها و ماتریس ها ساخته شده است.

بردارها در متلب:

یک بردار ، فهرستی از اعداد می باشد که پشت سرهم چیده شده اند . به هر کدام از این اعداد ، یک عنصر بردار می گوییم . بردارها را با روش های مختلفی می توان در متلب تعریف نمود .
روش اول : استفاده از علامت کاما ((،)) برای جدا کردن عناصر بردار و قرار دادن آنها درون براکت
مثال :

```
A=[1,2,3,4]
```

نتیجه :

```
A =

     1     2     3     4
```

روش دوم : استفاده از فاصله برای جدا کردن عناصر بردار و قرار دادن آنها درون براکت
مثال :

```
A=[1 2 3 4]
```

نتیجه :

```
A =

     1     2     3     4
```

ساخت بردارهای با عناصر قاعده مند:

فرض کنید بخواهیم برداری تعریف کنیم که عناصر آن شامل اعداد 1 تا 9 باشد که به ترتیب پشت سرهم باشند ، برای این گونه موارد ، نرم افزار متلب شیوه ای از علامت گذاری را به کار می برد که در مثال زیر نشان داده شده است و دیگر لازم نیست که این 9 عدد را به شیوه های ذکر شده قبلی در بردار تعریف کنیم:
مثال :

```
A=1:9
```

نتیجه :

```
A =

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

عدد 1 ، اولین عدد است و آن قدر به آن ، یک واحد یک واحد ، اضافه می شود تا به آخرین عدد که برابر 9 است برسیم .

حال فرض کنید که به جای اعداد 1 تا 9 می بایست اعداد 1 تا 900 را به عنوان عناصر بردار تعریف می کردیم ، مطمئنا نوشتن اعداد 1 تا 900 کار ساده ای نیست اما شیوه فوق این کار را به راحتی انجام می دهد.

نکته :

مراقب باشید که اگر دستوراتی می نویسید که بردارهای با طول بسیار بزرگ ایجاد می کنند ، حتما در انتهای آن خط از دستورات ، از علامت ((;)) استفاده کنید تا مقادیر عناصر بردار در پنجره Commandنمایش داده نشود . نمایش بردارهای با طول بزرگ در پنجره Command می تواند بسیار آزاردهنده باشد و باعث شود که برنامه نویس نتواند به راحتی نتایج خط های مختلف از دستورات را که مفید است در پنجره Command دنبال کند .

برای شیوه ذکر شده ، نوع دیگری از علامت گذاری نیز وجود دارد که در مثال زیر نشان داده شده است:

مثال :

```
A=1:1:9
```

نتیجه :

```
A =

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

تنها تفاوت در این است که میزان افزایش که برابر 1 واحد است را در علامت گذاری ذکر کرده ایم . در واقع دستور A=1:9 نوعی اختصار برای دستور A=1:1:9 در متلب می باشد.

مثال :

فرض کنید بخواهیم اعداد زوج بین 0 تا 10 را به عنوان عناصر یک بردار در متلب تعریف کنیم ، می نویسیم:

```
A=0:2:10
```

نتیجه :

```
A =

     0     2     4     6     8    10
```

اولین عنصر بردار برابر 0 است و هر بار 2 واحد به آن اضافه می شود تا عنصر بعدی بردار ساخته شود تا زمانی که به عدد 10 برسیم که آخرین عنصر بردار می باشد .

میزان افزایش عناصر بردار می تواند یک عدد اعشاری باشد . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم اعداد بین 0 تا 2 را با فاصله 0.2 به عنوان عناصر یک بردار در متلب تعریف کنیم ، می نویسیم:

```
A=1:0.2:2
```

نتیجه :

```
A =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

میزان تغییر منظم عناصر بردار می تواند به صورت کاهشی باشد . برای این منظور باید یک عدد منفی را ، متناسب با میزان کاهش ، مشخص کنیم . به مثال زیر توجه کنید:

مثال :

```
A=9:-1:1
```

نتیجه :

```
A =

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

ماتریس ها در متلب:

یک ماتریس ، آرایه ای مستطیلی از اعداد می باشد .
قبلا بردارها را معرفی کردیم ، باید دقت داشته باشید که یک بردار ستونی در واقع ماتریسی می باشد که تنها دارای یک ستون است و یک بردار ردیفی در واقع ماتریسی می باشد که تنها دارای یک ردیف می باشد.

نحوه تعریف ماتریس ها در متلب:

برای تعریف ماتریس ها در متلب ، چندین روش وجود دارد :
روش اول : تعریف عناصر ماتریس با استفاده از علامت ((،)) برای جدا کردن عناصر و استفاده از علامت ((;)) برای جدا کردن ردیف ها از یکدیگر . به مثال زیر توجه کنید:

مثال :

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

نتیجه :

```
A =

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

روش دوم : تعریف عناصر ماتریس با استفاده از فاصله برای جدا کردن عناصر و استفاده از علامت ((;)) برای جدا کردن ردیف ها از یکدیگر . به مثال زیر توجه کنید:

مثال :

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

نتیجه :

```
A =

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

روش سوم : نرم افزار متلب برای ایجاد برخی ماتریس های خاص دارای دستوراتی می باشد . برخی از این دستورها عبارتند از:

دستور ones :

این دستور ماتریسی ایجاد می کند که تمامی عناصر آن دارای مقدار عددی 1 می باشند . به مثال زیر توجه کنید:

مثال :

ماتریسی می سازیم که دارای 2 ردیف و 3 ستون باشد و تمامی عناصر آن دارای مقدار عددی 1 باشند:

```
A=ones(2,3)
```

نتیجه :

```
A =

     1     1     1
     1     1     1
```

دستور zeros :

این دستور ماتریسی ایجاد می کند که تمامی عناصر آن دارای مقدار عددی 0 می باشند . به مثال زیر توجه کنید:

مثال :

ماتریسی می سازیم که دارای 3 ردیف و 2 ستون باشد و تمامی عناصر آن دارای مقدار عددی 0 باشند:

```
A=zeros(3,2)
```

نتیجه :

```
A =

     0     0
     0     0
     0     0
```

دستور eye (ساخت ماتریس همانی)

ماتریس همانی ، ماتریسی می باشد که عناصر روی قطر اصلی آن برابر 1 و سایر عناصر آن برابر 0 باشد . چنانچه از دستور eye به صورت eye(n) استفاده کنیم آنگاه دستور eye یک ماتریس همانی با nردیف و n ستون را برمی گرداند . به مثال زیر توجه کنید:

مثال :

```
A=eye(4)
```

نتیجه :

```
A =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

نکته :

همچنین با دستور eye می توان ماتریس هایی با تعداد ردیف و تعداد ستون غیر مساوی ساخت که عناصر روی قطر اصلی آنها برابر 1 و سایر عناصر آنها برابر 0 باشد . به مثال زیر توجه کنید:

مثال :

```
A=eye(4,3)
```

نتیجه :

```
A =

     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

ماتریس تصادفی rand:

ماتریس تصادفی با درجه وارد شده میسازد

rand(n,m)

این دستور ماتریس n*m میسازد که درایه های آن تصادفی می باشد

rand(n)

این دستور ماتریسی مربعی میسازد که درایه های آن تصادفی می باشد

دستور randperm در متلب:

چنانچه از دستور randperm به صورت randperm(n) استفاده کنیم انگاه این دستور در خروجی ، یک بردار را برمی گرداند که در آن بردار ، اعداد صحیح 1 تا n به صورت تصادفی قرار گرفته اند . به مثال زیر توجه کنید:

مثال :

```
A=randperm(9)
```

نتیجه :

```
A =

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

دستور inv در متلب:

با دستور inv در متلب ، می توانیم معکوس یک ماتریس را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[10 12 13;4 5 6;7 8 9]
B=inv(A)
```

نتیجه :

```
A =

    10    12    13
     4     5     6
     7     8     9


B =
```

```
   -1.0000   -1.3333    2.3333
    2.0000   -0.3333   -2.6667
   -1.0000    1.3333    0.6667
```

دستور det در متلب:

با استفاده از دستور det در متلب ، می توانیم دترمینان یک ماتریس را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=det(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

  6.6613e-016
```

نکته :

جهت محاسبه دترمینان و معکوس یک ماتریس حتما باید ماتریس مربعی باشد.

محاسبه ترانهاده یک ماتریس در متلب:

ترانهاده یک ماتریس ، ماتریسی می باشد که در آن ، نسبت به ماتریس اولیه ، جای سطرها و ستون ها با هم عوض شده باشد . یعنی عناصر سطر اول به جای عناصر ستون اول و عناصر ستون اول نیز به جای عناصر سطر اول قرار گیرند و عناصر سایر سطرها و ستون ها نیز به همین شکل جایشان با یکدیگر عوض شود . در متلب برای آنکه ترانهاده یک ماتریس را محاسبه کنیم باید پس از نام آن ماتریس ، علامت ' را به کار ببریم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=A'
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

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

انجام عملیات ریاضی بر روی عناصر یک ماتریس:

در برنامه های متلب ، بسیار زیاد پیش می آید که بخواهیم یک عمل ریاضی را بر روی تمامی عناصر یک ماتریس انجام دهیم . نرم افزار متلب برای این موارد از نوعی علامت گذاری استفاده می کند که در مثال زیر بیان شده است:

مثال :

فرض کنیم بخواهیم تمامی عناصر ماتریس A را به توان 2 برسانیم ، می نویسیم:

```
A=[1 2 3 4]
B=A.^2
```

نتیجه :

```
A =

     1     2     3     4


B =

     1     4     9    16
```

نکته :

اهمیت علامت نقطه ((.)) بسیار زیاد است زیرا این علامت است که مشخص می کند که عمل ریاضی مشخص شده بر روی هر عنصر ماتریس صورت گیرد و در صورتی که علامت نقطه گذاشته نشود آن عمل ریاضی بر روی کل ماتریس انجام می شود که مطمئنا نتیجه ای غیر از آنچه می خواستیم به ما خواهد داد .

در صورت نگذاشتن علامت نقطه ، چنانچه آن عمل ریاضی قابل اجرا بر روی کل ماتریس نباشد ، متلب یک پیغام خطا را در خروجی نمایش می دهد زیرا متلب سعی می کند که آن عمل ریاضی را بر

روی کل ماتریس اجرا کند . برای روشن شدن این موضوع به مثال زیر توجه کنید که در واقع همان مثال قبلی بدون علامت نقطه ((.)) می باشد:

مثال :

```
A=[1 2 3 4]
B=A^2
```

نتیجه :

```
A =

     1     2     3     4

??? Error using \ mpower
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
```

پیام خطای فوق به این دلیل است که متلب می خواهد کل ماتریس A را به توان 2 برساند و نمی تواند این کار را انجام دهد ، زیرا تنها اعداد اسکالر و ماتریس های مربعی را می توان به توان 2 رساند .

مثال :

فرض کنید عناصر متناظر دو ماتریس A و B را بخواهیم تک تک در هم ضرب کنیم ، می نویسیم:

```
A=[1 2 3 4]
B=[2 3 4 5]
C=A.*B
```

نتیجه :

```
A =

     1     2     3     4


B =

     2     3     4     5


C =

     2     6    12    20
```

نگذاشتن علامت نقطه ((.)) در دستور فوق باعث می شود که متلب یک پیغام خطا را در خروجی نمایش دهد.

نکته :

برای دو عملگر ریاضی - و + احتیاجی نیست که علامت نقطه گذاشته شود زیرا نرم افزار متلب به طور خودکار این عملگرها را بر روی تک تک عناصر ماتریسها اجرا می کند . به مثال زیر توجه کنید:

مثال :

```
A=[4 4 4 4]
B=[1 2 3 4]
C=A-B
```

نتیجه :

```
A =

     4     4     4     4


B =

     1     2     3     4


C =

     3     2     1     0
```

اشاره به عناصر مختلف یک ماتریس:

اشاره به یک ردیف یا یک ستون ماتریس:

گاهی لازم است که به یک ردیف یا یک ستون یک ماتریس ، اشاره داشته باشیم . برای این منظور می توان از علامت ((:)) استفاده نمود . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=A(2,:)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     4     5     6
```

همان طور که مشاهده می کنید در دستور B=A(2,:) ، عدد 2 برای شماره ردیف و علامت ((:)) برای شماره ستون به کار رفته است یعنی اینکه عناصری مورد نظرمان است که شماره ردیف آنها برابر 2 باشد اما شماره ستون آنها می تواند شماره هر ستونی باشد . بنابراین حاصل برابر تمامی عناصر ردیف دوم ماتریس A می باشد.

اشاره به چند عنصر متوالی از یک ردیف یا یک ستون ماتریس:

ممکن است بخواهیم از یک ردیف یا یک ستون تنها به چند عنصر متوالی آن اشاره کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3 4;5 6 7 8;9 10 11 12]
B=A(2,2:4)
```

نتیجه :

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12


B =

     6     7     8
```

در دستور B=A(2,2:4) ، عدد 2 برای شماره ردیف و عبارت 2:4 برای شماره ستون نوشته شده است . بنابراین B برابر عناصری از ماتریس A خواهد بود که شماره ردیف آنها برابر 2 و شماره ستون آنها از 2 تا 4 می باشد (2:4.)

مثال :

```
A=[1 2 3 4;5 6 7 8;9 10 11 12]
B=A(1:2,2:4)
```

نتیجه :

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12


B =

     2     3     4
     6     7     8
```

اشاره به چند عنصر غیر متوالی از یک ردیف یا یک ستون ماتریس:

گاهی ممکن است عناصر مورد نظرمان متوالی نباشند در اینگونه موارد نمی توانیم از علامت ((:)) استفاده کنیم و باید شماره ردیف یا ستون عناصر مورد نظرمان را درون علامت های] و [قرار بدهیم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=A(2,[1 3])
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     4     6
```

در دستور B=A(2,[1 3]) ، عدد 2 برای شماره ردیف و عبارت [1 3] برای شماره ستون به کار رفته است . عبارت
[1 3]برای شماره ستون به این معنی است که ستون شماره 1 و ستون شماره 3 مورد نظرمان بوده است.

دستور size در متلب:

فرض کنید که ماتریسی داریم که می خواهیم بدانیم اندازه آن چقدر است ، برای این منظور دستور sizeدر متلب مورد استفاده قرار می گیرد . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6]
B=size(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6


B =

     2     3
```

مشاهده می کنید که دستور B=size(A) ، تعداد ردیف ها و تعداد ستون های ماتریس A را محاسبه کرده است و به ترتیب عدد مربوط به آنها را در متغیر B ذخیره کرده است.

محاسبه مینیمم (min) یک ماتریس در متلب:

در متلب برای محاسبه مینیمم (min) یک ماتریس از دستور min استفاده می شود . اگر A یک ماتریس دو بعدی باشد دستور min(A) ، برداری را بر می گرداند که در آن مینیمم هر ستون ماتریس Aمشخص شده است و دستور
min(min(A))، مینیمم ماتریس A را محاسبه می کند . به مثال زیر توجه کنید:

مثال :

```
A=[1 2;3 4]
B=min(A)
C=min(min(A))
```

نتیجه :

```
A =

     1     2
     3     4


B =

     1     2


C =

     1
```

محاسبه ماکزیمم (max) یک ماتریس در متلب:

در متلب برای محاسبه ماکزیمم (max) یک ماتریس از دستور max استفاده می شود . اگر A یک ماتریس دو بعدی باشد دستور max(A) ، برداری را بر می گرداند که در آن ماکزیمم هر ستون ماتریس A مشخص شده است و دستور
max(max (A))، ماکزیمم ماتریس A را محاسبه می کند . به مثال زیر توجه کنید:

مثال :

```
A=[1 2;3 4]
B=max(A)
C=max(max(A))
```

نتیجه :

```
A =

     1     2
     3     4


B =

     3     4


C =

     4
```

دستور length در متلب:

در متلب با استفاده از دستور length می توانیم طول یک بردار (تعداد کل عناصر بردار) را محاسبه کنیم . به مثال زیر توجه کنید:

مثال :

```
A=[7 8 9]
B=length(A)
```

نتیجه :

```
A =

     7     8     9


B =

     3
```

البته دستور length برای ماتریس ها نیز می تواند مورد استفاده قرار بگیرد . اگر دستور length را برای یک ماتریس به کار ببریم ، این دستور تعداد ردیف ها و تعداد ستون های ماتریس را محاسبه می کند و هر کدام از این دو عدد که بزرگتر باشد را در خروجی نمایش خواهد داد . به مثال زیر توجه کنید :

مثال :

```
A=[1 2 3;4 5 6]
B=length(A)
```

نتیجه :

```
A =
```

```
     1     2     3
     4     5     6


B =

     3
```

ساخت ماتریس بالا مثلثی و پایین مثلثی:

در متلب از دستور triu برای ساخت ماتریس بالامثلثی و از دستور tril برای ساخت ماتریس پایین مثلثی استفاده می شود.

ساخت ماتریس بالامثلثی با دستور triu :

یک ماتریس بالامثلثی ، ماتریسی می باشد که عناصری از ان که زیر قطر اصلی قرار دارند ، برابر با 0 باشند . چنانچه از دستور triu برای یک ماتریس استفاده کنیم آنگاه دستور triu آن ماتریس را به یک ماتریس بالا مثلثی تبدیل خواهد کرد . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=triu(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

     1     2     3
     0     5     6
     0     0     9
```

مشاهده می کنید که عناصر روی قطر اصلی و بالای قطر اصلی باقی مانده اند و سایر عناصر ماتریس برابر 0 قرار داده شده اند تا یک ماتریس بالا مثلثی ساخته شود.

ساخت ماتریس پایین مثلثی با دستور tril :

یک ماتریس پایین مثلثی ، ماتریسی می باشد که عناصری از آن که بالای قطر اصلی قرار دارند ، برابر 0 باشند . چنانچه از دستور tril برای یک ماتریس استفاده کنیم ، آنگاه دستور tril آن ماتریس را به یک ماتریس پایین مثلثی تبدیل خواهد کرد . به مثال زیر توجه کنید:

مثال :

```
A=[1 2 3;4 5 6;7 8 9]
B=tril(A)
```

نتیجه :

```
A =

     1     2     3
     4     5     6
     7     8     9


B =

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

مشاهده می کنید که عناصر روی قطر اصلی و عناصر پایین قطر اصلی باقی مانده اند و سایر عناصر ماتریس برابر 0 قرار داده شده اند تا یک ماتریس پایین مثلثی ساخته شود.

اتصال ماتریس ها به یکدیگر در متلب:

در متلب می توانیم دو یا چند ماتریس را به یکدیگر متصل کنیم و یک ماتریس بزرگتر بسازیم . به مثال زیر توجه کنید:

مثال :

```
A=[1 2;3 4]
B=[5 6;7 8]
C=[A B]
```

نتیجه :

```
A =

     1     2
     3     4
```

```
B =

     5     6
     7     8


C =

     1     2     5     6
     3     4     7     8
```

مشاهده می کنید که ماتریس C از الحاق دو ماتریس A و B ساخته شده است.

:Sum

این دستور مجموع سطر ها یا ستون ها را در یک ماتریس محاسبه میکند
نحوه استفاده از این دستور بدین گونه است:

```
sum(A,dim)
```

در صورتی که dim وارد نشود 1 در نظر گرفته میشود.(مجموع درایه های هر ستون نمایش داده میشود)

مثال :

```
s=[1 2 3;4 5 6];
b=sum(s,2)
```

نتیجه :

```
b =

     6
    15
```

دستور بالا مجموع درایه های موجود در سطر را با هم جمع میکند

نکته:

dot ضرب داخلی و cross ضرب خارجی دو بردار را انجام میدهد.

چندجمله ای در MATLAB :

فرم کلی یک چندجمله ای به صورت

$$f(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n$$

می باشد که در MATLAB با بردار سطری $[a_0\ a_1 \ldots\ a_{n-2}\ a_{n-1}\ a_n]$ نمایش داده می شود .

مثال) نمایش چندجمله ای $f(x)=3x^4-0.5x^3+x-5.2$ در MATLAB به صورت زیر می باشد :

```
>> p=[3 -0.5 0 1 -5.2]

p =

   3.0000   -0.5000         0    1.0000   -5.2000
```

دستور polyval :

برای محاسبه مقدار چندجمله ای در یک نقطه خاص از دستورpolyval استفاده می شود.

مثال) مقدار چمدجمله ای فوق را در نقطه x=1 پیدا کنید .

```
>> polyval(p,1)

ans =

  -1.7000
```

دستور roots :

برای پیدا کردن ریشه ها از دستور roots استفاده می شود .

مثال) ریشه های چندجمله ای فوق را پیدا کنید .

```
>> x=roots(p)

x =
```

```
   1.1239
0.1051 + 1.1446i
0.1051 - 1.1446i
-1.1674
```

دستور poly :

دستور poly جند جمله ای متناظر با ریشه های وارد شده را میسازد.

```
q=poly(x)

q =

   1.0000   -0.1667   -0.0000    0.3333   -1.7333
```

همانطور که مشاهده میکنید ضریب بزرگترین توان 1 میباشد که با ضرب کردن ضرایب در 3 همان چند جمله ای x حاصل خواهد شد

دستور conv:

برای ضرب دو چندجمله ای در یکدیگر از تابع conv استفاده می شود .

مثال) برای ضرب دو چندجمله ای $f(x)=3x^3-5x^2+6x-2$ و $g(x)=x^5+3x^4-x^2+2.5$ به صورت زیر عمل می شود .

```
>> a=[3 -5 6 -2];
>> b=[1 3 0 -1 0 2.5];
>> h=conv(a,b)

h =

   3.0000    4.0000   -9.0000   13.0000   -1.0000    1.5000  -
10.5000   15.0000   -5.0000
```

دستور deconv:

برای تقسیم دو چند جمله ای از دستور deconv استفاده میکنیم

به صورت پیش فرض با اجرای دستور فوق خارج قسمت نمایش داده میشود برای مشاهده باقیمانده میتوان به صورت زیر عمل کرد:

مثال :

```
a=[3 -5 6 -2];
b=[1 3 0 -1 0 2.5];
[p,q]=deconv(b,a)
```

deconv(b,a) همان b/a میباشد

نتیجه :

```
p =

    0.3333    1.5556    1.9259


q =

         0         0         0   -0.0370   -8.4444    6.3519
```

که p خارج قسمت و q باقیمانده تقسیم b بر a میباشد.

تعریف متغیرها در متلب به صورت سمبلیک:

وقتی می گوییم در متلب یک متغیر به صورت سمبلیک تعریف شود ، منظور این است که عدد خاصی را به آن متغیر نسبت نمی دهیم و تنها با نماد آن سر و کار داریم . در درس ریاضی بسیاری از معادلات به همین صورت حل می شوند و رایجترین نمادها x و y می باشند.

دستور syms :

دستور syms در متلب ، برای تعریف متغیرها به صورت سمبلیک به کار می رود . به عنوان مثال اگر بخواهیم دو متغیر x و y را به صورت سمبلیک تعریف کنیم باید اینگونه بنویسیم:

```
syms x y
```

حال می توانیم معادلاتی را به صورت سمبلیک بر حسب دو متغیر x و y بنویسیم .

با مثال زیر این موضوع را بهتر توضیح می دهیم:

مثال :

```
syms x y
(x+y)*(x+y)^5
```

نتیجه :

```
ans =

(x + y)^6
```

نکته :

فرض کنید x و y را به صورت سمبلیک تعریف نکنیم و تنها دستور زیر را اجرا نماییم (چنانچه قبلا آن دو را در متلب تعریف کرده باشیم باید ابتدا دستور clear all را اجرا کنیم:)

```
(x+y)*(x+y)^5
```

در این صورت با پیغام خطای زیر مواجه می شویم:

```
??? Undefined function or variable 'x'.
```

این پیغام خطا به این دلیل است که نرم افزار متلب ، به طور پیش فرض برای متغیر x و y مقدار می خواهد مگر آنکه قبلا این دو متغیر به صورت سمبلیک تعریف شده باشند.

حل معادلات خطی :

روش ماتریس:

بسیاری از معادلات خطی قابل نمایش به فرم ماتریسهای ساده می باشند و می توان با قوانین مربوط به ماتریسها به راحتی اقدام به حل اینگونه معادلات نمود . برای مثال دستگاه زیر را در نظر بگیرید :

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 366 \\ 4x_1 + 5x_2 + 6x_3 = 804 \\ 7x_1 + 8x_2 + 0x_3 = 351 \end{cases}$$

دستگاه فوق را می توان به فرم زیر نیز نوشت :

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 366 \\ 804 \\ 351 \end{bmatrix}$$

برای حل اینگونه معادلات هیچ محدودیتی وجود ندارد البته در حالت کلی اگر دترمینان ماتریس A صفر باشد دستگاه جواب ندارد . در زیر برنامه حل مثالی که در بالا بیان گردید مشاهده می شود .

```
>> A=[1 2 3;4 5 6;7 8 0];
>> B=[366;804;351];
>> x=inv(A)*B

x =

25.0000
22.0000
99.0000
```

حل دستگاه چند معادله و چند مجهول با دستورsolve :

دستور solve در متلب برای حل معادلات به کار می رود و چنانچه بخواهیم یک دستگاه چند معادله و چند مجهول را حل کنیم باید معادلات را به دستور solve بدهیم تا این دستور ، پاسخ دستگاه را محاسبه کند . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم دستگاه دو معادله و دو مجهول $\begin{cases} x+2y=7 \\ x-y=1 \end{cases}$ را حل کنیم . برای این منظور ، دو معادله را درون پرانتز دستور solve می نویسیم . هر معادله باید درون دو علامت ' قرار بگیرد:

```
clear all
close all
clc

S=solve('x+2*y=7','x-y=1')
S=[S.x S.y]
```

دستور S=solve('x+2*y=7','x-y=1') ، دستگاه دو معادله و دو مجهول را حل می کند و عبارت S=[S.x S.y] نیز برای نمایش نتایج به دست آمده از حل دستگاه دو معادله و دو مجهول است .

نتیجه :

```
S =

    x: [1x1 sym]
    y: [1x1 sym]


S =

[ 3, 2]
```

مثال :

فرض کنید بخواهیم دستگاه سه معادله و سه مجهول $\begin{cases} x+y+z=6 \\ x-y^2+2z=7 \\ x+y-z=0 \end{cases}$ را حل کنیم ، می نویسیم:

```
clear all
close all
clc

S=solve('x+y+z=6','x-y^2+2*z=7','x+y-z=0')
S=[S.x S.y S.z]
```

نتیجه :

```
S =

    x: [2x1 sym]
    y: [2x1 sym]
    z: [2x1 sym]


S =

[ 2,  1, 3]
[ 5, -2, 3]
```

مشاهده می کنید که دستگاه سه معادله و سه مجهول فوق ، دارای دو سری پاسخ می باشد.

مشتق گیری از عبارت های سمبلیک با دستور diff :

معمولا در ریاضیات بسیار پیش می آید که بخواهیم مشتق عبارتی را محاسبه نماییم . در متلب برای آنکه بتوانیم از عبارتی مشتق بگیریم ابتدا باید متغیرهای به کار رفته در آن عبارت را با دستور syms به صورت سمبلیک تعریف کنیم . سپس با دستور diff می توانیم مشتق آن عبارت را محاسبه نماییم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم مشتق عبارت x^4 را محاسبه کنیم . می نویسیم:

```
syms x
diff(x^4)
```

نتیجه :

```
ans =

4*x^3
```

مشاهده می کنید که مشتق عبارت x^4 در خروجی نمایش داده شده است . دستور syms x باعث می شود که در متلب ، متغیر x به صورت سمبلیک تعریف شود.

نکته :

می توانیم ابتدا با دستور inline تابعی را به صورت f(x) تعریف کنیم و سپس از این تابع برحسب متغیر x مشتق بگیریم . به مثال زیر توجه کنید:

مثال :

```
syms x
f=inline('x^4','x')
diff(f(x))
```

نتیجه :

```
f =

     Inline function:
     f(x) = x^4


ans =

4*x^3
```

مشتق مرتبه دوم و مرتبه های بالاتر با دستور diff :

برای گرفتن مشتق مرتبه دوم و یا مرتبه های بالاتر باید در دستور diff مرتبه مشتق را مشخص کنیم . به مثال زیر توجه کنید:

مثال :

```
syms x
diff(x^4,2)
```

نتیجه :

```
ans =

12*x^2
```

مشاهده می کنید که مشتق مرتبه دوم عبارت x^4 در خروجی نمایش داده شده است . مرتبه دوم بودن مشتق را با نوشتن عدد 2 در درون پرانتز دستور diff مشخص کرده ایم.

نکته :

اگر بخواهیم از عبارت x^4 مشتق مرتبه n ام بگیریم باید دستور diff(x^4,n) را اجرا کنیم.

حل معادله دیفرانسیلی در متلب با دستور dsolve :

دستور dsolve برای حل معادله دیفرانسیلی در متلب به کار می رود . فرض کنید y تابعی از متغیر x باشد . معادله دیفرانسیلی شامل مشتق مرتبه اول و یا مرتبه های بالاتر از y خواهد بود . اما چگونه باید y' را برای دستور dsolve مشخص کنیم ؟ روش مورد استفاده این است که به جای y' از نماد D استفاده کنیم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم معادله دیفرانسیلی $xy'+1=y$ را حل کنیم . می نویسیم:

```
dsolve('x*Dy+1=y','x')
```

نتیجه :

```
ans =

C2*x + 1
```

مشاهده می کنید که پاسخ معادله دیفرانسیلی در خروجی نمایش داده شده است.

حل معادلات با شرایط مرزی :

کافی است شرایط مرزی را در تابع dsolve وارد نماییم .

```
>> syms x y
>> y=dsolve('Dy=y*x','y(1)=1','x')

y =

1/exp(1/2)*exp(1/2*x^2)
```

برای آشنایی بیشتر به حل مثال دیگری می پردازیم :

$$y' = 1 + x + y^2 + xy^2$$

```
>> syms x y
>> y=dsolve('Dy=1+x+y^2+x*y^2','x')

y =

tan(x+1/2*x^2+C1)
```

بعد از حل معادله ، برای نمایش بهتر y ، می توان دستور pretty را اجرا نمود .

حل معادلات دیفرانسیل مرتبه دوم و بالاتر :

همان طور که گفتیم مشتق مرتبه اول y' را با نماد Dy برای دستور dsolve مشخص می کنیم اما اگر مشتق مرتبه دوم و یا بالاتر باشد آنگاه باید ابتدا نماد D را نوشته ، سپس عدد مربوط به مرتبه مشتق را بنویسیم و در آخر نیز نماد y نوشته شود . مثلا برای تعریف y'' باید نماد D2y و برای تعریف y'''باید نماد D3y را به کار ببریم

معادلات مرتبه دوم :

در این حالت نیز همانند معادلات مرتبه اول از تابع dsolve استفاده می کنیم . به عنوان مثال ، معادله زیر را به کمک MATLAB حل می کنیم .

$$y'' - y' - 6y = 0$$

```
>> syms x y
>> y=dsolve('D2y-Dy-6*y=0','x')

y =

C1*exp(-2*x)+C2*exp(3*x)
```

.

دستگاه معادلات دیفرانسیل خطی

در این حالت باز هم از دستور dsolve استفاده می شود ، با این تفاوت که تمامی معادلات را وارد دستور می نماییم .

$$\begin{cases} y_1' = 2y_1 - 5y_2 \\ y_2' = 5y_1 - 6y_2 \end{cases}$$

```
>> syms y1 y2
>> [y1,y2]=dsolve('Dy1=2*y1-5*y2','Dy2=5*y1-6*y2')

y1 =

1/5*exp(-2*t)*(4*C1*sin(3*t)+3*C1*cos(3*t)+4*C2*cos(3*t)-3*C2*sin(3*t))

y2 =

exp(-2*t)*(C1*sin(3*t)+C2*cos(3*t))
```

به عنوان مثالی دیگر ، دستگاه معادلات زیر را حل می کنیم .

$$\begin{cases} \dot{x} - 2x - 3y = 2e^{2t} \\ -x + \dot{y} - 4y = 3e^{2t} \end{cases}$$

```
>> syms x y
>> [x y]=dsolve('Dx-2*x-3*y=2*exp(2*t)','-x+Dy-4*y=3*exp(2*t)')

x =

-3*exp(t)*C2+exp(5*t)*C1-5/3*exp(2*t)


y =

exp(t)*C2+exp(5*t)*C1-2/3*exp(2*t)
```

انتگرال گیری در متلب با دستور int :

معمولا انتگرال گیری به دو شیوه صورت می گیرد . در شیوه اول ، حدود بالا و پایین انتگرال مشخص نیست و ما تنها به صورت سمبلیک انتگرال را محاسبه می کنیم . در شیوه دوم حدود بالا و پایین انتگرال مشخص می باشد و با توجه به حد بالا و حد پایین ، مقدار انتگرال محاسبه می شود و نتیجه به صورت عدد در خروجی نمایش داده می شود . در متلب دستور int به هر دو شیوه ذکر شده می تواند انتگرال را محاسبه نماید . در ادامه هر دو شیوه را شرح خواهیم داد.

انتگرال نامعین:

زمانی که حد بالا و حد پایین انتگرال مشخص نباشد باید تنها عبارت زیر انتگرال و همچنین متغیری که باید بر حسب آن انتگرال گرفته می شود را برای دستور int مشخص کنیم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم انتگرال $\int 4x^3 dx$ را محاسبه نماییم . می نویسیم:

```
syms x
int(4*x^3,x)
```

نتیجه :

```
ans =

x^4
```

نکته :

دقت شود مثال بالا را به صورت int('4*x^3','x') نیز می توان نوشت اما استفاده از دستور syms توصیه می شود زیرا این شیوه اجرای دستور int بدون دستور syms در آینده از نرم افزار متلب حذف خواهد شد.

انتگرال معین:

در صورتی که حد بالا و حد پایین انتگرال مشخص باشند باید این دو حد را در دستور int بنویسیم . به مثال زیر توجه کنید:

مثال :

```
syms x
int(4*x^3,0,1)
```

نتیجه :

```
ans =

1
```

محاسبه انتگرال های چندگانه:

برای محاسبه انتگرال های چندگانه باید از دستور int به صورت تو در تو استفاده کنیم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم انتگرال $\int_0^{x}\int_0^{\sin x}\left(x^2+y^2\right)dydx$ را محاسبه کنیم . می نویسیم:

```
syms x y
int(int(x^2+y^2,y,0,sin(x)),0,pi)
```

نتیجه :

```
ans =

pi^2 - 32/9
```

محاسبه حد در متلب با دستور limit :

در متلب برای محاسبه حد از دستور limit استفاده می شود . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم حد $\lim_{x\to 0}\left(\frac{\sin x}{x}\right)$ را محاسبه کنیم . می نویسیم:

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

نتیجه :

```
ans =

1
```

در دستور limit(sin(x)/x,x,0) ، نماد x را نوشته ایم تا مشخص کنیم که حد برای متغیر x می باشد و عدد 0 نیز همان عددی است که باید متغیر x به سمت آن میل کند.

محاسبه حد راست یا حد چپ با دستور limit :

برای این که با دستور limit حد راست یا حد چپ یک عبارت را محاسبه نماییم باید درون پرانتز دستور limit، عبارت 'right' یا 'left' را بنویسیم . به مثال زیر توجه کنید:

مثال :

حد $\lim_{x\to 0^-}\left(\frac{|x|}{x}\right)$ که به صورت حد چپ می باشد را محاسبه می کنیم:

```
syms x
limit(abs(x)/x,x,0,'left')
```

نتیجه :

```
ans =

-1
```

محاسبه حد بی نهایت:

برای محاسبه حد بی نهایت تنها کافی است از نماد Inf استفاده کنیم که در متلب برای نمایش بی نهایت به کار می رود . به مثال زیر توجه کنید:

مثال :

$$\lim \frac{\left(x^3-x^2+x\right)}{\left(5x^3-3\right)}$$

فرض کنید می خواهیم حد $x\to\infty$ را محاسبه کنیم . می نویسیم:

```
syms x
limit((x^3-x^2+x)/(5*x^3-3),x,Inf)
```

نتیجه :

```
ans =

1/5
```

محاسبه لاپلاس و عکس لاپلاس:

دستور laplace :

دستور laplace در متلب برای محاسبه تبدیل لاپلاس به کار می رود . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم از تابع $f(t)=t^2+t$ تبدیل لاپلاس بگیریم . کدهای زیر را می نویسیم:

```
syms t
f=t^2+t;
laplace(f)
```

ابتدا با استفاده از دستور syms t ، متغیر t را به صورت سمبلیک تعریف کرده ایم . سپس f بر حسب tنوشته شده است . در آخر نیز دستور laplace(f) تبدیل لاپلاس f را محاسبه خواهد کرد.

نتیجه :

```
ans =

1/s^2 + 2/s^3
```

دستور ilaplace :

دستور ilaplace برای محاسبه معکوس تبدیل لاپلاس در متلب به کار می رود . به مثال زیر توجه کنید :

مثال :

این بار از تبدیل لاپلاس مثال قبل ، تبدیل لاپلاس معکوس می گیریم:

```
syms s
f=1/s^2 + 2/s^3;
ilaplace(f)
```

نتیجه :

```
ans =

t^2 + t
```

مشاهده می کنید که دقیقا همان تابع تعریف شده در مثال قبل می باشد.

m-fileها در متلب:

چنانچه بخواهید برنامه ای طولانی و پیچیده بنویسید ، دیگر پنجره Command جوابگوی نیاز شما نیست و به محیطی فراتر از آن برای نوشتن دستورات و تصحیح کردن آنها نیاز دارید . متلب برای این گونه موارد امکان ساخت m-file ها را فراهم کرده است . شما می توانید در یک m-file تمامی دستورات خود را نوشته و تنها بر روی یک دکمه گرافیکی کلیک کرده و سپس نتیجه اجرای دستورات را در پنجره Command ببینید.

ساخت یک m-file در متلب:

برای ساخت یک m-file جدید می توانید از هر یک از روش های زیر استفاده کنید :

-1در بالای پنجره اصلی نرم افزار متلب بر روی گزینه New script کلیک کنید . این گزینه به شکل  می باشد .

-2با نگه داشتن کلید Ctrl و فشار دادن کلید N از کیبورد ، این کار را انجام دهید .

-3در پنجره Command بنویسید edit و سپس کلید enter از کیبورد را فشار بدهید .

هر یک از روش های بالا را که انتخاب کنید ، نتیجه این است که متلب یک پنجره خالی باز می کند که می توانید در آن دستورات خود را اجرا کنید .

توصیه می شود اولین دستوری که در یک m-file می نویسید ، دستور clear all باشد تا تمامی متغیرهایی که قبلا در متلب تعریف شده است را پاک کند و اختلالی در روند اجرای برنامه ایجاد

نشود .
باید دقت داشته باشید که در نرم افزار متلب ، m-fileها برای دو هدف اصلی به کار می روند ، کاربرد اول آن نوشتن برنامه های پیچیده وطولانی و کاربرد دوم آن ساخت تابع می باشد . ساخت تابع با استفاده از m-file را در مباحث بعدی توضیح خواهیم داد . در این مبحث تنها در مورد نوشتن برنامه در m-file ها صحبت خواهیم کرد .
پس از آنکه دستورات برنامه را در m-file نوشتیم ابتدا باید با استفاده از گزینه Save در بالای همان پنجره m-file ، آن را ذخیره کنیم . همچنین با نگه داشتن کلید Ctrl و فشار دادن کلید S ، می توانید این کار را انجام دهید .
سپس برای اجرای برنامه باید بر روی گزینه Save and run که به شکل  می باشد کلیک کنید تا نتایج برنامه در پنجره Command نمایش داده شود . همانطور که از نام این گزینه مشخص است ، این گزینه عمل ذخیره کردن را هم انجام می دهد یعنی اگر تغییراتی در برنامه ایجاد کنید و سپس بر روی این گزینه کلیک کنید ، این تغییرات در m-file ذخیره می شود . اگر قبلا فایل ذخیره نشده باشد ابتدا از شما می خواهد که نامی برای آن انتخاب کرده و سپس آن را ذخیره کنید .
m-fileها دارای پسوند m می باشند (به عنوان مثال. program.m) :

نوشتن توضیحات در m-file :

زمانی که یک برنامه طولانی بنویسید ، به دلیل حجم زیاد دستورات ، ممکن است بخشی از روند برنامه نویسی را فراموش کنید . گذشت زمان نیز بسیار تاثیرگذار است و گاهی آن قدر از زمان نوشتن برنامه گذشته است که خود برنامه نویس مجبور می شود برنامه را بارها بخواند تا درک کند که از چه روش هایی استفاده کرده است و گاهی نوشتن یک برنامه جدید به صرفه تر است و زمان کمتری نیاز دارد . بر حسب تجربه ثابت شده است که با استفاده از 2 تکنیک زیر می توان این مشکل را تا حد زیادی برطرف کرد .
1-انتخاب هوشمندانه نام متغیرها به گونه ای که هدف استفاده از آنها را بتوان از نامشان به طور کامل درک کرد .
2-می توانیم هنگام نوشتن برنامه ، توضیحاتی را در کنار کدها بنویسیم تا با خواندن آنها خود برنامه نویس یا هر شخص دیگری به راحتی درک کند که روش های استفاده شده در برنامه چیست.
در متلب چنانچه از علامت درصد ((%)) استفاده کنیم ، تمامی نوشته های بعد از علامت درصد به صورت توضیح در نظر گرفته می شوند . به مثال زیر توجه کنید:
مثال :

```
x=2
% ali eskandari
y=3
```

نتیجه :

```
x =
```

```
     2


y =

     3
```

همان طور که مشاهده می کنید ، ali eskandari به عنوان دستور در نظر گرفته نشده و در خروجی نیز نمایش داده نشده است.

باید دقت داشته باشید که متلب نوشته های بعد از علامت درصد را تنها در خط فعلی به صورت توضیح در نظر می گیرد و نوشته های خط بعد را به صورت دستور (نه توضیح) در نظر می گیرد .

بنابراین چنانچه بخواهیم توضیحاتی را در چند خط پشت سرهم بنویسیم باید در ابتدای هر کدام از آن خط ها از علامت درصد استفاده کنیم . به مثال زیر توجه کنید:

مثال :

```
x=2
% ali eskandari
% this is a simple code
y=3
```

نتیجه :

```
x =

     2


y =

     3
```

نکته :

برای اجرای برنامه از  یا F5 استفاده میکنیم.

ساخت حلقه در متلب با for :

گاهی اوقات تعدادی دستور داریم که باید به دفعات زیاد اجرا شوند اگر بخواهیم به صورت معمولی آنها را بنویسیم مجبور می شویم کدهای مربوط به آنها را به دفعات زیاد تکرار کنیم اما انتخاب مناسب برای اجرای دستورات تکراری ساخت یک حلقه می باشد . در متلب ساده ترین روش برای ساخت حلقه استفاده از for می باشد . در مثال زیر نحوه استفاده از for را برای ساخت یک حلقه شرح داده ایم:

مثال :

فرض کنید بخواهیم حاصل 10!=1*2*3*4*5*6*7*8*9*10 را با نرم افزار متلب محاسبه کنیم برای این منظور کدهای زیر را می نویسیم:

```
k=1;
for m=2:10
    k=k*m;
end
k
```

نتیجه :

```
k =

     3628800
```

در کدهای فوق مشخص کرده ایم که مقدار m از 2 تا 10 باید باشد و در هر بار اجرای دستورات حلقه ، mبرابر مقدار یکی از اعداد این بازه خواهد بود (2 و 3 و ... و 9 و 10) . ابتدا مقدار k را قبل از شروع حلقه برابر 1 تعریف کرده ایم . سپس حلقه for شروع می شود . ابتدا مقدار m برابر 2 که اولین عدد است قرار داده می شود ، kدر m=2 ضرب می شود و چون دستورات حلقه تمام شده است و مقدار بعدی m=3 در نظر گرفته می شود و این بار مقدار جدید k در m=3 ضرب می شود و همین طور این روند ادامه می یابد تا زمانی که m=10 نیز در مقدار جدید k ضرب شود و چون دیگر مقدار جدیدی برای m وجود ندارد حلقه پایان می یابد و در آخر مقدار k نمایش داده می شود.

اجرای دستورات شرطی با دستور if در متلب:

از دستور if در متلب برای اجرای دستورات شرطی استفاده می شود . یعنی اینکه در ابتدا شرط یا شرط هایی توسط متلب چک می شود و اگر آن شرط یا شرط ها برآورده شده باشد آنگاه متلب دستورات مشخص شده را اجرا خواهد کرد . به مثال زیر توجه کنید:

مثال :

```
A=5
if A>=0
    B=A
end
if A<=0
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

همان طور که مشاهده می کنید از دو دستور if استفاده کرده ایم . هدف این است که مقدار B برابر قدرمطلق A باشد ، بنابراین اگر A مساوی یا بزرگتر از صفر باشد باید B را برابر A قرار دهیم و اگر A مساوی یا کوچکتر از صفر باشد باید B را برابر –A قرار دهیم.
دقت کنید که در پایان دستور if حتما باید end نوشته شود تا نرم افزار متلب بداند که دستور if پایان یافته است.

دستور if به همراه else :

همان طور که گفتیم زمانی که از دستور if در متلب استفاده می کنیم ، متلب شرط یا شرط هایی را چک می کند و در صورت برآورده شدن آنها ، دستورات را اجرا می کند . اما شاید بخواهیم به متلب اعلام کنیم که اگر شرط یا شرط ها برآورده نشدند آنگاه چه دستوراتی را اجرا کند . در اینگونه موارد دستور if را با else به کار می بریم . به مثال زیر توجه کنید:

مثال :

در مثال قبلی از دو دستور if استفاده کردیم اما این بار همان مثال را تنها با یک دستور if می نویسیم:

```
A=5
if A>=0
    B=A
else
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

هدف این بوده است که B برابر قدرمطلق A باشد ، ابتدا متلب چک می کند که A مساوی یا بزرگتر از صفر هست یا نه ، اگر باشد آنگاه B را برابر A قرار می دهد و چون شرط برآورده شده است دستورات نوشته شده برای else را نادیده می گیرد . اما اگر A مساوی یا بزرگتر از صفر نباشد آنگاه متلب تنها دستورات مربوط به else را اجرا می کند.

دستور if به همراه elseif :

گاهی نیاز داریم که چندین شرط به صورت پی در پی چک شوند ، اگر اولین شرط صحیح بود دستورات مربوط به ان اجرا شوند و دستورات مربوط به سایر شرط ها نادیده گرفته شوند ، اما اگر شرط اول برآورده نشده بود شرط دوم چک شود و در صورت برآورده شدن شرط دوم ، دستورات مربوط به آن اجرا شود و دستورات مربوط به شرط های باقیمانده نادیده گرفته شود ، در صورت برآورده نشدن شرط دوم آنگاه شرط سوم چک شود و همین طور تا آخر . در اینگونه موارد باید از دستور if به همراه elseif استفاده کنیم . به مثال زیر توجه کنید:

مثال :

همان مثال قبل را این بار با استفاده از elseif می نویسیم . تنها تفاوت این است که حالت خاص A=0را جداگانه بررسی کرده ایم:

```
A=5
if A>0
    B=A
elseif A==0
    B=0
else
    B=-A
end
```

نتیجه :

```
A =

     5


B =

     5
```

دقت شود که برای چک کردن شرط تساوی حتما باید از دو علامت تساوی به صورت == استفاده شود ، زیرا علامت = در متلب برای نسبت دادن مقدار به متغیرها در نظر گرفته شده است و بنابراین برای چک کردن شرط تساوی مجبوریم از علامت == استفاده کنیم.

دستور while:

در این دستور دقیقا مانند if شرطی وارد میشود ولی نحوه کنترل برنامه بدین صورت است که تا زمانی که شرط صادق باشد دستورات داخل عبارت while تکرار خواهند شد.

مثال :

همان مثال قبل را این بار با استفاده از elseif می نویسیم . تنها تفاوت این است که حالت خاص A=0را جداگانه بررسی کرده ایم:

```
clc
close all
clear all
x=.5
while (sin(x)<.8)
    x=x+.1;
end
x
sin (x)
```

نتیجه :

```
x =

    1.0000


ans =

    0.8415
```

ترسیم گرافیکی توابع در متلب:

دستور ezplot :

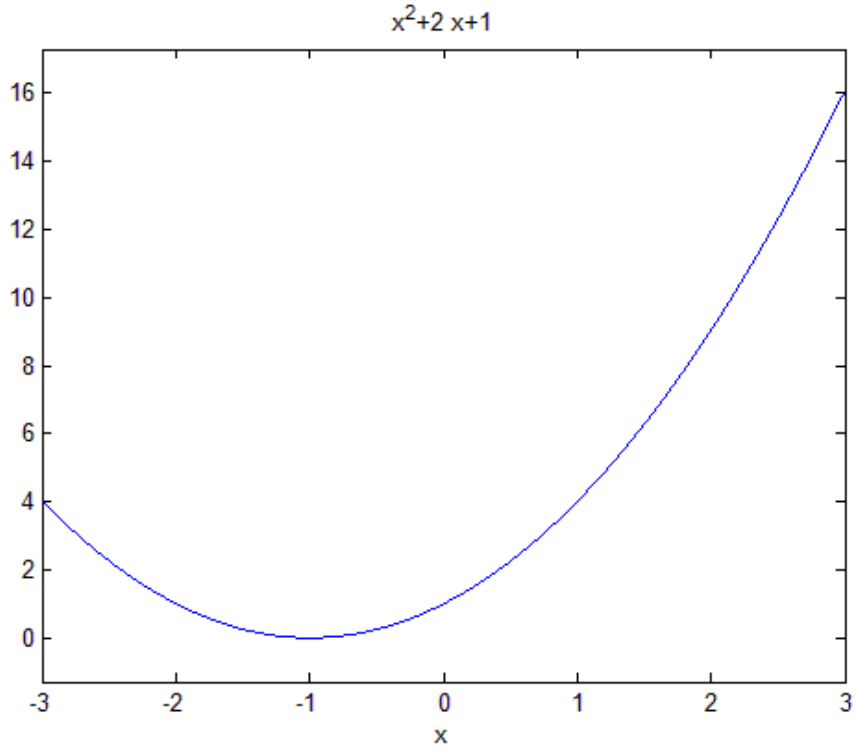
در متلب با استفاده از دستور ezplot می توانیم توابع را به صورت گرافیکی رسم کنیم . تنها کافی است که ابتدا عبارت تابع و سپس محدوده ای که می خواهیم تابع در آن محدوده رسم شود را مشخص کنیم . به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2+2*x+1',[-3,3])
```

نتیجه :

متلب یک پنجره جدید را باز کرده و نتیجه را به صورت یک شکل نمایش می دهد:



عبارت [-3,3] ، محدوده ای که می خواهیم تابع رسم شود را مشخص کرده است.

دقت شود که برای تعریف عبارت مربوط به تابع از علامت ' استفاده کردیم ، در صورتی که بخواهیم از این علامت استفاده نکنیم باید قبل از استفاده از دستور ezplot ، متغیر x را برای متلب به صورت سمبلیک تعریف کنیم . به مثال زیر توجه کنید:

مثال :

```
syms x
ezplot(x^2+2*x+1,[-3,3])
```

نتیجه :

نتیجه دقیقا همان شکل مثال قبل می باشد.

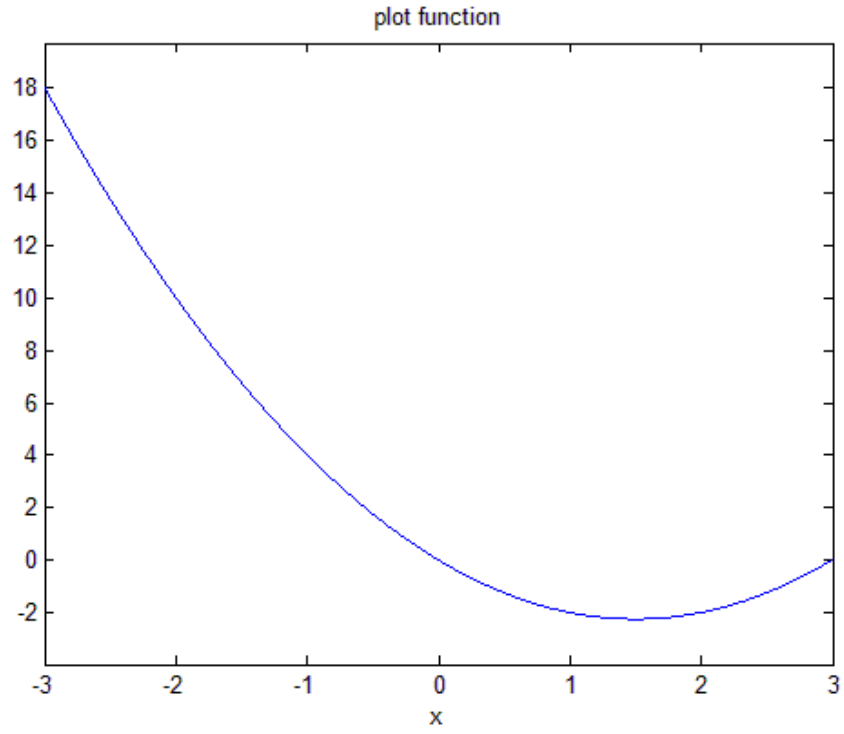
مشخص کردن عنوان برای شکل خروجی:

چنانچه بخواهیم که شکل ترسیم شده توسط دستور ezplot و یا هر روش دیگر) دارای عنوان خاصی باشد ، باید در خط بعدی پس از دستور ezplot از دستور title استفاده کنیم . این دستور یک رشته (مشخص شده با علامت ') را دریافت کرده و به صورت عنوان در بالای شکل نمایش می دهد . به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2-3*x',[-3,3])
title 'plot function'
```

نتیجه :



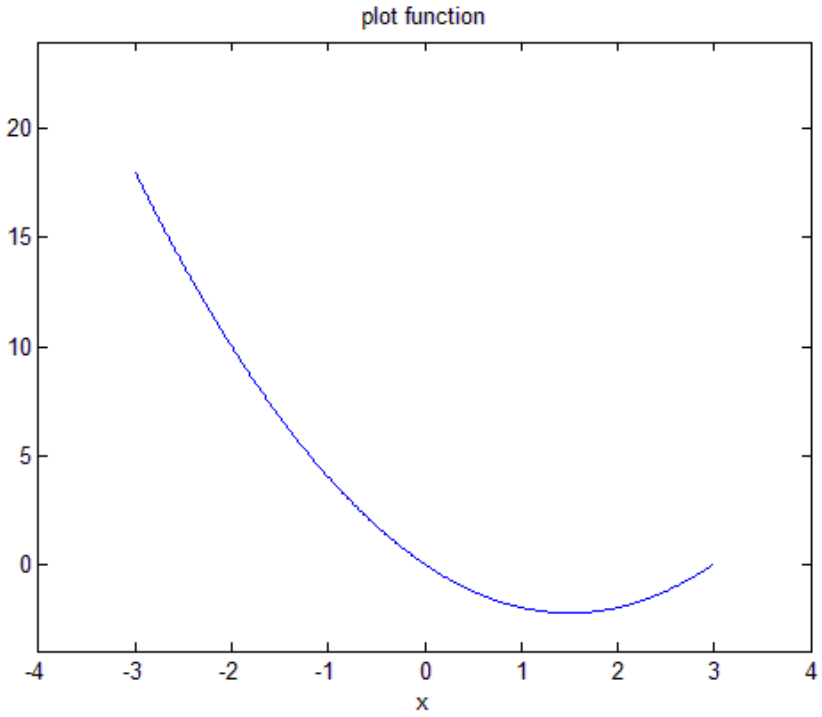
مشاهده می کنید که عبارت plot function در بالای شکل نمایش داده شده است.

فرض کنید بخواهیم حدود محورهای افقی و عمودی را تغییر بدهیم ، برای این کار باید از دستور axis استفاده کنیم . به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2-3*x',[-3,3])
title 'plot function'
axis([-4 4 -4 24])
```

نتیجه :



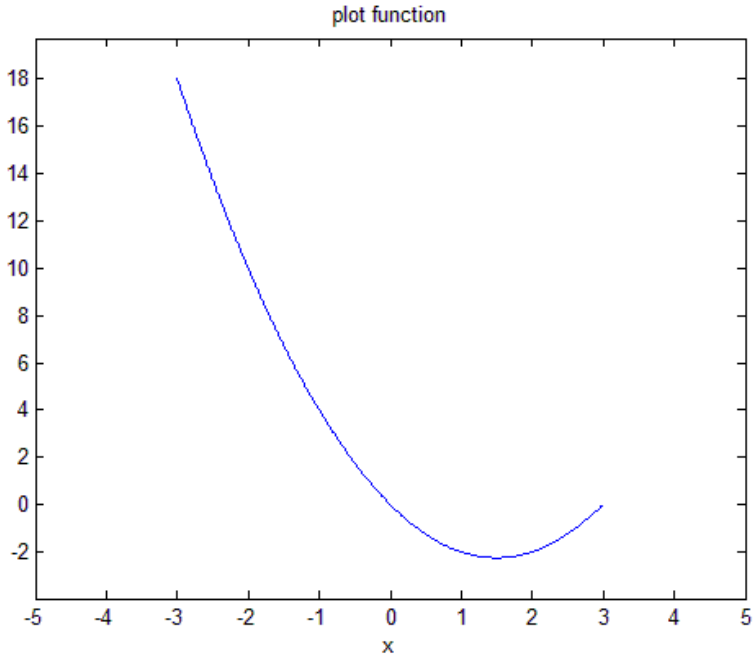
در دستور axis ، درون براکت ، چهار عدد نوشته شده است که دو تای اول حدود محور افقی و دو تای دوم حدود محور عمودی را مشخص می کند.

چنانچه تمایل داشته باشیم که تنها حدود یکی از محورها را تغییر دهیم می توانیم از دستور axis استفاده کنیم ، به این صورت که باید هر چهار عدد نوشته شود و تنها دو عدد مربوط به محوری که می خواهیم حدودش تغییر کند را باید تغییر دهیم . اما این کار را با دستورهای xlim و ylim نیز می توانیم انجام دهیم . به مثال زیر توجه کنید:

مثال :

```
ezplot('x^2-3*x',[-3,3])
title 'plot function'
xlim([-5 5])
```

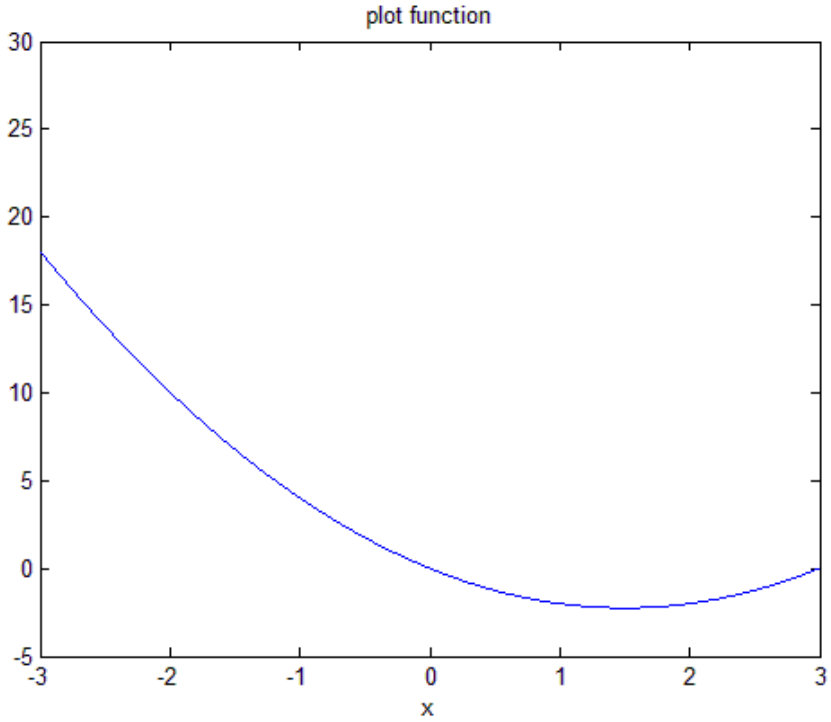
نتیجه :



مثال :

```
ezplot('x^2-3*x',[-3,3])
title 'plot function'
ylim([-5 30])
```

نتیجه :



برای بستن پنجره ای که شکل را نمایش می دهد می توانید به روش های زیر عمل کنید :

1-در پنجره Command ، کلمه close را تایپ کرده و کلید enter را فشار دهید (اجرای دستور close) .

2-بر روی دکمه close در بالای خود پنجره نمایش دهنده شکل ، کلیک کنید .

3-از منوی file ، گزینه close را انتخاب کنید.

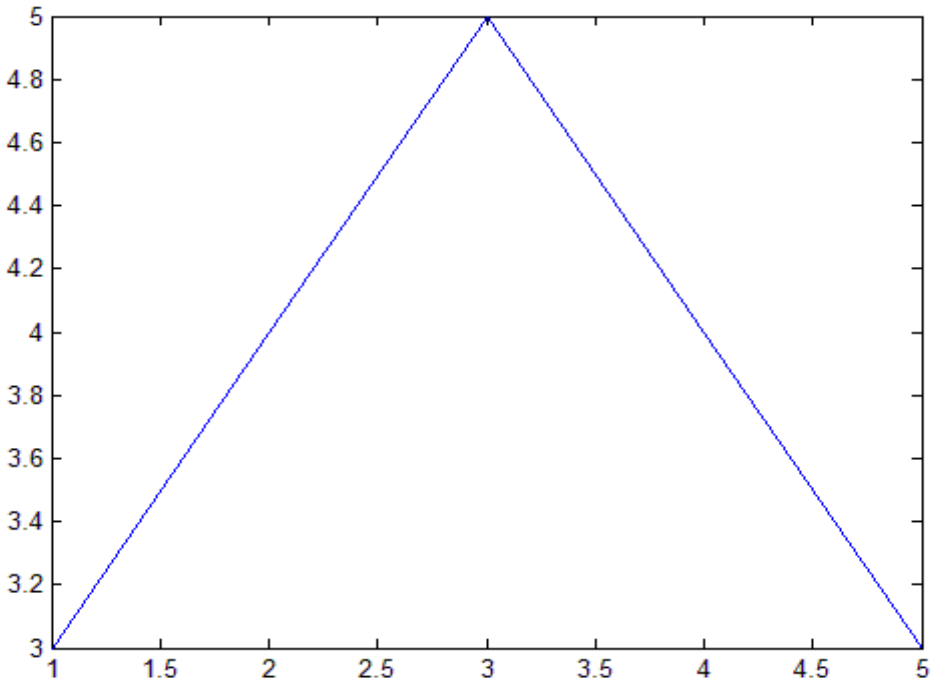
دستور plot :

دستور plot بردارهایی از اعداد را دریافت کرده و آنها را به صورت شکل ترسیم می کند . در واقع دستور plot مقادیر گسسته را که هر کدام به صورت یک نقطه می باشند پشت سرهم قرار می دهد و سپس آنها را با خط به هم وصل می کند تا بتوانیم آنها را به صورت یک شکل پیوسته ببینیم و بدین ترتیب به ارتباط کلی آنها پی ببریم . به مثال زیر توجه کنید:

مثال :

```
x=[1 2 3 4 5]
y=[3 4 5 4 3]
plot(x,y)
```

نتیجه :

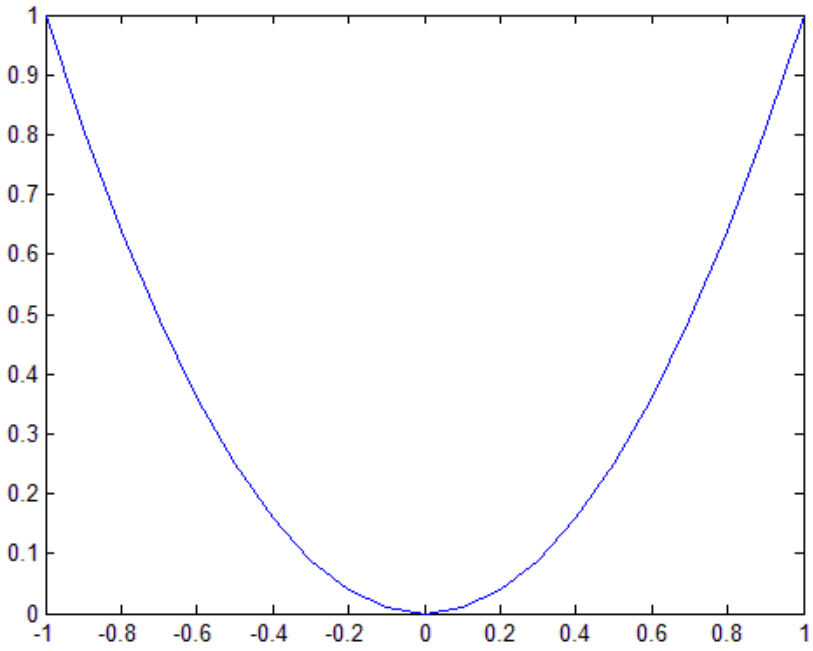


حال فرض کنید بخواهیم تابعی بر حسب متغیر x را برای بازه ای از تغییرات x رسم کنیم ، ابتدا باید متغیر x را به صورت برداری از نقاط آن بازه تعریف کنیم . بازه مورد نظر ما پیوسته است و شامل تعداد بینهایت عدد می باشد اما ما باید تعداد نقاط را به گونه ای انتخاب کنیم که حداقل تعدادی باشند که شکل تابع را به خوبی نمایش بدهند . به مثال زیر توجه کنید:

مثال :

```
x=-1:0.1:1;
plot(x,x.^2-2.*x)
```

نتیجه :



در دستور فوق دقت کنید که در تعریف تابع ، پس از x یک علامت نقطه ((.)) نوشته شده است . وجود این علامت ضروری است و مشخص می کند که هر عنصر بردار x باید به توان 2 برسد نه این که کل بردار x به توان 2 برسد.

دستور xlabel و دستور ylabel :

همان طور که می دانید در نرم افزار متلب ، دستورات مختلفی همچون plot و ezplot برای ترسیم شکل و منحنی ها به کار می روند . این دستورات به محورهای افقی و عمودی یا عنوان اختصاص نمی دهند و یا اگر عنوان اختصاص بدهند ممکن است آن عنوان مد نظر ما نبوده باشد . ممکن است بخواهیم در شکل ترسیم شده توسط این دستورات ، محورهای افقی و عمودی دارای عنوان خاصی باشند . برای این منظور در متلب از دو دستور xlabel و ylabel استفاده می شود.

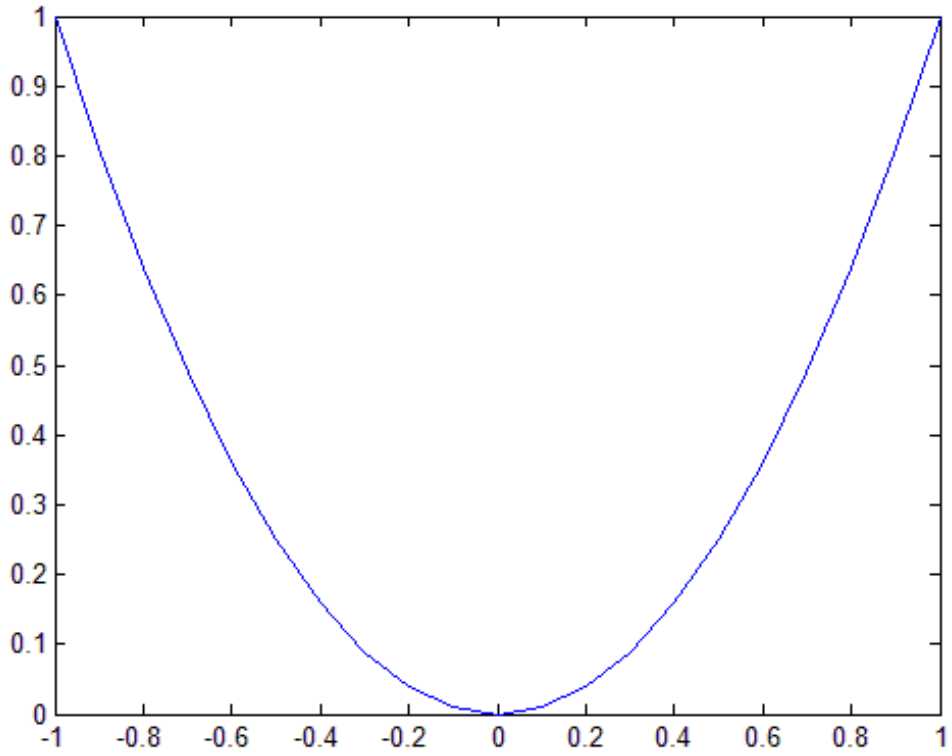
از دستور xlabel در متلب برای تعیین یک عنوان برای محور افقی شکل استفاده می شود و همچنین دستور ylabel نیز برای تعیین یک عنوان برای محور عمودی شکل به کار می رود . برای آشنایی با نحوه استفاده از دو دستور xlabel و ylabel به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم تابع $y = x^2$ را در بازه [-1,1] با دستور plot رسم کنیم . می نویسیم:

```
x=-1:0.1:1;
plot(x,x.^2)
```

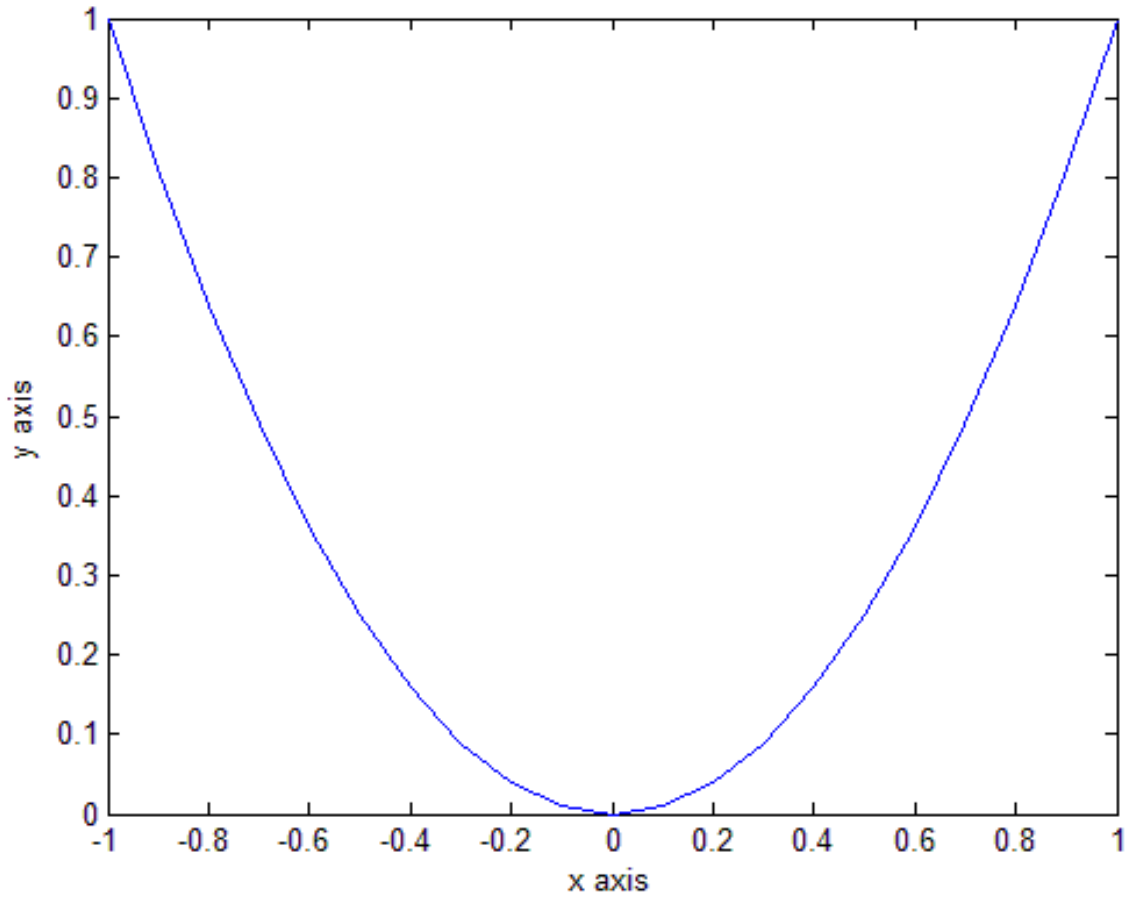
نتیجه :



مشاهده می کنید که شکل خروجی دارای عنوان برای محور افقی و عنوان برای محور عمودی نمی باشد . حال این بار با دستور xlabel و دستور ylabel برای هر دو محور شکل ، عنوان تعیین می کنیم :

```
x=-1:0.1:1;
plot(x,x.^2)
xlabel('x axis')
ylabel('y axis')
```

نتیجه :



مشاهده می کنید که دستور xlabel('x axis') تعیین کرده است که محور افقی دارای عنوان x axis باشد و دستور ylabel('y axis') نیز تعیین کرده است که محور عمودی دارای عنوان y axis باشد.

تعیین رنگ خطوط منحنی های رسم شده با دستور plot در متلب:

در متلب با استفاده از دستور plot می توانیم منحنی های مختلف را رسم کنیم . اما دستور plot ، خطوط منحنی را با یک رنگ پیش فرض نمایش می دهند . چنانچه بخواهیم خطوط منحنی با رنگی دیگر نمایش داده بشوند باید عبارت مربوط به آن رنگ را درون پرانتز این دستورات بنویسیم . برای تعیین رنگ باید در میان دو علامت ' یک حرف انگلیسی را که نشان دهنده آن رنگ می باشد بنویسیم.

در متلب برای هر رنگ یک حرف انگلیسی در نظر گرفته شده است . لیست این حروف در جدول زیر نمایش داده شده است:

رنگ	حرف متناظر برای آن رنگ
Red	r
Green	g
Blue	b
Cyan	c
Magenta	m
Yellow	y
Black	k
White	w

اگرچه در تعدادی از مباحث اشاراتی به شیوه های مختلف نمایش منحنی ها در متلب داشتیم اما در این مبحث قصد داریم تمامی حالت ها را به طور کامل توضیح دهیم و برای هر یک نیز مثالی نمایش بدهیم . یک تابع را در نظر گرفته و تمامی شیوه ها را برای نمایش آن به کار خواهیم برد.

شیوه های مختلف نمایش خطوط منحنی برای دستور plot در متلب:

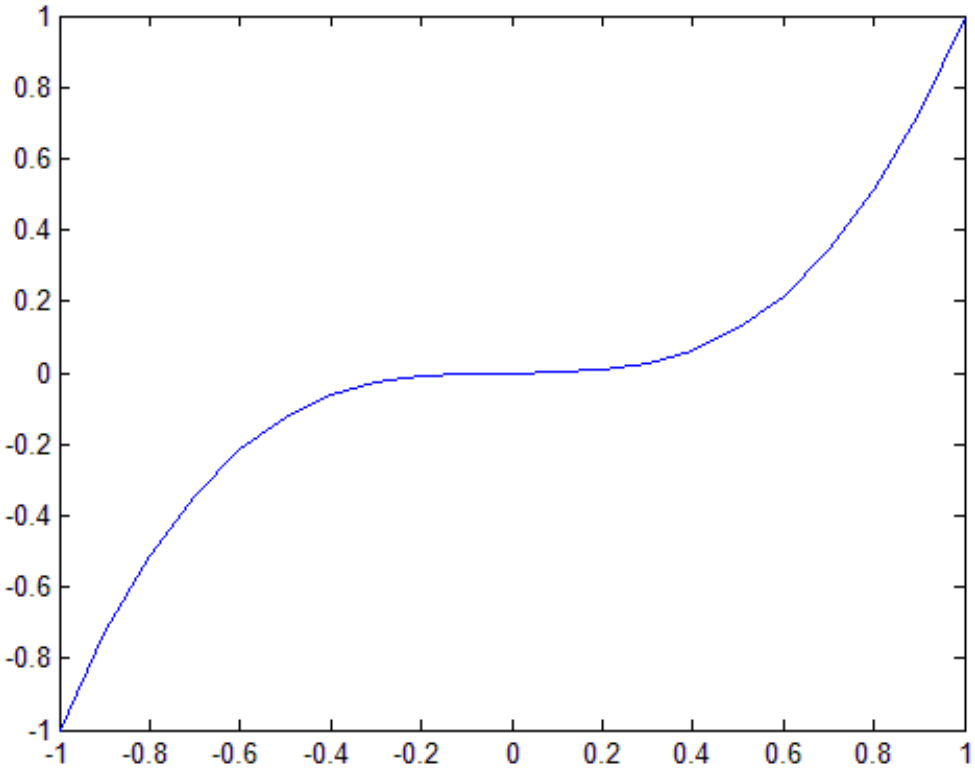
همان طور که می دانید دستور plot در متلب برای ترسیم منحنی ها به کار می رود . این دستور برای نحوه نمایش خطوط منحنی ها دارای یک پیش فرض می باشد اما می توان این پیش فرض را تغییر داد تا خطوط منحنی ها با اشکال و شیوه های دیگری نمایش داده شوند . برای این کار به دستور جدیدی نیاز نیست و تنها باید درون پرانتز دستور plot ، عبارت مربوط به شیوه نمایش مورد نظرمان را بنویسیم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم تابع $y = x^3$ را در بازه [-1,1] با دستور plot رسم کنیم . می نویسیم:

```
x=-1:0.1:1;
plot(x,x.^3)
```

نتیجه :



این شیوه نمایش منحنی ، شیوه نمایش پیش فرض برای دستور plot می باشد . حال شیوه های دیگر را به کار می بریم:

شیوه های نمایش خطوط منحنی (Line Style Specifiers) :

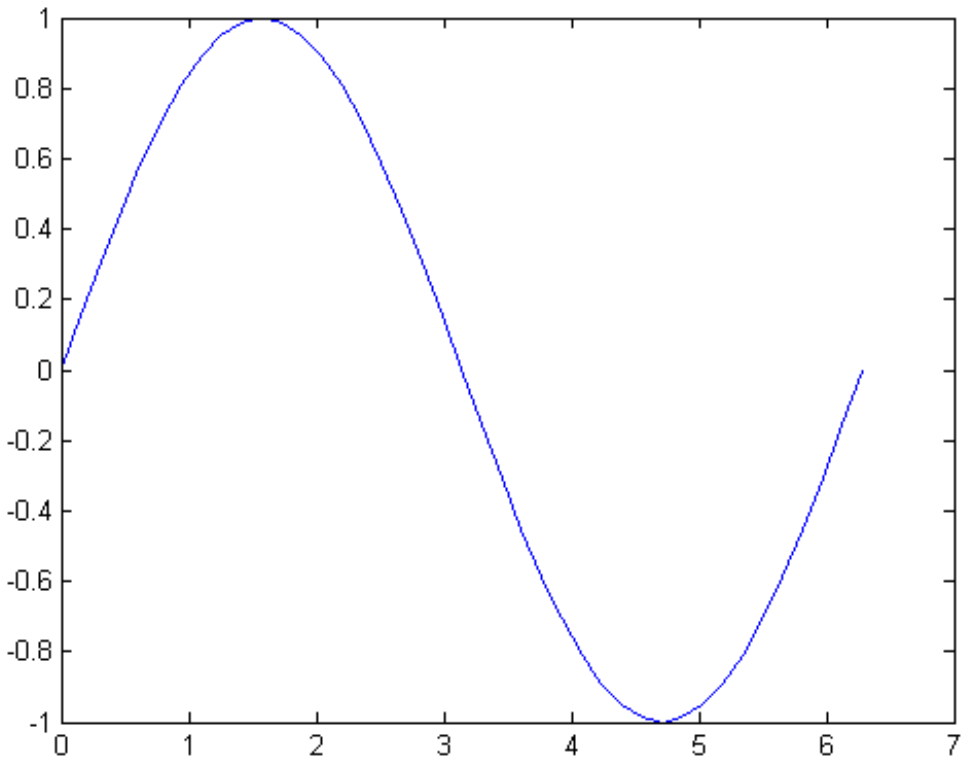
شیوه های نمایش خطوط منحنی (Line Style Specifiers) در جدول زیر خلاصه شده است:

Specifier	Line Style
'-'	Solid line (default)
'--'	Dashed line
':'	Dotted line
'-.'	Dash-dot line
'none'	No line

مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'-')
```

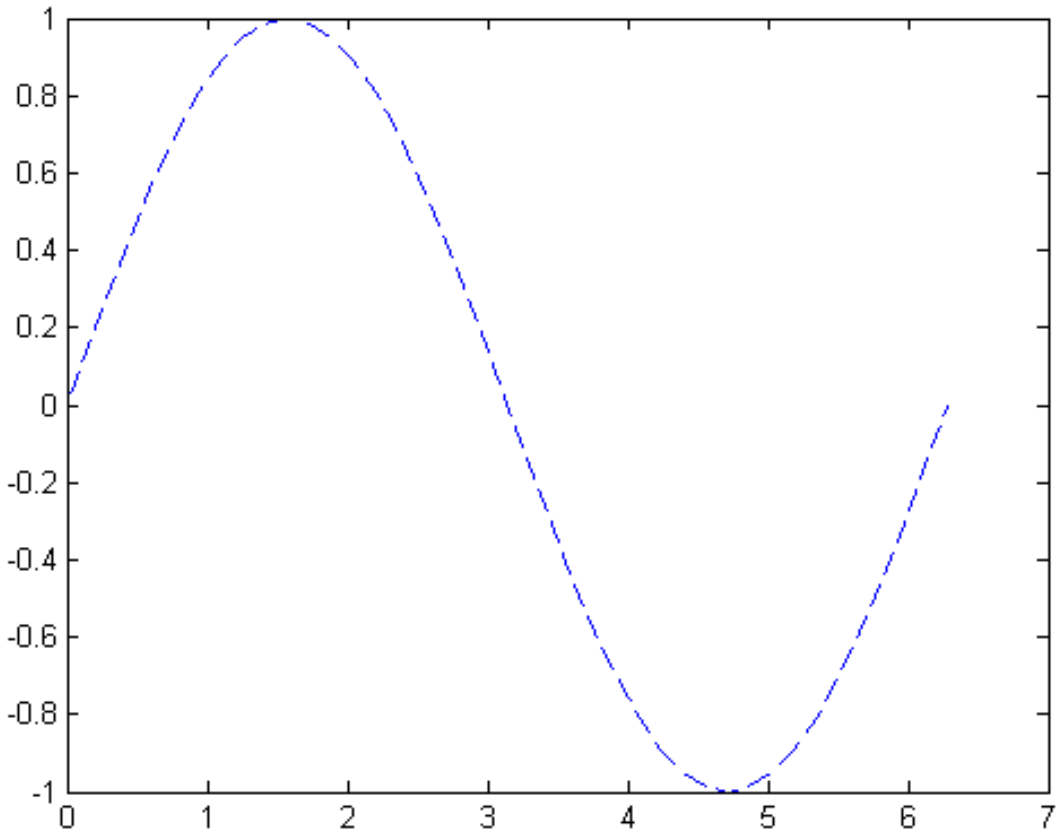
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'--')
```

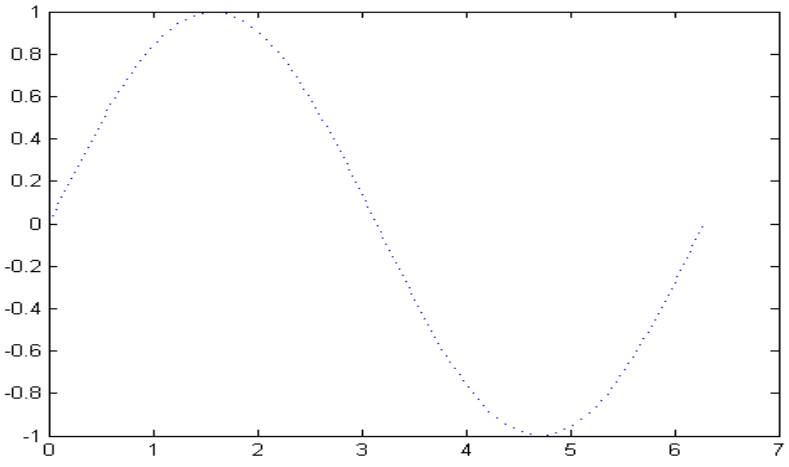
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),':')
```

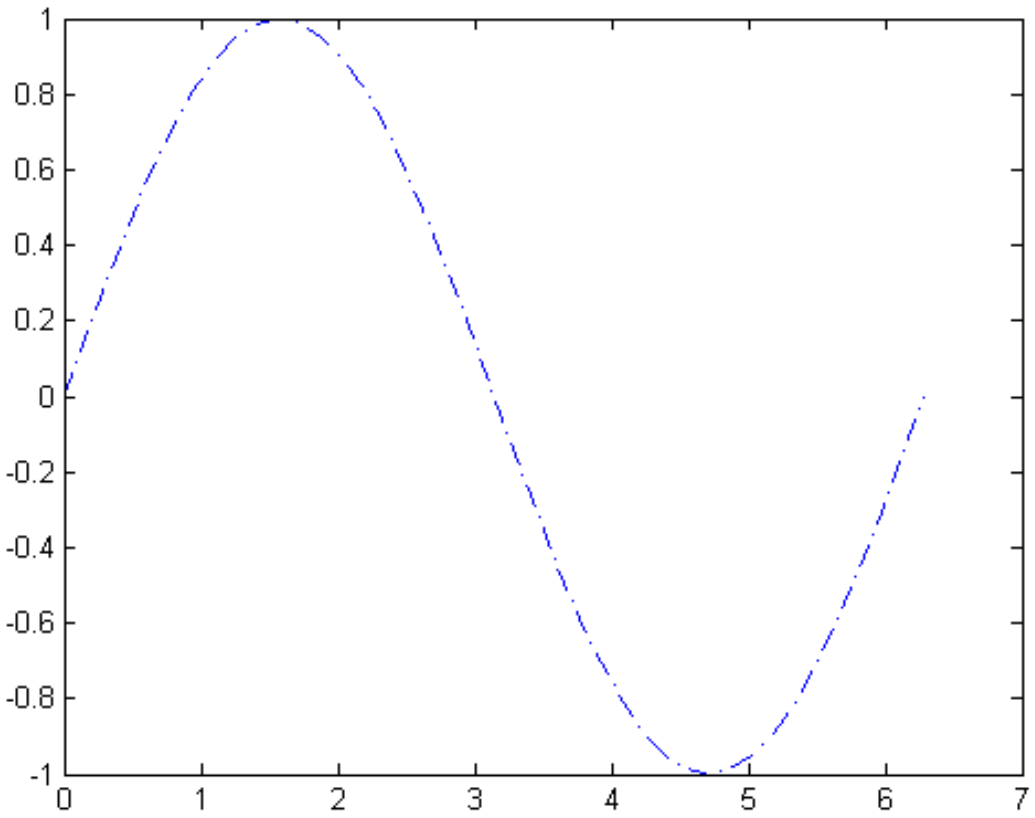
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'-.')
```

نتیجه :



شیوه های نمایش نقاط (Marker Specifiers) :

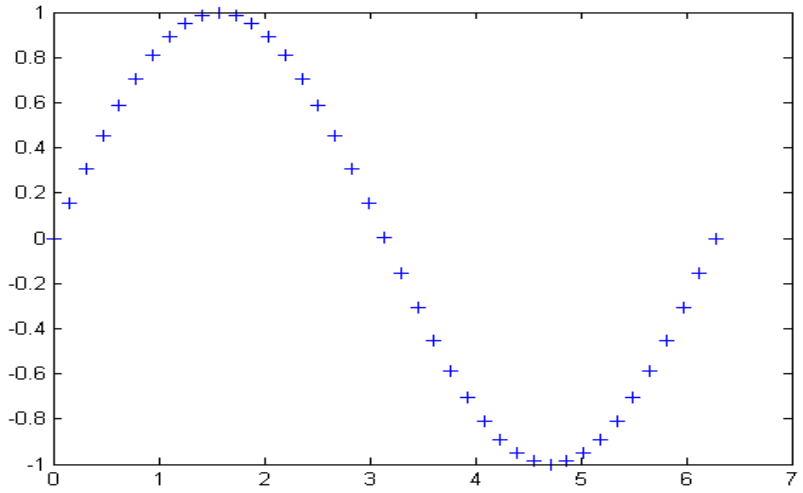
شیوه های نمایش نقاط (Marker Specifiers) در جدول زیر خلاصه شده است:

Specifier	Marker Type
'+'	Plus sign
'o'	Circle
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h'	Six-pointed star (hexagram)
'none'	No marker (default)

مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'+')
```

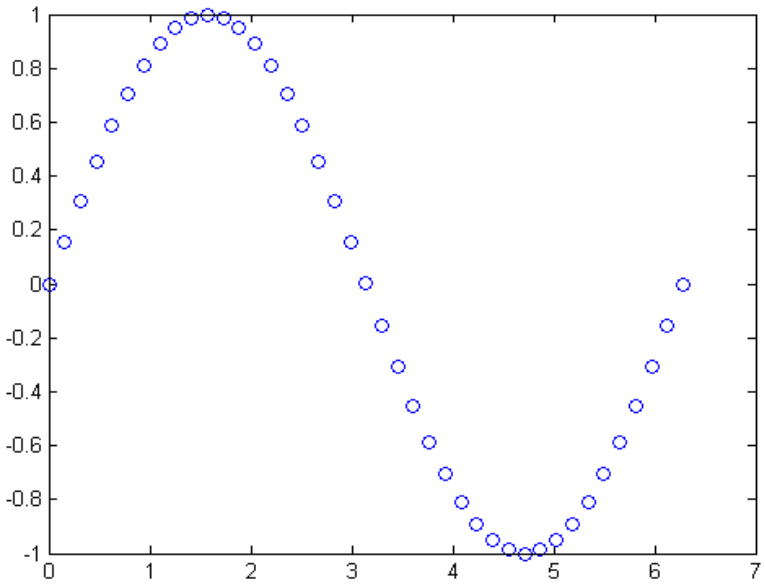
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'o')
```

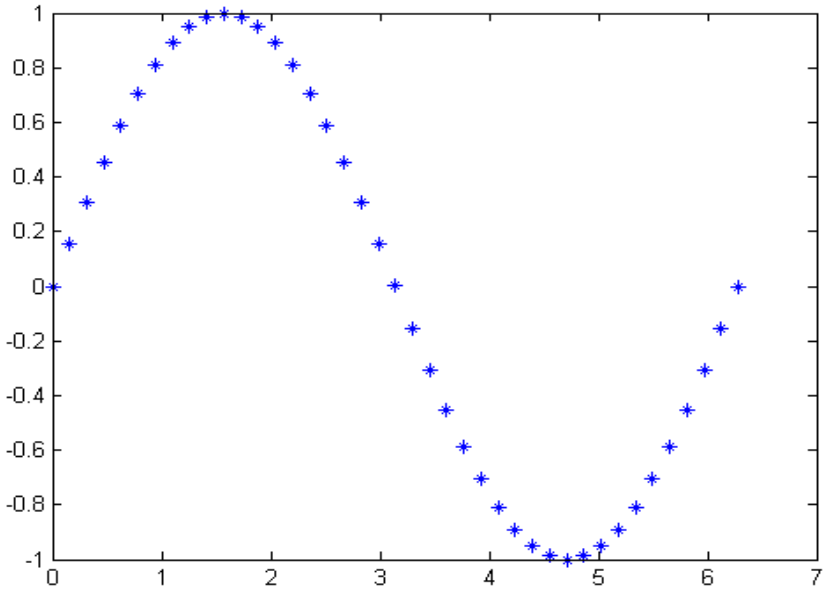
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'*')
```

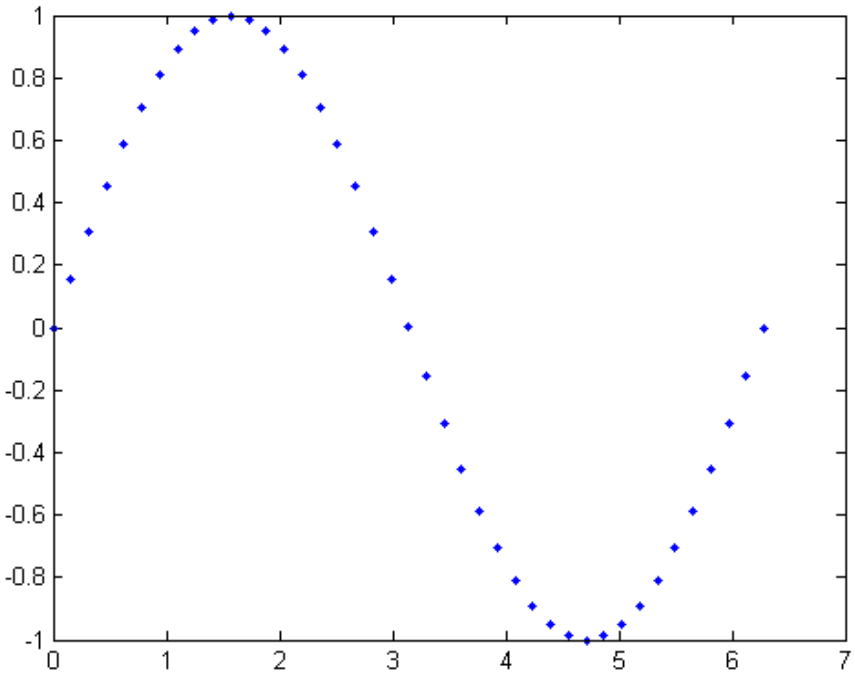
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'.')
```

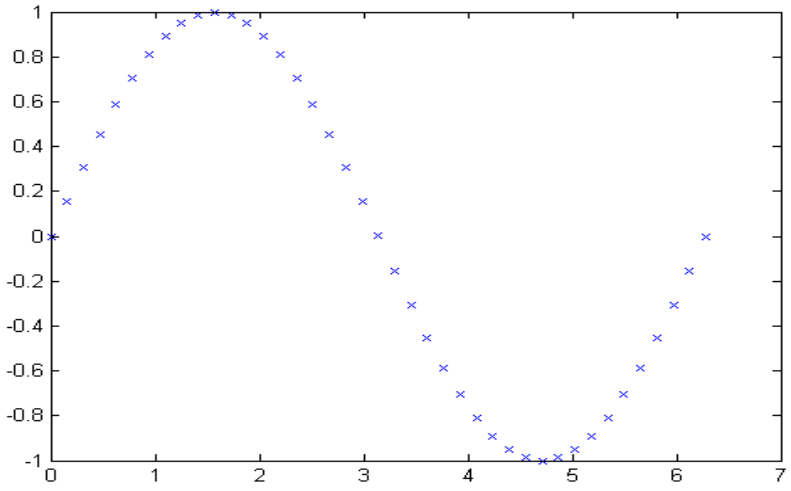
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'x')
```

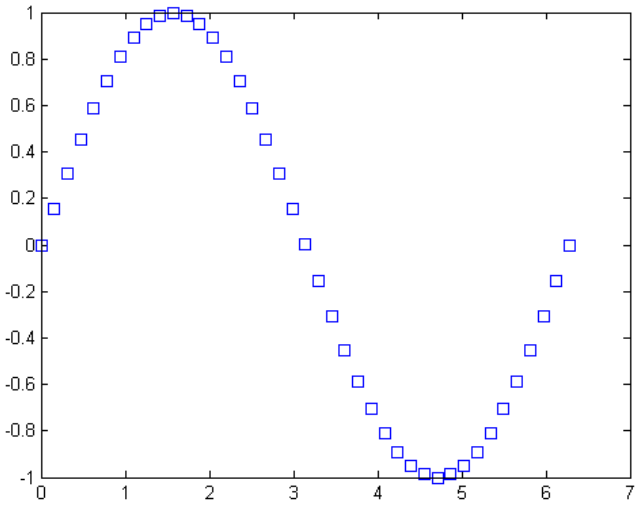
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'square')
```

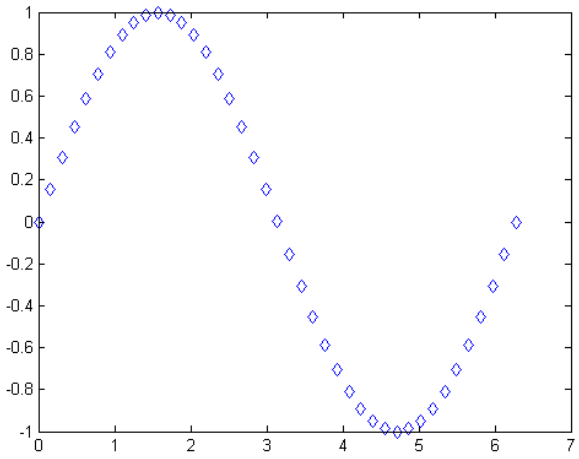
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'diamond')
```

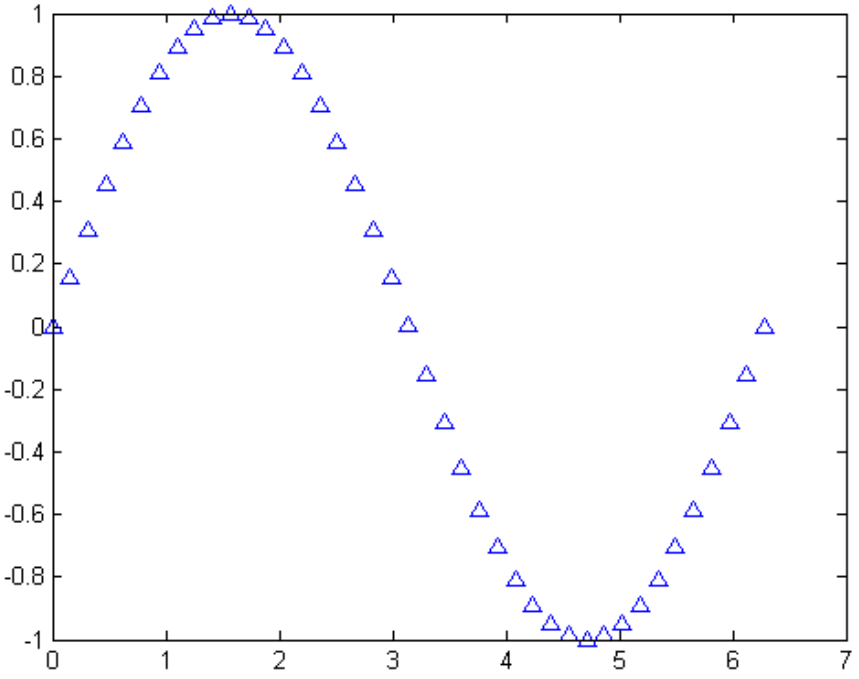
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'^')
```

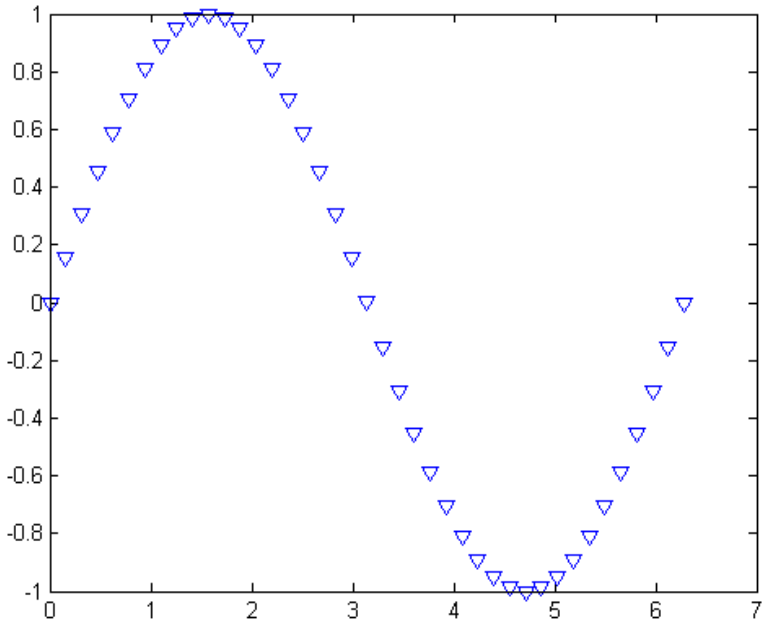
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'v')
```

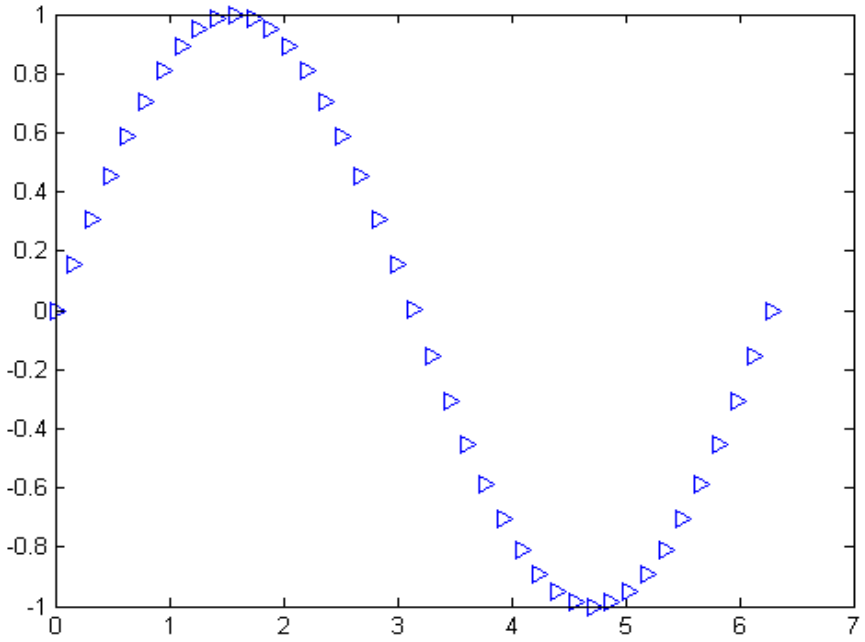
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'>')
```

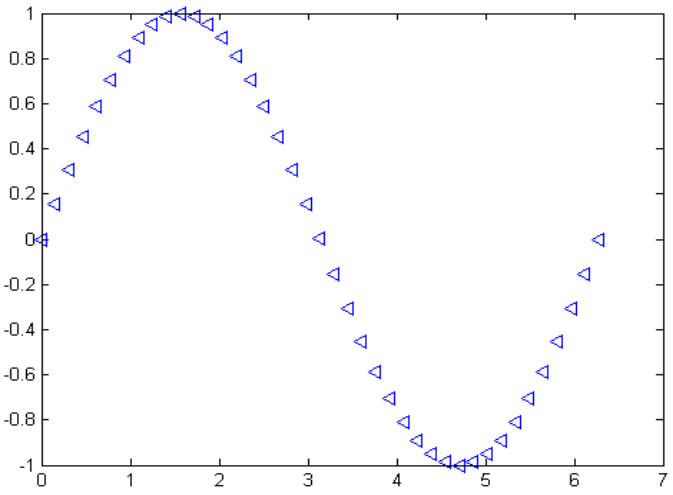
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'<')
```

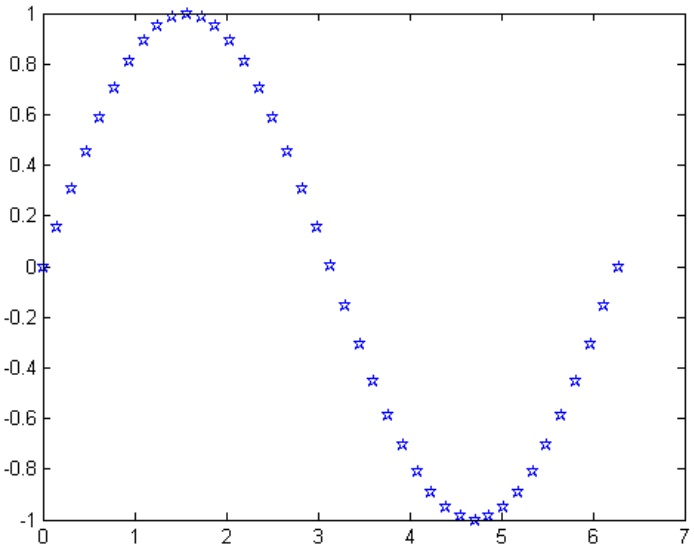
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'pentagram')
```

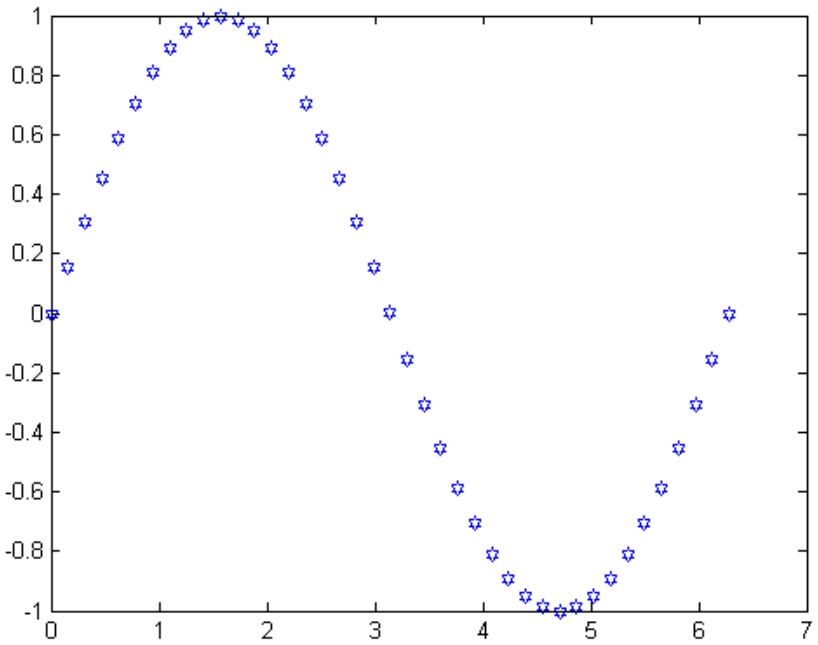
نتیجه :



مثال :

```
t=0:pi/20:2*pi;
plot(t,sin(t),'hexagram')
```

نتیجه :



نمایش پس زمینه شکل به صورت چهارخانه با دستور grid در متلب:

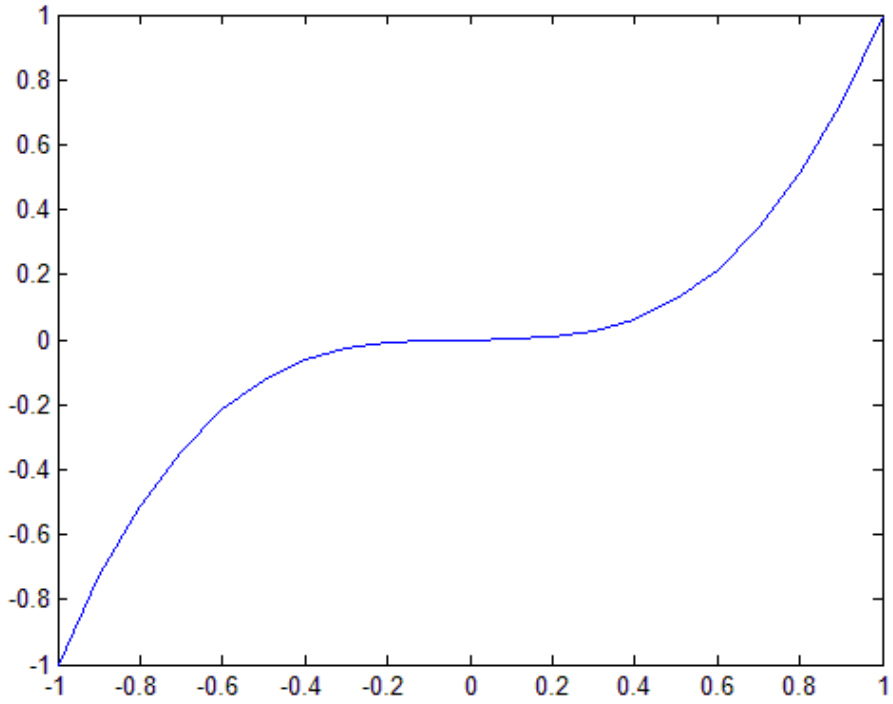
با دستور grid در متلب می توانیم پس زمینه یک شکل را به صورت چهارخانه درآوریم . مثلا فرض کنید با دستور plot یک منحنی را رسم کرده باشیم . چنانچه با دستور grid ، پس زمینه منحنی را به صورت چهارخانه درآوریم ، در این صورت راحت تر می توانیم مقادیر متناظر با هر نقطه از منحنی را تشخیص بدهیم . به مثال زیر توجه کنید:

مثال :

فرض کنید بخواهیم $y = x^3$ را در بازه [-1,1] با استفاده از دستور plot رسم کنیم . ابتدا این منحنی را به صورت معمولی و بدون استفاده از دستور grid ، رسم می کنیم:

```
x=-1:0.1:1;
plot(x,x.^3)
```

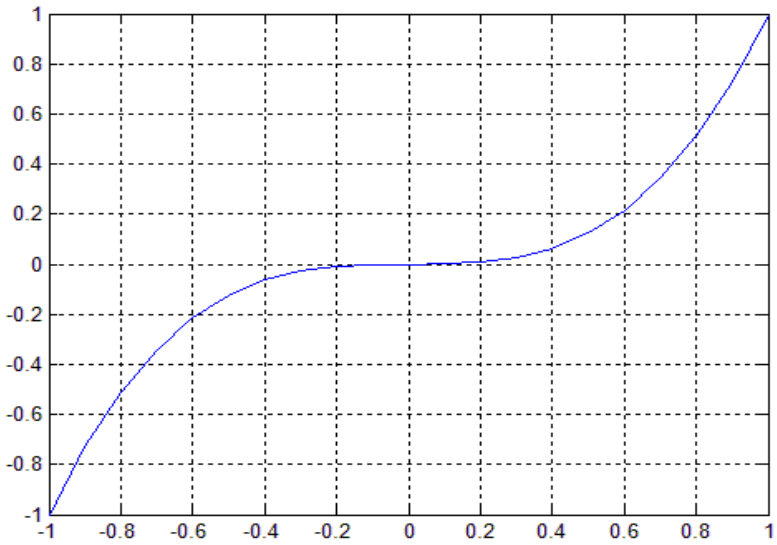
نتیجه :



این بار همان منحنی را به همراه دستور grid ترسیم می کنیم:

```
x=-1:0.1:1;
plot(x,x.^3)
grid
```

نتیجه :



مشاهده می کنید که پس زمینه شکل به صورت چهارخانه نمایش داده شده است و بدین ترتیب می توان خیلی راحت تر مقادیر مربوط به هر نقطه از منحنی را تشخیص داد.

تعیین رنگ پس زمینه شکل ها با دستور whitebg در متلب:

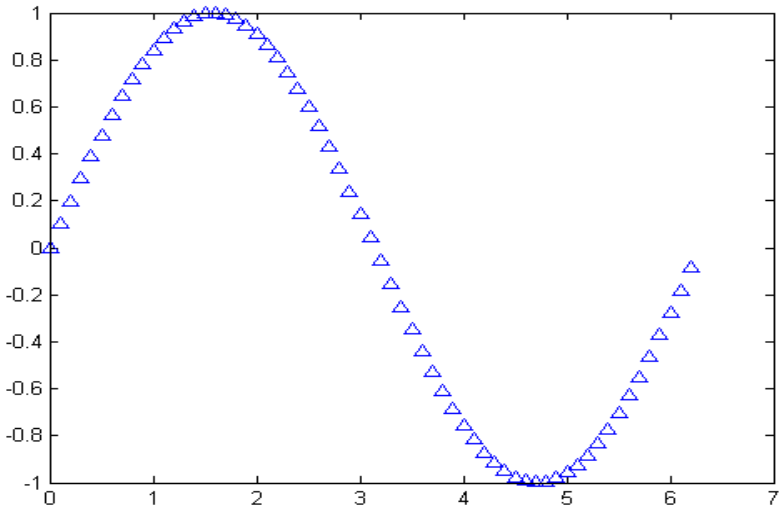
همان طور که می دانید ، نرم افزار متلب به طور خودکار رنگ پس زمینه شکل ها را سفید در نظر می گیرد اما ممکن است در مواردی بخواهیم رنگ پس زمینه شکل ، رنگ دیگری باشد . در این موارد می توانیم از دستور whitebg در متلب استفاده کنیم . باید دقت کنید که اگر رنگ پس زمینه شکل ها را تغییر دادید و دوباره خواستید به حالت اولیه ، یعنی رنگ سفید ، برگردد باید مجددا از دستور whitebgاستفاده کرده و این بار رنگ سفید را انتخاب کنید . به مثال زیر توجه کنید:

مثال :

ابتدا شکلی را به صورت معمولی و با پس زمینه سفید رسم می کنیم:

```
clear all
close all
clc

t=0:0.1:2*pi;
x=sin(t);
plot(t,x,'^')
```

نتیجه :

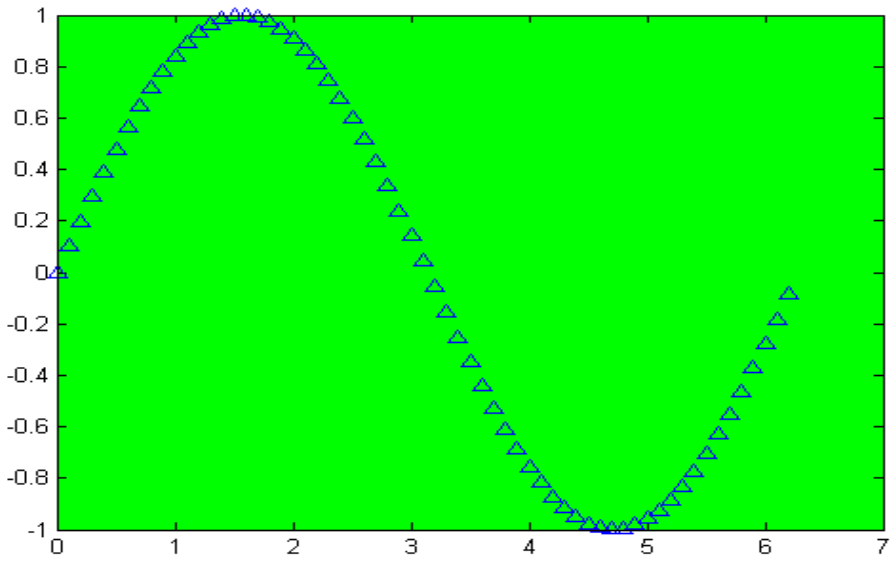


حال این بار با دستور whitebg رنگ پس زمینه را به سبز تغییر می دهیم:

```
clear all
close all
clc

t=0:0.1:2*pi;
x=sin(t);
plot(t,x,'^')
whitebg('green')
```

مشاهده می کنید که باید رنگ مورد نظرمان را درون پرانتز دستور whitebg بنویسیم.

نتیجه :



برای اینکه شکل هایی که از این به بعد رسم می کنیم با پس زمینه سفید اجرا شوند ، باید دستور زیر را بنویسیم:

```
whitebg('white')
```

رسم چند شکل کنار هم:

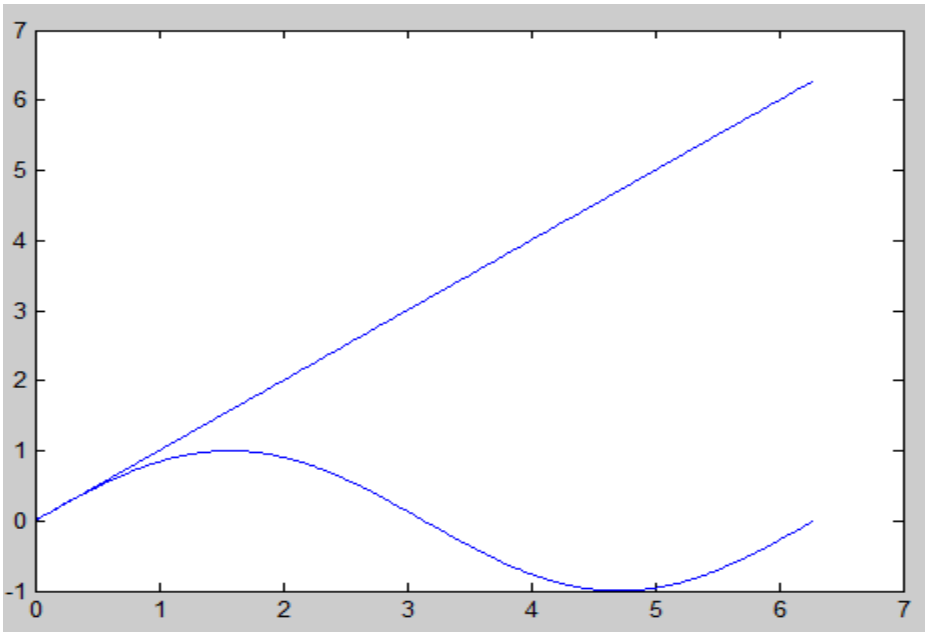
Hold on:

با استفاده از این دستور مانع پاک شدن صفحه نمایش میشویم تا نمودار های بعدی نیز بر روی نمودار فعلی بیافتند.

مثال :

```
t=0:.01:2*pi
y1=sin(t)
y2=t
y3=t-(t.^3)/6+(t.^5)/120-(t.^7)/5040
plot(t,y1)
hold on
plot(t,y2)
hold on
```

نتیجه :



:Figure

دستور figure باعث ایجاد یک صفحه رسم دیگر میشود به گونه ای که شماره figure در درایه آن نوشته میشود

مثال :

```
t=-1:0.1:30;
t2=-5:0.1:5;
x=sin(t);
y=cos(x);
plot(t,x,'xr',t,y,'^b');
legend('sint','cosx')
figure(2);
plot(t2,sinc(t2),'^b');
grid
legend('tan(t2)')
```

:Subplot

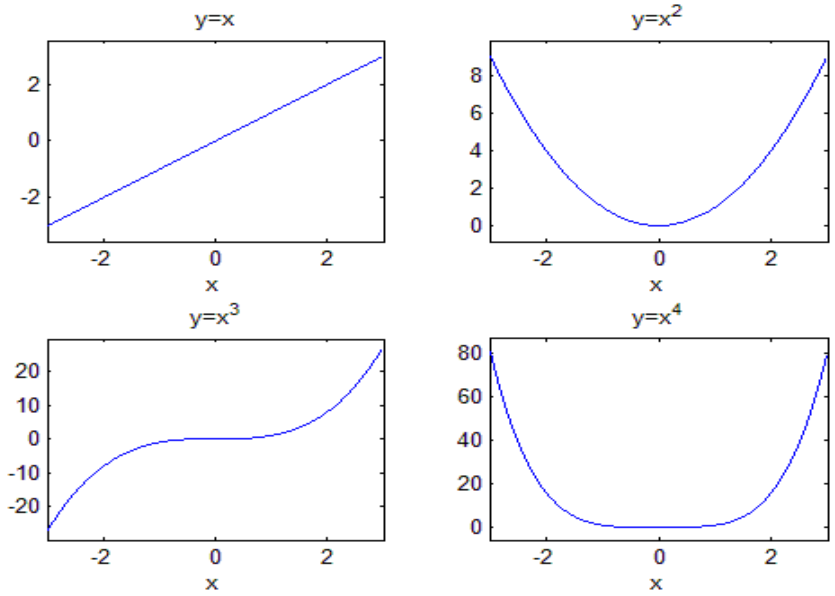
متلب برای هر شکل که باید در خروجی نمایش داده شود ، پنجره ای جدید را باز می کند . اما ممکن است که نیاز داشته باشیم که چندین شکل به طور جداگانه اما کنار هم و در یک پنجره رسم شوند تا بتوانیم آنها را با هم مقایسه کنیم . برای این منظور در متلب از دستور subplot استفاده می شود .
نحوه استفاده از دستور subplot را در مثال زیر شرح می دهیم:

مثال :

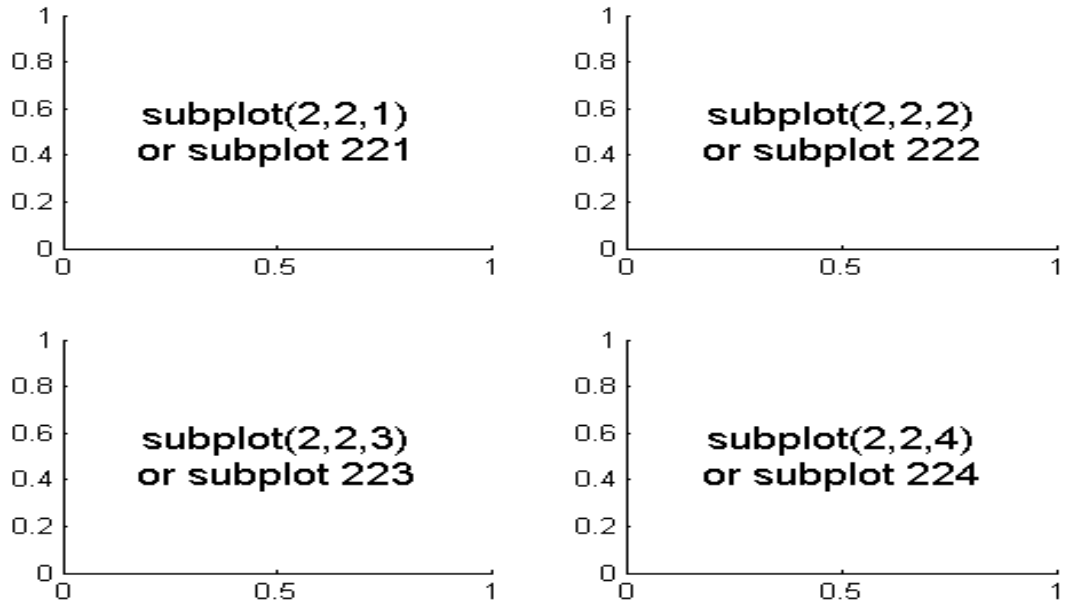
فرض کنید بخواهیم 4 تابع $y=x$ ، $y=x^2$ ، $y=x^3$ و $y=x^4$ را با دستور ezplot رسم کنیم و همچنین بخواهیم که نتیجه به صورت 4 شکل جداگانه اما در یک پنجره و در کنار هم نمایش داده شود برای این منظور کدهای زیر را می نویسیم:

```
subplot(2,2,1)
ezplot('x',[-3,3])
title('y=x')

subplot(2,2,2)
ezplot('x^2',[-3,3])
title('y=x^2')

subplot(2,2,3)
ezplot('x^3',[-3,3])
title('y=x^3')

subplot(2,2,4)
ezplot('x^4',[-3,3])
title('y=x^4')
```

نتیجه :



مشاهده می کنید که چهار شکل مورد نظرمان در کنار هم و در یک پنجره نمایش داده شده اند . اما اکنون شرح بدهیم که چگونه با دستور subplot تعداد شکل ها و موقعیت آنها در کنار هم را تعیین کرده ایم . اگر به کدها نگاه کنید متوجه خواهید شد که برای هر شکل ، 3 خط کد نوشته ایم ، خط اول با دستور subplot است که تعیین می کند موقعیت آن شکل در کنار سایر شکل ها چگونه باید باشد ، خط دوم با دستور ezplot است که برای رسم تابع مورد نظرمان می باشد و خط سوم عنوانی را به شکل اختصاص می دهد تا آن را در کنار سایر شکل ها به راحتی تشخیص بدهیم . چون 4 شکل داریم بنابراین 4 بار از دستور subplot در کدها استفاده کردیم . برای 4 شکل ، مقادیری که باید درون پرانتز هر دستور subplot نوشته شود به صورت شکل زیر می باشد:



با توجه به شکل بالا ، مشاهده می کنید که در پنجره نمایش شکل ها ، 2 ردیف و 2 ستون متشکل از شکل ها خواهیم داشت . بنابراین عدد اول درون پرانتز دستور subplot نشان دهنده تعداد کل ردیف ها و عدد دوم درون پرانتز دستور subplot نشان دهنده تعداد کل ستون ها برای چیدمان شکل ها در کنار هم می باشد . برای هر موقعیت یک عدد در نظر گرفته شده است که چون 4 شکل داریم این عدد از 1 تا 4 می تواند باشد . این عدد ، سومین عدد درون پرانتز دستور subplot خواهد بود . بنابراین با دستور subplot قبل از هر دستور ezplot ، موقعیت شکل مربوط به آن دستور ezplot را مشخص کرده ایم.

دستور legend در متلب:

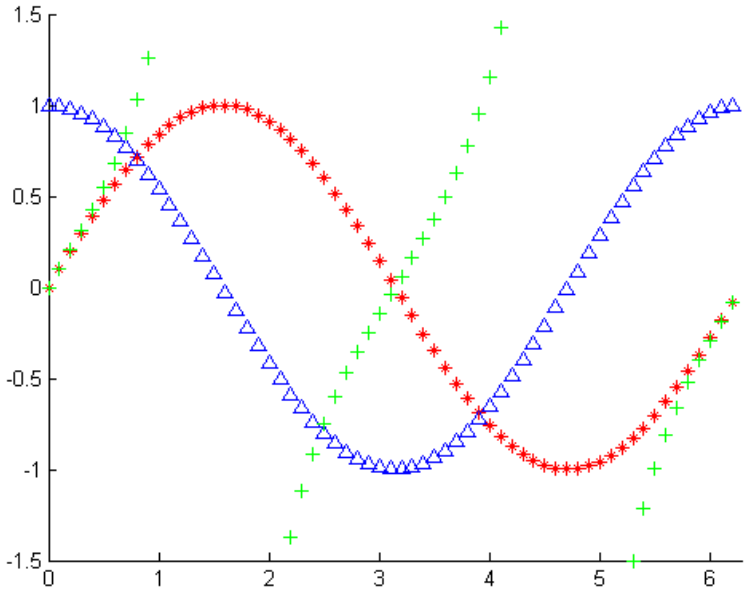
شاید در مقالات زیاد دیده باشید که در یک شکل ، مثلا سه منحنی با علامت های متفاوت کشیده شده باشد و نویسنده برای اینکه خوانندگان گیج نشوند ، لیستی از علامت های به کار رفته برای ترسیم سه منحنی و عنوانی در مقابل آنها در کنار شکل قرار می دهد . این نوع نمایش لیست علامت ها و اطلاعات هر منحنی ، برای چندین منحنی به کار رفته در یک شکل ، با دستور legend در متلب ایجاد می شود . به مثال زیر توجه کنید:

مثال :

ابتدا شکلی را به صورت معمولی ، شامل سه منحنی رسم می کنیم:

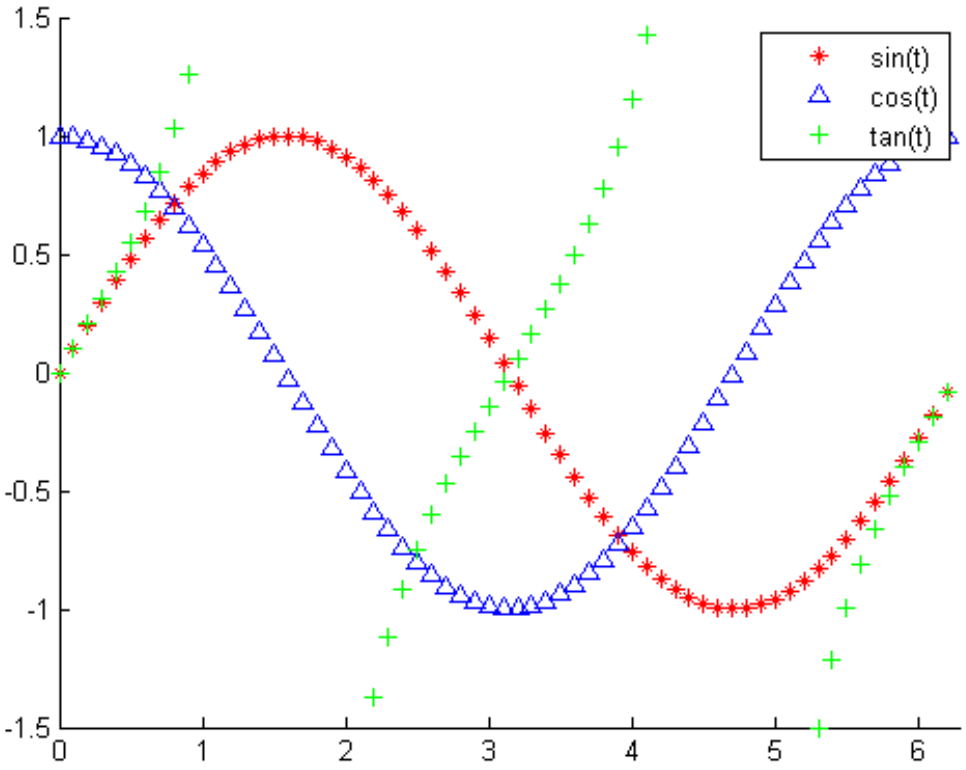
```
clear all
close all
clc

t=0:0.1:2*pi;
x=sin(t);
y=cos(t);
z=tan(t);

hold on
plot(t,x,'*r')
plot(t,y,'^')
plot(t,z,'+g')
axis([0 2*pi -1.5 1.5])
```

نتیجه :



اکنون با افزودن دستور legend ، می خواهیم مشخص کنیم که هر منحنی مربوط به کدامیک از توابع sin، cosو tan می باشد:

```
clear all
close all
clc

t=0:0.1:2*pi;
x=sin(t);
y=cos(t);
z=tan(t);

hold on
plot(t,x,'*r')
plot(t,y,'^')
plot(t,z,'+g')
axis([0 2*pi -1.5 1.5])

legend('sin(t)','cos(t)','tan(t)')
```



مشاهده می کنید که کادری در بالا و سمت راست شکل نمایش داده شده است که مشخص کرده است که هر منحنی نمایش دهنده کدام تابع می باشد . همچنین می توانید این کادر را با موس در شکل جابجا کنید.

دستور plot3 در متلب:

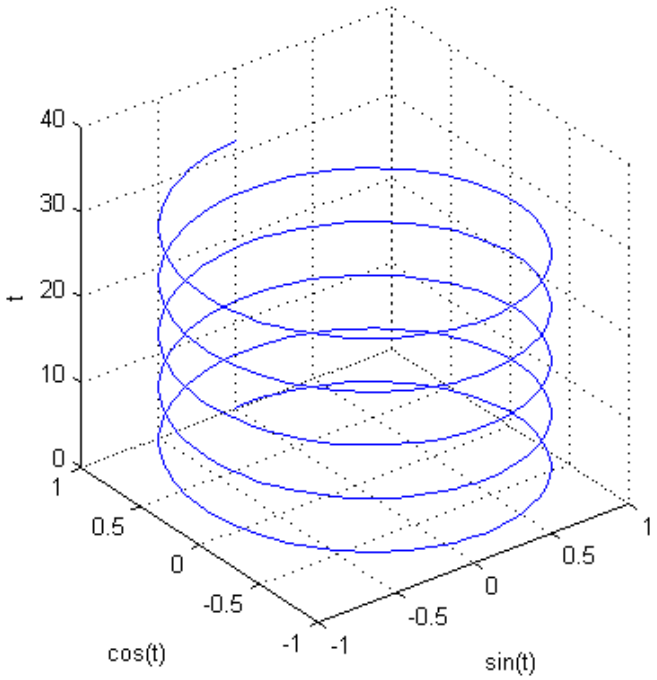
دستور plot3 در متلب برای رسم خطوط به صورت سه بعدی به کار می رود . یک مثال مشهور در این زمینه را شرح می دهیم:

مثال :

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

نتیجه :

دقت شود که محور عمودی برای متغیر t خواهد بود و دو محور افقی نیز به صورت دو تابع از متغیر t می باشند . دستورات xlabel ، ylabel و zlabel برای تعیین عنوان برای محورهای مختصات می باشند . دستور grid نیز باعث می شود که مقیاس ها بر روی دیواره های پشت شکل نمایش داده شوند تا سه بعدی بودن شکل به خوبی نمایش داده شود . دستور axis square نیز تعیین کرده است که دو محور افقی شکل با اندازه ای یکسان نمایش داده شوند که باعث می شود تغییرات دایره ای شکل منحنی در جهت سطوح افقی به خوبی مشاهده شود.



کنترل در متلب:

یافتن تابع تبدیل با دستور tf :

مثال :

ابتدا یک ماتریس ضرایب صورت تعریف میکنیم:

```
num=[1 2 0 3];
```

سپس ماتریس دیگری با ضرایب مخرج تعریف میکنیم:

```
den=[4 1 2 3 4];
```

بااستفاده از دستور tf(num,den) تابع تبدیل کسر $\frac{num}{den}$ را به ما نمایش خواهد داد:

نتیجه :

```
tf(num,den)

Transfer function:
       s^3 + 2 s^2 + 3
-----------------------------
4 s^4 + s^3 + 2 s^2 + 3 s + 4
```

یافتن صفرها و قطبهای B(S)/A(S) با MATLAB :

در MATLAB برای یافتن صفرها ، قطبها و بهره k تابع B(s)/A(s) می توان دستور زیر را بکار برد :

$$[z, p, k] = tf2zp(num, den)$$

سیستم تعریف شده به صورت زیر را در نظر بگیرید :

$$\frac{B(s)}{A(s)} = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}$$

برای یافتن صفرها (z) ، قطبها (p) و بهره (k) دستورات زیر را وارد می کنیم :

```
>> num=[0 0 4 16 12];
>> den=[1 12 44 48 0];
>> [z,p,k]=tf2zp(num,den)
```

در اینصورت ، کامپیوتر خروجی زیر را بدست می دهد :

```
z =

    -3
-1



p =

         0
-6.0000
-4.0000
-2.0000



k =

     4
```

بسط به کسرهای جزئی با MATLAB :

تابع B(s)/A(s) زیر را در نظر بگیرید .

$$\frac{B(s)}{A(s)} = \frac{num}{den} = \frac{b_0 s^n + b_1 s^{n-1} + \ldots + b_n}{s^n + a_1 s^{n-1} + \ldots + a_n}$$

که در آن ، بعضی ai ها و bi ها ممکن است صفر باشد . در MATLAB ضرایب چندجمله ایهای صورت و مخرج در بردارهای ردیفی مشخص می شوند . یعنی :

$$num = [b_0 \ b_1 \ \ldots \ b_n]$$
$$den = [1 \ a_1 \ \ldots \ a_n]$$

دستور

$$[r, p, k] = residue(num, den)$$

باقیمانده ها (r) ، قطبها (p) و جملات غیرکسری (k) نسبت دو چندجمله ای B(s) و A(s) را بدست می دهد . بسط به کسرهای جزئیB(s)/A(s) عبارت است از :

$$\frac{B(s)}{A(s)} = \frac{r(1)}{s-p(1)} + \frac{r(2)}{s-p(2)} + \ldots + \frac{r(n)}{s-p(n)} + k(s)$$

مثال) تابع تبدیل زیر را در نظر بگیرید :

$$\frac{B(s)}{A(s)} = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}$$

برای این تابع

$$num = [2\ 5\ 3\ 6]$$
$$den = [1\ 6\ 11\ 6]$$

با اجرای دستور معرفی شده ، خواهیم داشت :

```
>> num=[2 5 3 6];
>> den=[1 6 11 6];
>> [r,p,k]=residue(num,den)

r =

-6.0000
-4.0000
3.0000



p =

-3.0000
-2.0000
-1.0000



k =

    2
```

توجه کنید که باقیمانده ها در بردار ستونی r ، محل قطبها در بردار ستونی p و خارج قسمت در بردار ردیفی k برگردانده می شود . یعنی بسط کسرهای جزئی B(s)/A(s) بدست آمده با MATLAB عبارت است از :

$$\frac{B(s)}{A(s)}=\frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6}=\frac{-6}{s+3}+\frac{-4}{s+2}+\frac{3}{s+1}+2$$

یافتن پاسخ به ورودی دلخواه :

برای یافتن پاسخ به ازای ورودی دلخواه می توان دستور Isim را به کار برد . دستور زیر پاسخ به ورودی زمانی r بدست می دهند .

$$y=lsim(num,den,r,t)$$

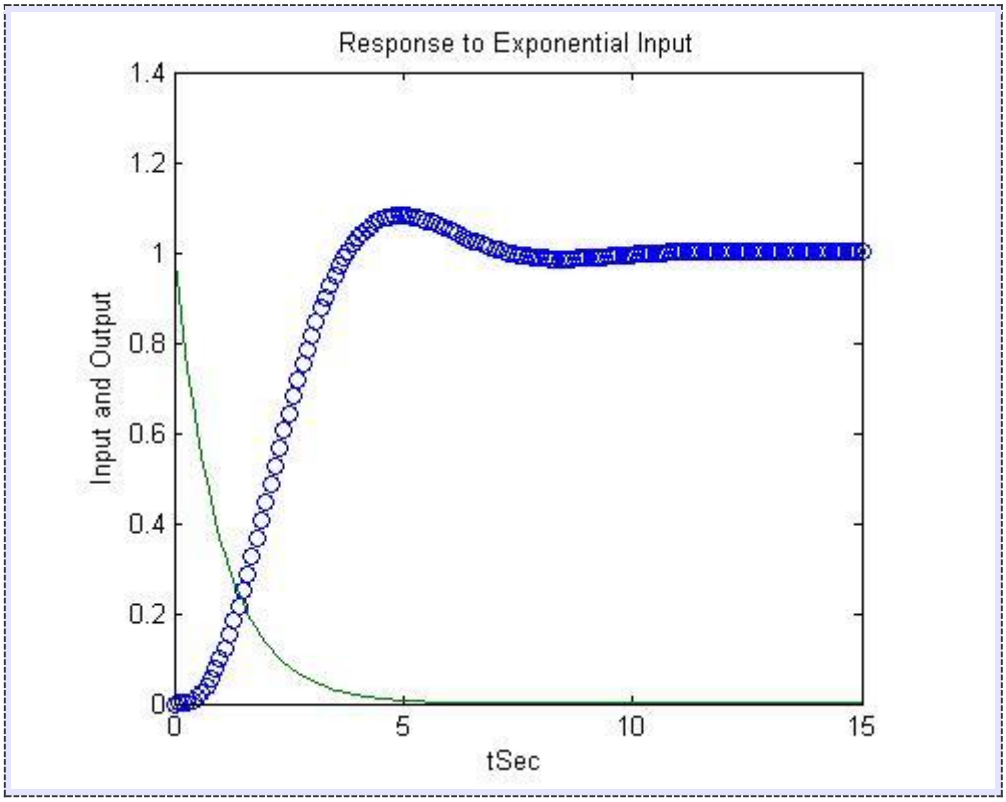
مثال) سیستمی با تابع تبدیل حلقه بسته زیر را در نظر بگیرید .

$$\frac{C(s)}{R(s)}=\frac{1}{s^3+s^2+s}$$

پاسخ این سیستم به ورودی R = e-t را رسم کنید .
در محیط MATLAB ، یک m-file ایجاد کرده و کدهای زیر را در آن وارد می کنیم .

```
num=[ 1];
den=[1 1 1 0];
t=0:0.1:15;
r=exp(-t);
c=lsim(num,den,r,t);
plot(t,c,'o',t,r,'-')
title('Response to Exponential Input')
xlabel('tSec')

ylabel('Input and Output')
```

پس از اجرا ، خروجی برنامه به صورت زیر خواهد بود .



پاسخ ضربه

با اجرای دستورات زیر میتوان پاسخ ضربه یک سیستم را با MATLAB به دست اورد.

impulse(num,den)

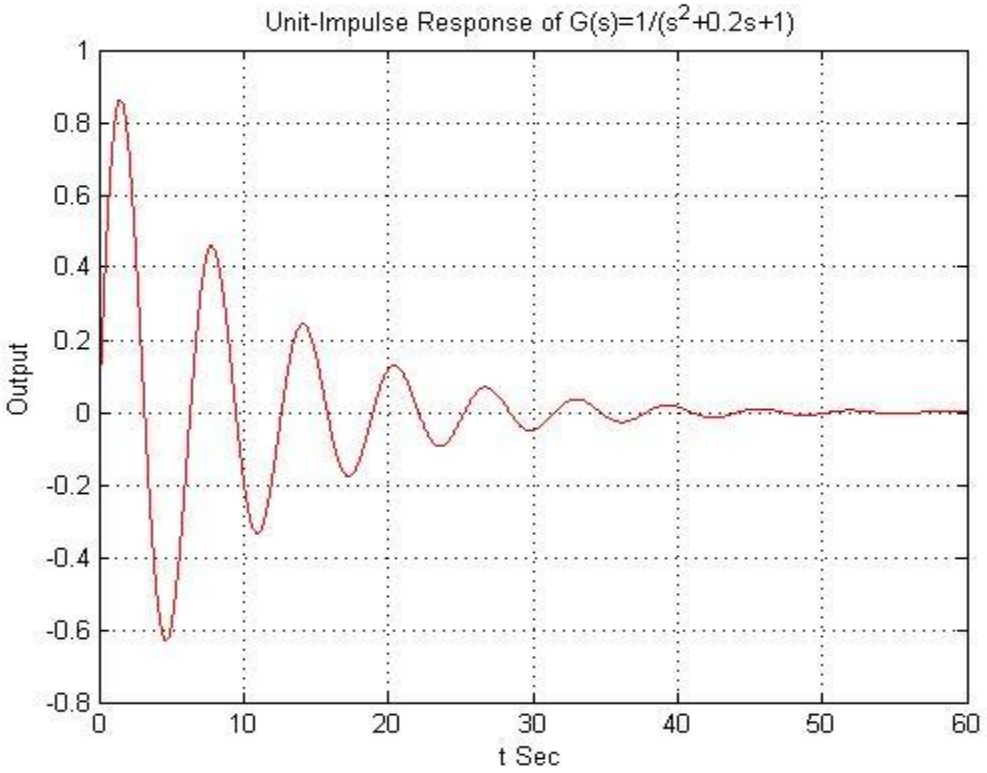
impulse(A,B,C,D)

[y,x,t]=impulse(num,den,t)

[y,x,t]=impulse(A,B,C,D,t)

تابع تبدیل G(s)=1/s^2+0.2s+1را به ازای ورودی ضربه بررسی میکنیم:

```
num = [0 0 1];

den = [1 0.2 1];

t=0:0.01:60;

[y,x,t]=impulse(num,den,t)

plot(t,y)

grid on

title('Impulse Response of G(s)=25/(sˆ2+4s+25)’)

xlabel (‘t Sec’)

ylabel('Output’)
```



رسم نمودارهای پاسخ پله با MATLAB :

سیستم با تابع تبدیل حلقه بسته زیر را در نظر بگیرید .

$$\frac{C(s)}{R(s)} = \frac{1}{s^2 + 2\xi s + 1}$$

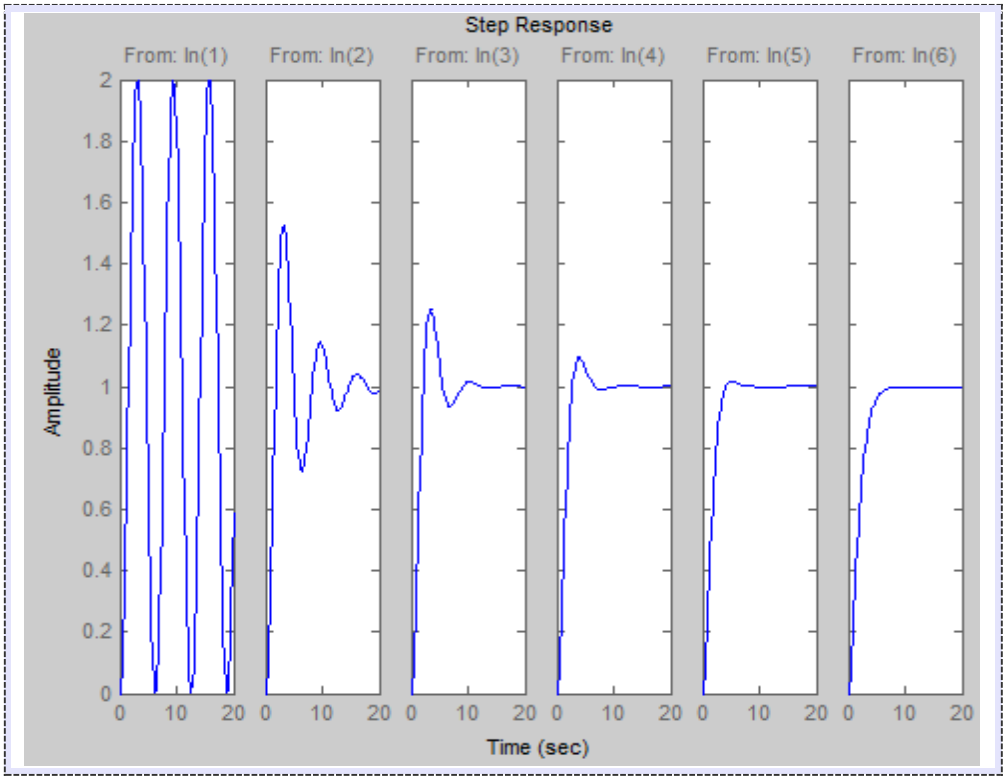
منحنی پاسخ پله c(t) را به ازای ξ های مختلف رسم کنید .

$$\xi = 0 \, , 0.2 \, , 0.4 \, , 0.6 \, , 0.8 \, , 1$$

برای رسم منحنی پاسخ پله این سیستم کنترلی ، یک m-file ایجاد کرده و کدهای زیر را درون آن وارد می کنیم .

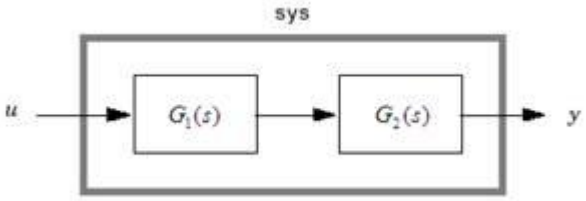
```
t=0:.1:10
zeta=[0 .2 .4 .6 .8 1]
for n=1:6

   sys(n)=tf([1],[1 2*zeta(n) 1])

end
step(2*t,sys)
```

و پس از اجرا ، خواهیم داشت :

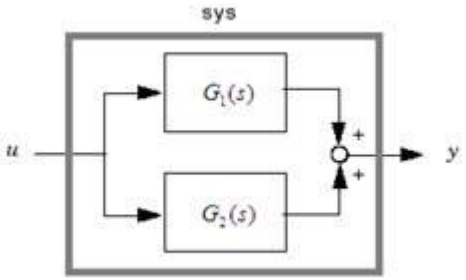


یافتن تابع تبدیل متوالی ، موازی و فیدبک دار (حلقه بسته) با MATLAB :

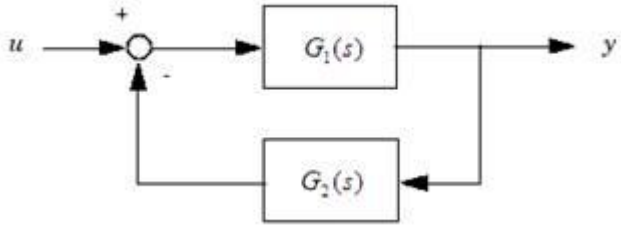
در تحلیل سیستمهای کنترل غالبا لازم می شود که تابع تبدیل اتصالهای متوالی ، موازی و فیدبک دار (حلقه بسته) را حساب کنیم . فرض کنید دو جزء با توابع تبدیل G1(s) و G2(s) ، به صورتهای نشان داده شده در زیر ، به هم متصل اند



سیستمهای متوالی شده



سیستمهای موازی شده



سیستمهای فیدبک دار (حلقه بسته)

که

$$G_1(s) = \frac{num1}{den1} \quad , \quad G_2(s) = \frac{num2}{den2}$$

برای بدست آوردن تابع تبدیل سیستمهای متوالی شده ، موازی شده یا فیدبک دار (حلقه بسته) می توان دستورهای زیر را به کار برد :

$$[num, den] = series(num1, den1, num2, den2)$$
$$[num, den] = parallel(num1, den1, num2, den2)$$
$$[num, den] = feedback(num1, den1, num2, den2)$$

مثلا حالت زیر را در نظر بگیرید :

$$G_1(s)=\frac{10}{s^2+2s+10} \quad , \quad G_2(s)=\frac{5}{s+5}$$

برنامه زیر تابع تبدیل هر یک از آرایشهای G1(s) و G2(s) را نشان می دهد .

```
>> num1=[0 0 10];
>> den1=[1 2 10];
>> num2=[0 5];
>> den2=[1 5];
>> [num,den]=series(num1,den1,num2,den2);
>> printsys(num,den)

num/den =

           50
  ----------------------
  s^3 + 7 s^2 + 20 s + 50
>> [num,den]=parallel(num1,den1,num2,den2);
>> printsys(num,den)

num/den =

    5 s^2 + 20 s + 100
  ----------------------
  s^3 + 7 s^2 + 20 s + 50
>> [num,den]=feedback(num1,den1,num2,den2);
>> printsys(num,den)

num/den =

        10 s + 50
  ----------------------
  s^3 + 7 s^2 + 20 s + 100
```

به دستور زیر توجه کنید :

$$print\ sys=(num,den)$$

این دستور num/den سیستم مورد نظر را نشان می دهد

رسم مکان هندسی ریشه ها با MATLAB :

در رسم مکان هندسی ریشه ها با MATLAB با سیستمهایی کار می کنیم که معادله آنها را بتوان به شکل معادله زیر نوشت .

$$1+k\frac{num}{den}=0$$

که در آن num ، چندجمله ای صورت و den ، چندجمله ای مخرج است . یعنی :

$$num=(s+z_1)(s+z_2)...(s+z_m)=s^m+(z_1+z_2+...+z_m)s^{m-1}+...+z_1z_2...z_m$$
$$den=(s+p_1)(s+p_2)...(s+p_n)=s^n+(p_1+p_2+...+p_n)s^{n-1}+...+p_1p_2...p_n$$

پاسخ این سیستم به ورودی R = e-t را رسم کنید .توجه کنید که هر دو بردار num و den باید بر حسب توانهای نزولی نوشته شوند . دستور MATLAB برای رسم نمودار مکان هندسی ریشه ها به صورت زیر است :

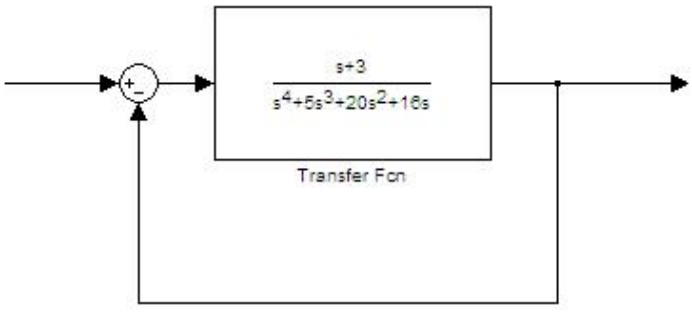
$$rlocus(num,den)$$

با دادن این فرمان ، نمودار مکان هندسی ریشه ها روی صفحه ترسیم می شود . بردار بهره k به طور خودکار تعیین می شود . بردار بهره ، تمام مقادیر بهره ای را که طبهای حلقه بسته به ازای آنها محاسبه می شود در بردارد .

اگر بخواهیم مکان هندسی ریشه ها با علامت 'o' یا 'x' رسم شوند باید دستورهای زیر را بکار بریم :

$$r=rlocus(num,den)$$
$$plot(r,'o') \quad or \quad plot(r,'x')$$

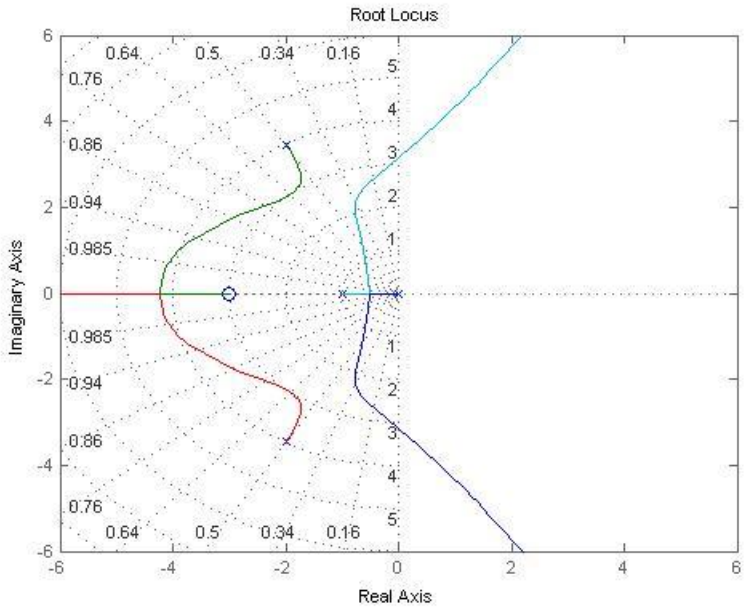
مثال) سیستم کنترل شکل زیر را در نظر بگیرید .



مکان هندسی ریشه ها روی محورهایی با مقیاس برابر رسم کنید .
در محیط MATLAB ، یک m-file ساخته و کدهای زیر را در آن وارد می کنیم .

```
num=[0 0 0 1 3];
den=[1 5 20 16 0];
rlocus(num,den)
v=[-6 6 -6 6];
axis(v);
grid;
```

در اینصورت ، نمودار مکان هندسی ریشه ها بدین صورت خواهد شد .



رسم نمودار نایکوئیست با دستور nyquist در متلب:

نمودار نایکوئیست در درس های کنترل خطی و پردازش سیگنال به کار می رود و معمولا برای بررسی پایداری سیتم های دارای فیدبک استفاده می شود . نمودار نایکوئیست یک تابع به این صورت رسم می شود که قسمت حقیقی تابع بر روی محور افقی و قسمت موهومی تابع بر روی محور عمودی رسم می شود ، اما این می شود یک نقطه به ازای یک فرکانس خاص و چون تابع ، تابعی از فرکانس است این تابع را برای فرکانس های مختلف رسم می کنیم که نتیجه آن به صورت خط های نمودار نایکوئیست می باشد . دستور nyquist در متلب برای رسم نمودار نایکوئیست یک تابع به کار می رود . به مثال زیر توجه کنید:

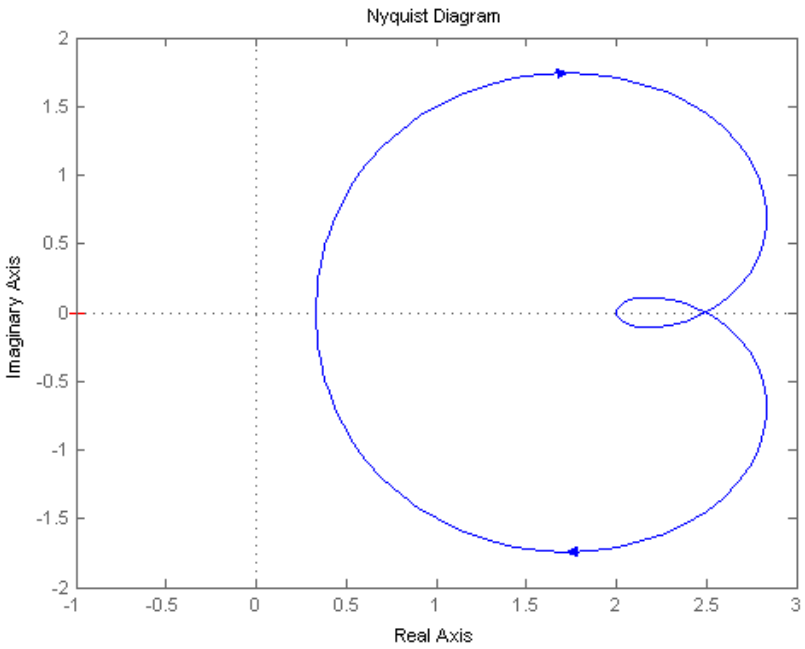
مثال :

فرض کنید بخواهیم نمودار نایکوئیست تابع $H(s)=\frac{2s^2+5s+1}{s^2+2s+3}$ را رسم کنیم . می نویسیم:

```
clear all
close all
clc

H=tf([2 5 1],[1 2 3])
nyquist(H)
```

ضرایب صورت و مخرج تابع را به دستور tf می دهیم تا تابع را بسازد و سپس نمودار نایکوئیست تابع ساخته شده را با دستور nyquist رسم می کنیم.

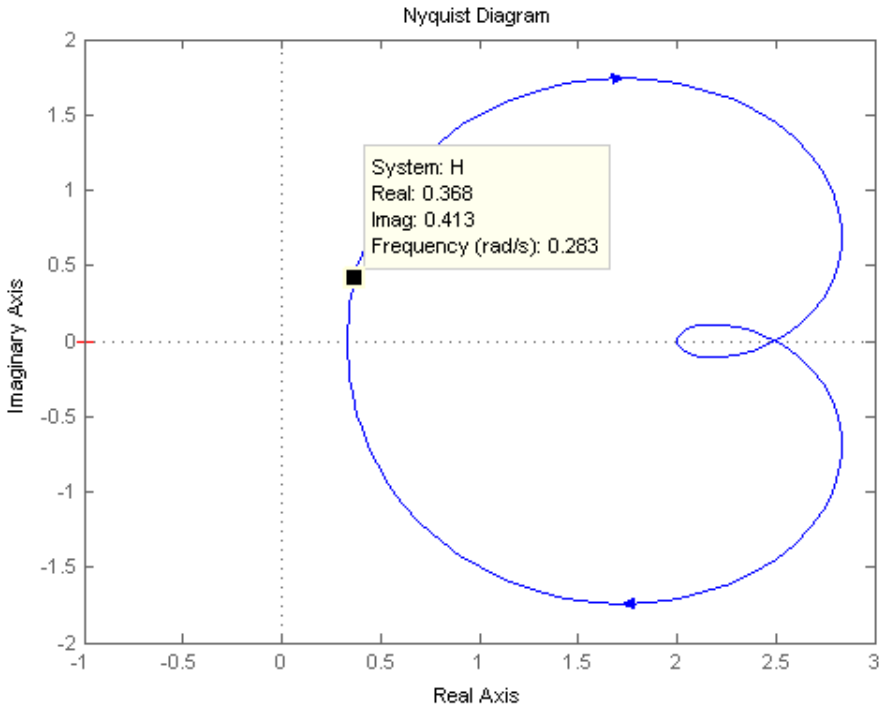
نتیجه :



مشاهده می کنید که خود دستور nyquist ، عنوان های محورها را مشخص کرده است.

نکته :

شاید بخواهید مقادیر قسمت های حقیقی و موهومی تابع را برای یک فرکانس خاص بیابید ، برای این منظور تنها کافی است که در بالای پنجره شکل باز شده ، بر روی گزینه Data Cursor به شکل  کلیک کنید و سپس بر روی یک نقطه از نمودار کلیک کنید تا اطلاعات مربوط به آن نقطه ، یعنی فرکانس و بخش های حقیقی و موهومی تابع در آن نقطه ، نمایش داده شود . نتیجه به صورت شکل زیر می باشد:



نکته :

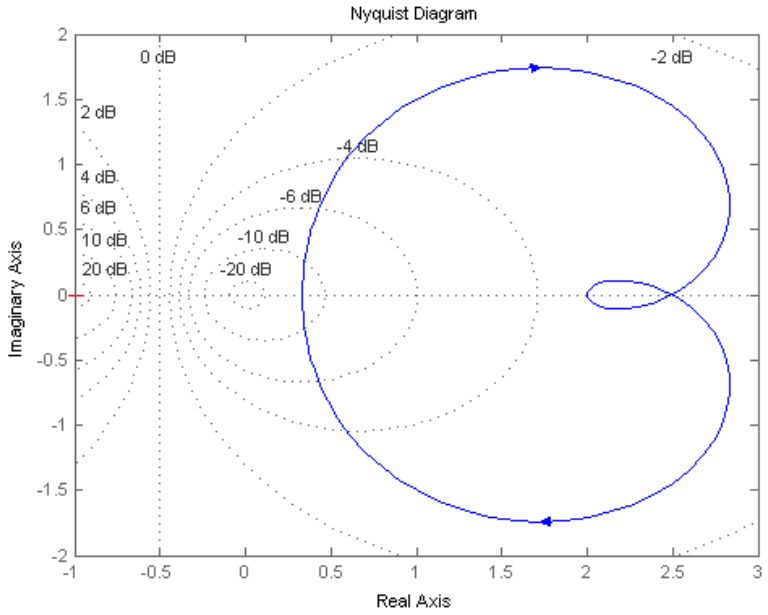
با استفاده از دستور grid ، می توانیم دوایر M (M circles) را در نمودار نایکوئیست رسم کنیم . به مثال زیر توجه کنید:

مثال :

هم مثال قبل را این بار با دستور grid می نویسیم:

```
clear all
close all
clc

H=tf([2 5 1],[1 2 3])
nyquist(H)
grid
```

نتیجه :



رسم نمودارهای بوده با MATLAB :

دستور bode دامنه و زاویه فاز پاسخ فرکانسی سیستمهای خطی پیوسته مستقل از زمان را حساب می کند .

مثال) تابع تبدیل زیر را در نظر بگیرید .

$$G(s) = \frac{25}{s^2 + 4s + 25}$$

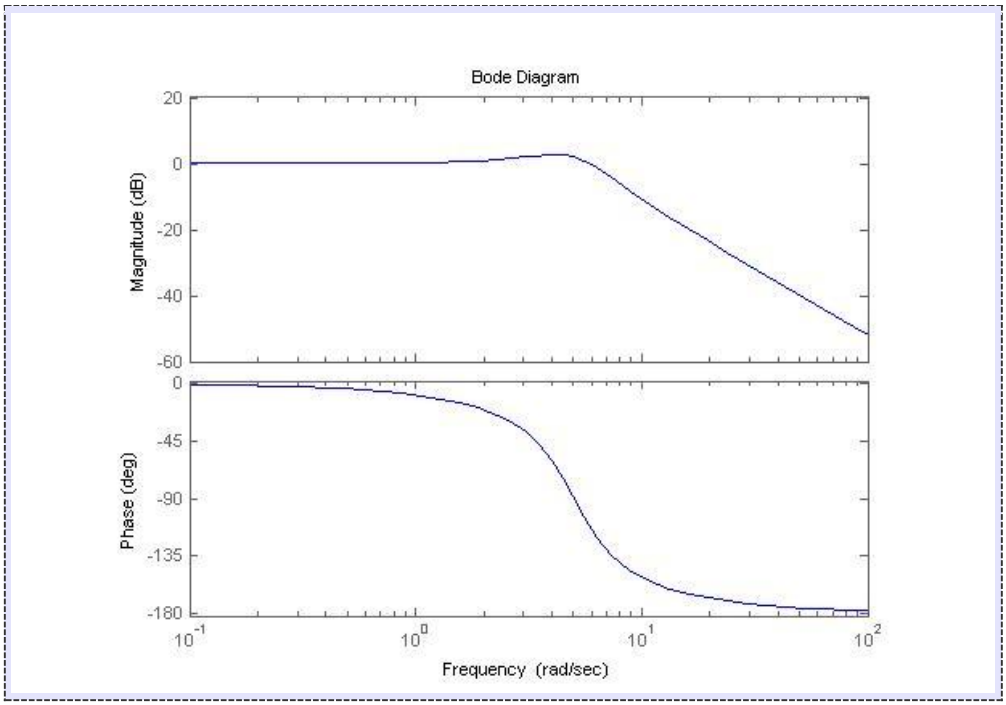
نمودار بوده این تابع تبدیل را رسم کنید .

وقتی سیستم به شکل زیر تعریف شده است

$$G(s) = \frac{num(s)}{den(s)}$$

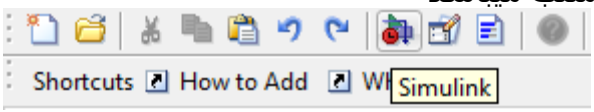
برای رسم نمودار بوده این سیستم ، کدهای زیر را در MATLAB وارد می کنیم .

```
>> num=[0 0 25];
>> den=[1 4 25];
>> bode(num,den)
```

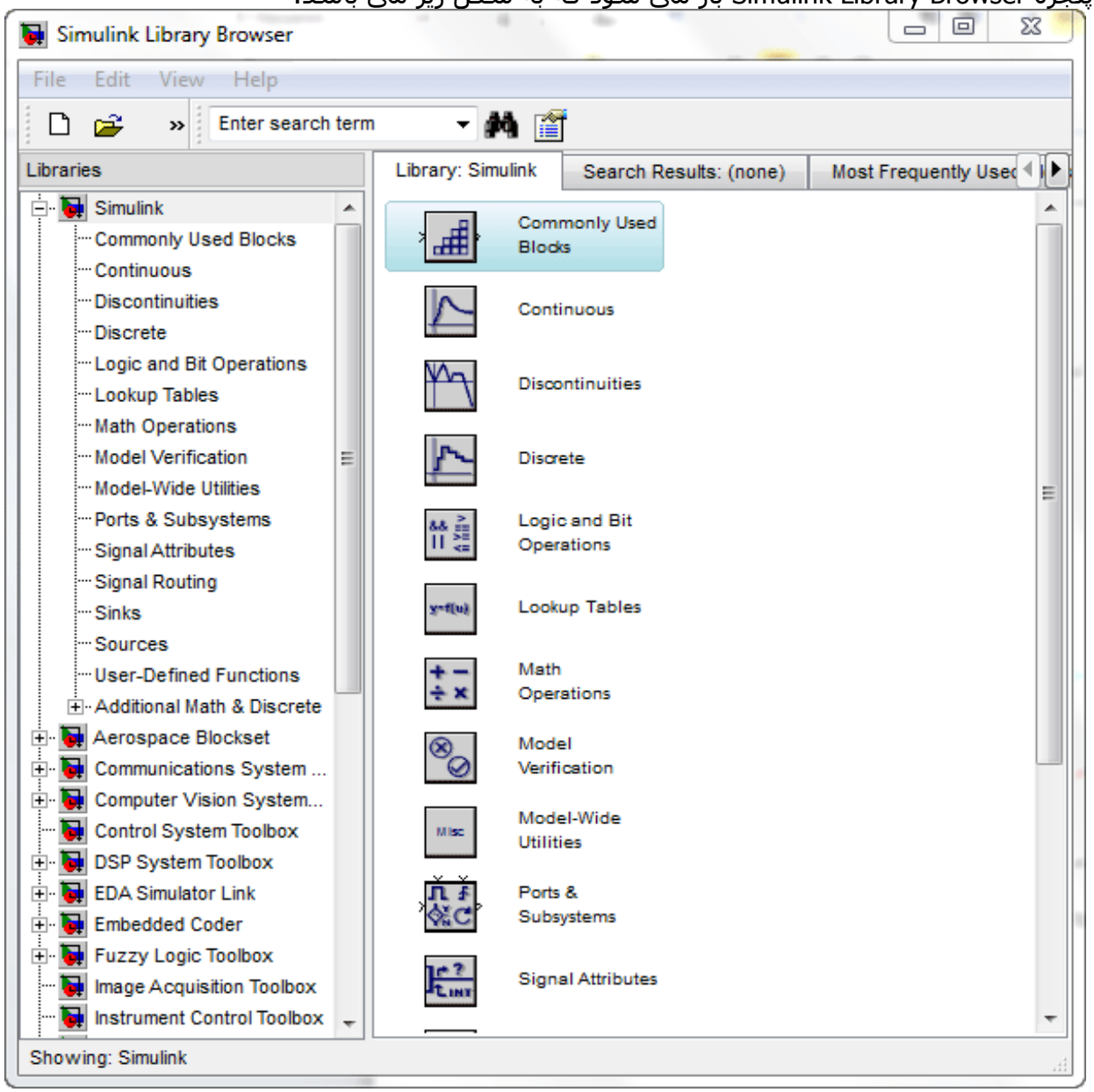


سیمولینک (SIMULINK) در متلب:

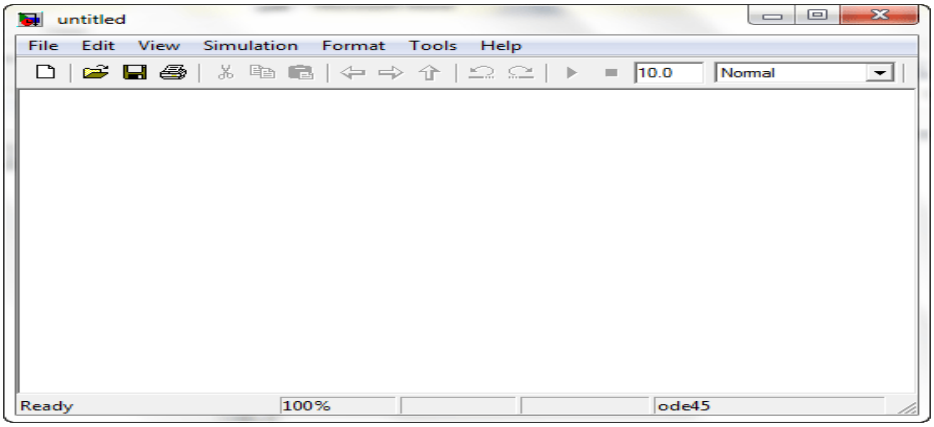
در نرم افزار متلب برای باز کردن پنجره مربوط به سیمولینک (SIMULINK) ، در پنجره Command دستور simulink را نوشته و سپس کلید enter از کیبورد را فشار بدهید یا راه ساده تر آن کلیک بر روی Simulink در نوار ابزار متلب میباشد



پنجره Simulink Library Browser باز می شود که به شکل زیر می باشد:

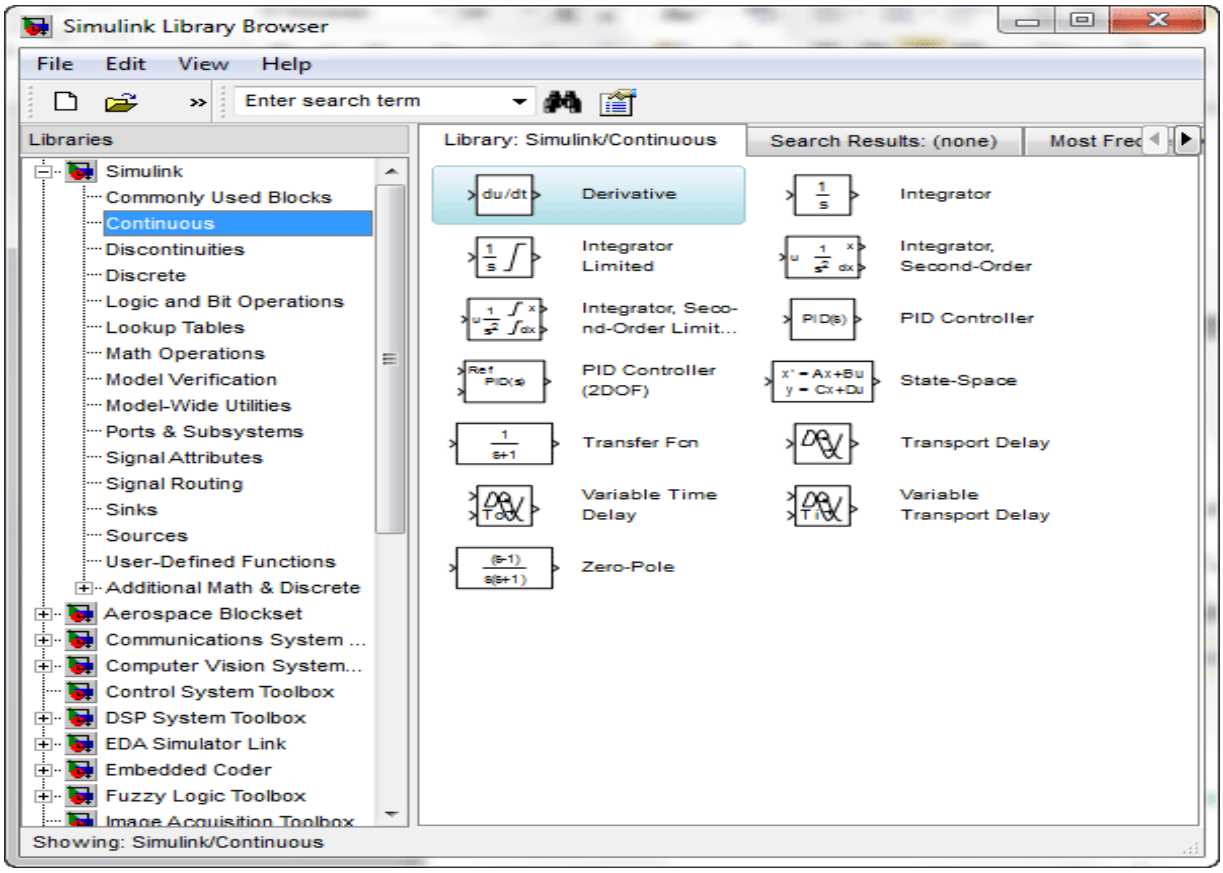


از منوی File ، گزینه New و سپس گزینه Model را انتخاب کنید . یک پنجره خالی به شکل زیر باز می شود:

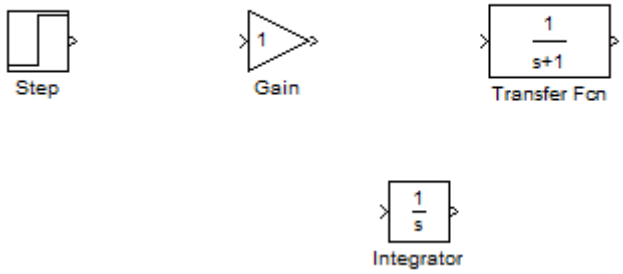


این پنجره را پنجره Model می نامیم . آنچه که از این به بعد باید انجام دهیم این است که بلوک هایی (blocks) را که می خواهیم از library های سیمولینک انتخاب کنیم و آنها را در پنجره Model کپی کنیم.

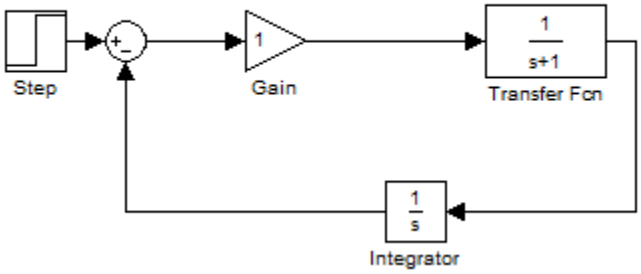
در اینجا میخواهیم پاسخ سیستم کنترلی $\frac{s+1}{2s^2+3s+3}$ با ضریب بهره 100 را به فیدبک 1/s به ازای ورودی پله در سیمولینک مشاهده کنیم. بنابراین از پنجره Simulink Library Browser ، گزینه Continuous را انتخاب می کنیم . بلوک های library Continuous به صورت شکل زیر نمایش داده می شوند:



. برای قرار دادن بلوک تابع تبدیل در پنجره Model ، ابتدا در پنجره Simulink Library Browser بر روی بلوک تابع تبدیل (Transfer Fcn) با موس کلیک کنید (کلیک سمت چپ) و بدون رها کردن کلید موس ، موس را بر روی محلی در پنجره model ببرید و سپس کلید موس را رها کنید . یک بلوک تابع تبدیل در پنجره Model نمایش داده خواهد شد. سپس به یک بلوک انتگرال گیر نیاز داریم . بلوک انتگرال گیر (integrator) را مانند قبل به Model اضاف میکنیم و سپس از Commonly Used Blocks بهره(Gain) را اضاف میکنیم و برای ورودی پله از Source ورودی پله(Step) را اضاف میکنیم.تا بدین جا شکلی مانند زیر خواهیم داشت:



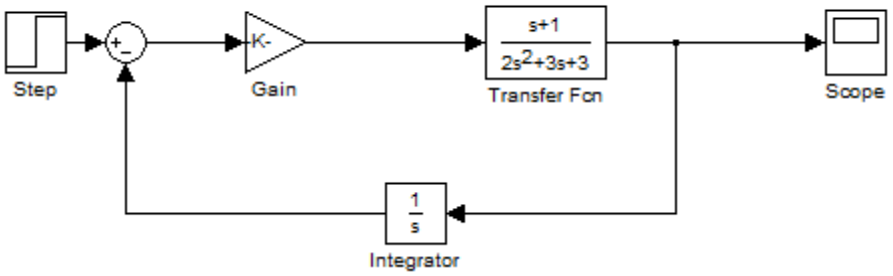
حال برای آنکه دو بلوک را به یکدیگر رسم کنیم باید ابتدا بر روی خروجی بلوک اول با موس کلیک کنیم (کلیک سمت چپ) و بدون رها کردن کلید موس ، موس را بر روی ورودی بلوک دوم ببریم و سپس کلید موس را رها کنیم . یک فلش بین دو بلوک رسم خواهد شد که نشان دهنده اتصال دو بلوک به یکدیگر می باشد . البته برای اتصال دو بلوک به یکدیگر روش دیگری نیز وجود دارد و آن این است که ابتدا بلوک منبع (source block) را با موس انتخاب کنیم و سپس کلید Ctrl از کیبورد را پایین نگه داریم و سپس بر روی بلوک مقصد (destination block) با موس کلیک کنیم (کلیک سمت چپ) . میتوانیم برای زیبا تر شدن شکل بلوک انتگرال را با کلیک راست کردن Format>Flip Block بچرخانیم شکل بلوک ها به صورت زیر خواهد بود:



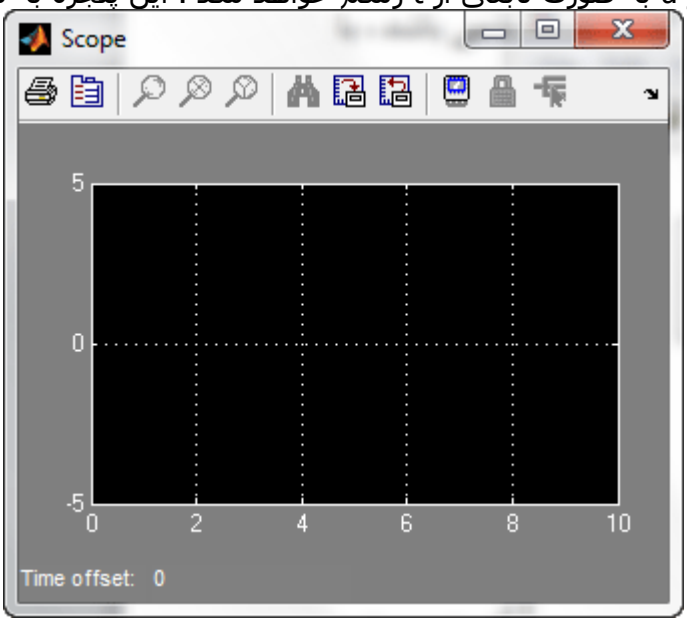
اما اگر در شکل بالا دقت کنید ، متوجه می شوید که میزان بهره بلوک های بهره برابر 1 می باشد و البع تبدیل به صورت پیشفرض میباشد باید تابع تبدیل و مقدار بهره آنها را تغییر بدهیم . برای این منظور باید بر روی هر کدام از بلوک های بهره دو بار کلیک (دابل کلیک) کنیم تا پنجره Function Block Parametersباز شود .

در مورد تابع تبدیل در numerator ماتریس ضرایب صورت و denominator ماتریس ضرایب مخرج را مینویسیم و در مورد بهره مقدار gain را برابر 100 قرار میدهیم.

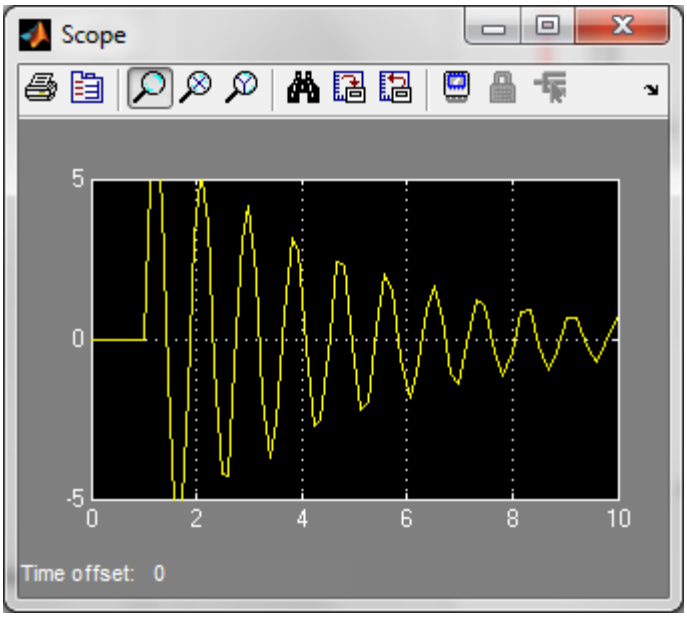
بلوک های لازم برای مدل سازی را در پنجره Model قرار دادیم اما اکنون به بلوکی نیاز داریم تا خروجی را با آن مشاهده کنیم . برای این منظور از Sinks Library بلوک Scope را انتخاب کرده و آن را در پنجره مدل قرار می دهیم ؛ اما چون بلوک تابع تبدیل تنها یک خروجی دارد باید یک انشعاب از خروجی آن بگیریم . برای این منظور موس را بر روی وسط فلشی که دو بلوک انتگرال گیر و تابع تبدیل را به هم متصل کرده است می بریم و سپس کلید Ctrl از کیبورد را فشار داده و پایین نگه می داریم آنگاه با موس کلیک می کنیم (کلیک سمت چپ) و همان طور که کلید موس را نگه داشته ایم نشانگر موس را بر روی ورودی بلوک Scope می بریم و آنگاه کلید موس را رها می کنیم . یک فلش در بین دو بلوک ساخته خواهد شد . شکل بلوک ها به صورت زیر در می آید:



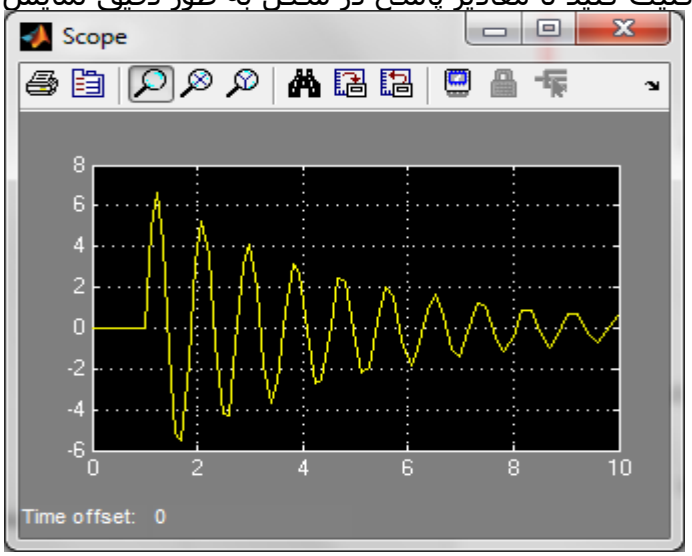
حال برای مشاهده نتیجه ابتدا باید مدل ساخته شده را Save کنیم . از منوی File گزینه Save را انتخاب کنید و سپس با نامی دلخواه مدل ساخته شده را در محل مورد نظرتان Save کنید.
برای آنکه بتوانید مشاهده کنید که در حین شبیه سازی چه اتفاقاتی می افتد باید قبل از شبیه سازی بر روی بلوک Scope با موس دو بار کلیک کنید تا پنجره نمایش آن باز شود . در این پنجره پس از شبیه سازی ، مقدار u به صورت تابعی از t رسم خواهد شد . این پنجره به شکل زیر می باشد:



حال برای مشاهده شبیه سازی از منوی Simulation گزینه Start را انتخاب کنید . شبیه سازی انجام می شود و مقدار u به صورت تابعی از t به صورت شکل زیر نمایش داده می شود:

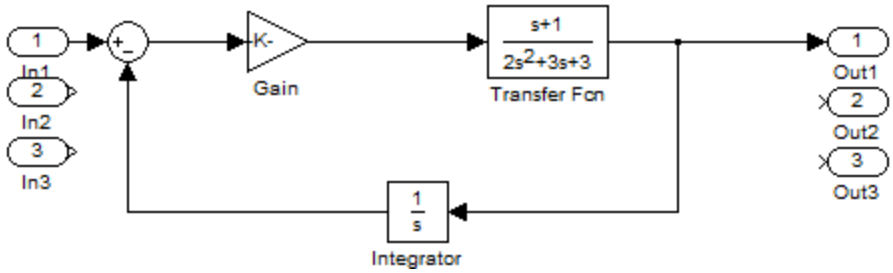


این شکل نمایش پاسخ معادله دیفرانسیل با شرایط اولیه ذکر شده می باشد . در پنجره Scopeبر روی گزینه Autoscaleکلیک کنید تا مقادیر پاسخ در شکل به طور دقیق نمایش داده شود:

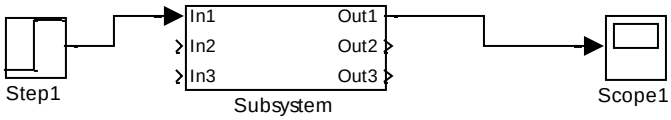


میتوانیم کل سیستم طراحی شده را در یک subsystem قرار دهیم جهت این کار از ....Commonly Subsystem را به Model اضاف میکنیم و جهت این کار ابتدا مدل طراحی شده(بدون ورودی و اسکوپ) را انتخاب میکنیم و با Copy کردن آن و دو بار کلیک بر روی Subsystem سپس Paste کردن آن در subsystem از شلوغی شکل بکاهیم میتوانیم در subsystem هر تعداد ورودی و خروجی داشته باشیم

نکته قابل ذکر این است که باید Scope را به خروجی ای که در subsystem تعریف کرده ایم و ورودی پله را به ورودی مربوطه بدهیم به عنوان مثال اگر مدل را به ورودی یک وصل کردیم باید ورودی پله را نیز به یک وصل کنیم.



و در نهایت شکل ما میتواند به صورت زیر خلاصه شده باشد که مدل اصلی با دوبار کلیک کردن بر روی subsystem قابل مشاهده است.



واضح است برای مشاهده خروجی مانند قبل عمل میکنیم.

گرد آوری و تهیه:

علی اسکندری

هرگونه کپی برداری از مطالب ممنوع میباشد!!!