بسم الله الرحمن الرحيم

مبانـــی کد نویسی ریاضی در نرم‌افـــزار

MATLAB

مهندس مهران غریب

سر شناسه : غریب، مهران، ۱۳۶۶ -
عنوان و نام پدیدآور : مبانی کدنویسی ریاضی در نرم‌افزار متلب/ تالیف مهران غریب.
مشخصات نشر : تبریز: اختر، ۱۳۹۴.
مشخصات ظاهری : ۱۶۰ ص.: مصور.
شابک : ۰-۶۷۹-۵۱۷-۹۶۴-۹۷۸
وضعیت فهرست نویسی: فیپا
موضوع : متلب
موضوع : ریاضیات مهندسی -- داده‌پردازی
موضوع : آنالیز عددی --برنامه‌های کامپیوتری
موضوع : حساب عددی -- برنامه‌های کامپیوتری
رده بندی کنگره : ۱۳۹۴ ۲م۸م/TA۳۴۵
رده بندی دیویی : ۶۲۰/۰۰۱۵۱
شماره کتابشناسی ملی : ۴۰۴۶۸۸۹

نشر اختر

مبانی کدنویسی ریاضی در نرم افزار متلب

مهران غریب

طراحی جلد: مهندس نیما مرغوب
سرمایه‌گذار: مؤسسه آموزش عالی صنعتی مراغه
ناشر: نشر اختر/ چاپ سبز/صحافی امین

چاپ اول ۱۳۹۴ / ۱۶۰ صفحه/ قطع وزیری/ ۵۰۰ جلد
شابک: ۰-۶۷۹-۵۱۷-۹۶۴-۹۷۸

مرکزفروش : تبریز- اول خیابان طالقانی، روبروی مصلی، نشر اختر
تلفن: ۰۹۱۴۱۱۶۶۸۹۷ و ۳۵۵۵۵۳۹۳-۰۴۱

قیمت : ۸۵۰۰ تومان

احتراماً بدین کلام پیام تقدیر و تشکر از ریاست محترم موسسه ی آموزش عالی صنعتی مراغه

جناب آقای مهندس سید حمید بطحائی

که در تمامی مراحل تألیف و نشر این کتاب حامی اینجانب بوده اند را می رسانم .

مقدمه

امروزه استفاده از نرم‌افزارهای مختلف در راستای تحلیل و محاسبات مسائل و در زمینه‌های مختلف علمی مخصوصاً زمینه‌ی ریاضیـــات و علوم مهندسی، امری اجتناب‌ناپذیر و لازم می‌باشد. بدیهی است که در سطوح مختلف علوم ریاضی و مهندسی به دلیل پیچیدگی مسائل و همچنین امکان بالای خطای انسانی در تحلیل و محاسبات، نیازمند روشی مطمئن و منظم برای رسیدن به نقطه‌ی هدف می‌باشد.

یکی از نرم‌افزارهای بسیار قوی و مطلوب که در رشته‌های ریاضی و علوم مهندسی به‌طور گسترده‌ای مورد استفاده قرار می‌گیـــرد، نرم‌افزار **MATLAB** است که توسط شرکـــت **Math Work** ارائه می‌گردد. نام این نرم‌افزار از عبارت **Matrix Laboratory** اقتباس شده است و از آن در حل مسائل مختلف و شبیه‌سازی استفاده می‌گردد که محدود نبودن این نرم‌افزار برای رشته‌ی خاص، باعث فراگیر بودن آن در اکثریت سطوح علمی مختلف بوده و کمتر کسی را می‌توان یافت که با نام آن آشنایی نداشته باشد.

پس بدیهی است که برای دانشجویان و اساتید رشته‌های مختلف، مخصوصا رشته‌هـای ریاضـی، فنی و مهندسی یادگیری این نرم‌افزار در اولویت باشد.

در این کتاب نیز به آموزش قدم به قدم کدنویسی مجموعه عملیات ریاضی و محاسباتی در این نرم‌افزار می‌پردازیم که پایه و اساس یادگیری نرم‌افزار **MATLAB** می‌باشد.

مهران غریب

mehran.gharib@yahoo.com

فهرست مطالب

عنوان **صفحه**

فصل اول: مفاهیم و دستورات کلی

۱-۱- معرفی نرم‌افزار MATLAB ۱۲

۱-۲- آشنایی با محیط نرم‌افزار MATLAB..... ۱۲

۱-۳- نحوه نوشت دستورات در پنجره Command Window ۱۶

۱-۴- پاک کردن دستورات در پنجره Command Window ۱۷

فصل دوم: دستورات و عملگرهای ریاضـــی

۲-۱- عملگرهای ریاضی مستقیم ۲۳

۲-۲- عملگرهای ریاضی دستوری ۲۴

۲-۳- نمادهای پرکاربرد در نرم‌افزار MATLAB..... ۳۲

فصل سوم: ماتریس‌ها

۳-۱- تعریف ماتریس در MATLAB ۳۴

۳-۲- عملیات ریاضی بر روی ماتریس‌ها و دستورات مربوط به آن..... ۳۹

فصل چهارم: بردارها

۴-۱- تعریف بردار در MATLAB ۶۸
۴-۲- ایجاد بردار با فواصل خطی ۷۲
۴-۳- ایجاد بردار با فواصل لگاریتمی ۷۲

فصل پنجم: توابع و چند جمله‌ای‌ها

۵-۱- تعریف توابع و چند جمله‌ای‌ها در MATLAB ۷۵
۵-۲- مقدار دهی به متغیرها در توابع و چند جمله‌ای‌ها ۷۸

فصل ششم: مشتـــــق

۶-۱- مشتق در MATLAB ۸۱
۶-۲- مشتق مرتبه n ام ۸۳
۶-۳- مشتق نسبت به متغیر مورد نظر ۸۴

فصل هفتم: انتگرال

۷-۱- انتگرال‌گیری در MATLAB ۸۹
۷-۲- انتگرال‌های معین ۹۱
۷-۳- انتگرال توابع چند متغیره ۹۲
۷-۴- انتگرال‌های چند گانه ۹۳
۷-۵- انتگرال‌های چند گانه معین ۹۴

فصل هشتم: توابع حدی

۸-۱- حد در MATLA ۹۸

۸-۲- توابع حدی با حدود چپ و راست ۹۹

فصل نهم: لاپلاس

۹-۱- تبدیل لاپلاس در MATLAB ۱۰۳

۹-۲- عکس تبدیل لاپلاس در MATLAB ۱۰۵

فصل دهم: اعداد مختلط

۱۰-۱- مختصات دکارتی ۱۰۸

۱۰-۲- مختصات قطبی..... ۱۱۰

فصل یازدهم: معادلات چند مجهولی

۱۱-۱- حل معادلات چند مجهولی..... ۱۱۴

فصل دوازدهم: M - File

۱۲-۱- تعریف M-File..... ۱۱۹

۱۲-۲- ایجاد یک M-File در نرم‌افزار MATLAB..... ۱۱۹

فصل سیزدهم: ترسیم گرافیکی توابع

۱۳-۱- ترسیم گرافیکی توابع ۱۳۱

۱۳-۲- نحوه نمایش نمودار با شیوه‌های متفاوت..... ۱۴۹

۱۳-۳- ترسیم سه بعدی توابع در MATLAB..... ۱۵۷

فصل اول:
مفاهیم و دستورات کلی

- معرفی نرم‌افزار MATLAB
- آشنایی با محیط نرم‌افزار MATLAB
- نحوه‌ی نوشتن دستورات در پنجره Command Window
- پاک کردن دستورات در پنجره Command Window

۱-۱- معرفی نرم‌افزار MATLAB

نرم‌افزار متلب، نرم‌افزار جامعی برای رشته‌های ریاضی، فیزیک، مهندسی و حتی سایر رشته ها می‌باشد که شامل دستورات و توابع مختلفی بوده؛ و به دو روش کلی می‌توان از آن استفاده کرد:

الف) کد نویسی

ب) سیمیولینک یا همان شبیه سازی

که در این کتاب با کدنویسی در محیط نرم‌افزار **MATLAB** آشنا خواهیم شد.

۱-۲- آشنایی با محیط نرم افزار MATLAB

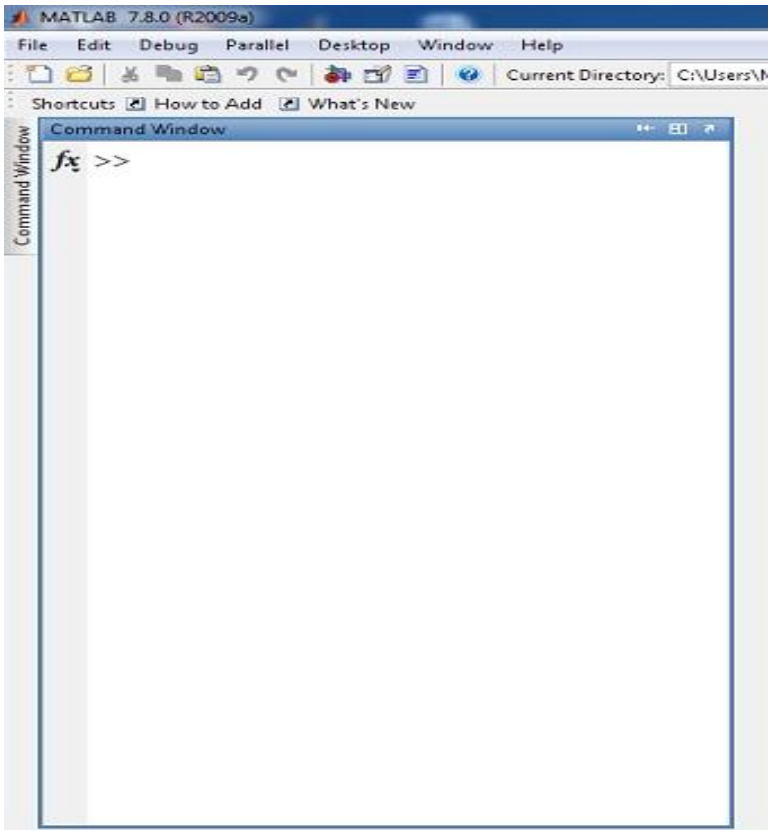
نرم‌افزار متلب دارای پنجره‌های مختلفی بوده که در این نوشتـــــه به معرفی و توصیف کاربردهای آن اشاره خواهیم کرد.

پنجره Command Window

پنجره **Command Window** محیطی است که در آن، ما می‌توانیم دستورات لازم برای اجرای برنامه خود را بنویسیم.

بعد از نوشتن هر دستور و فشردن کلید **Enter** از صفحه کلید کامپیوتر، دستور مورد نظر اجرا خواهد شد.

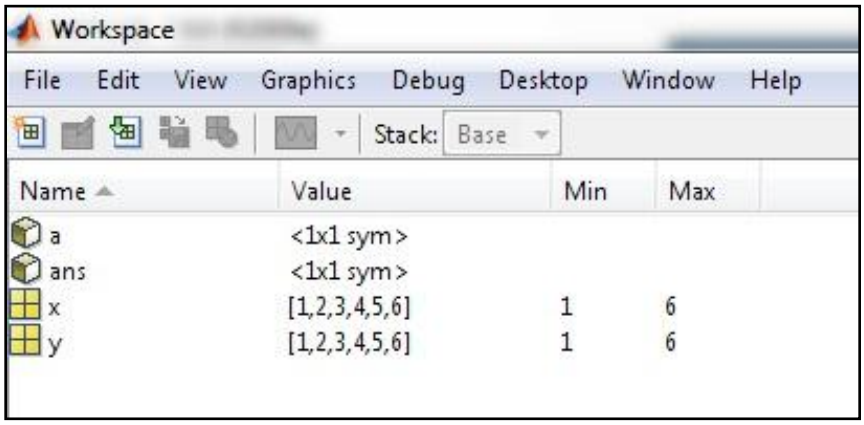
شکل زیر، تصویر پنجره Command Window در نرم‌افزار متلب را نشان می‌دهد.



پنجره Work Space

پنجره **Work Space** محیطی است که در آن لیستی از متغیرهای تعریف شده به وسیله‌ی دستورات متلب، نمایش داده می‌شوند.

شکل زیر، تصویر پنجره‌ی Work Space در نرم‌افزار متلب را نشان می‌دهد.



پنجره Current Directory

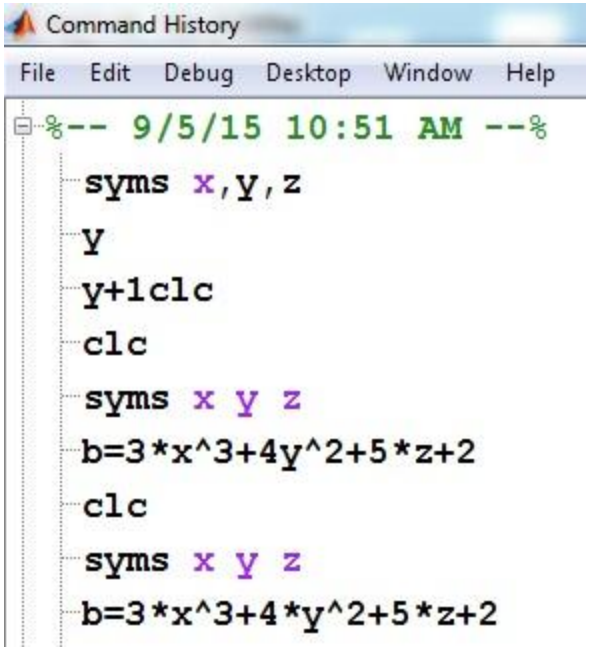
در این پنجره این امکان وجود دارد که فایل‌های متلب موجود و مورد نظر را در آن پنجره مشاهده کرده و به راحتی به فایل‌ها دسترسی پیدا کنیم.

شکل زیر، تصویر پنجره‌ی Current Directory در نرم‌افزار متلب را نشان می‌دهد.

پنجره Command History

در این پنجره، لیستی از دستوراتی که قبلا اجرا کرده‌ایم را می‌توانیم مشاهده کنیم و همچنین با دابل کلیک کردن بر روی آن دستور، مجددا می‌توانیم آن دستور را به محیط برنامه نویسی جدید بازخوانی کنیم.

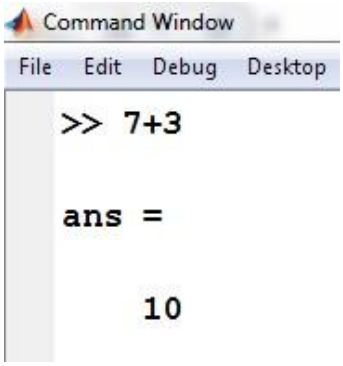
شکل زیر ، تصویر پنجره‌ی Command History در نرم‌افزار متلب را نشان می‌دهد.



۱-۳- نحوه نوشت دستورات در پنجره Command Window

برای شروع ابتدا باید پنجره **Command Window** را باز کرد، سپس با نوشتن دستورات مورد نظر و فشردن کلید **Enter** دستور مورد نظر اجرا خواهد شد.

مثال:

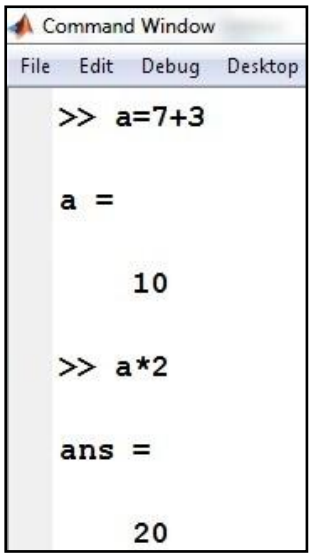


منظور از عبارت **ans**، مخفف کلمه **answer** به معنای پاسخ است.

⇦ نکته:

بهتر است، تمامی توابع و دستورات نوشته شده را نام‌گذاری کنیم، چرا که به هنگام استفاده مجدد از آن توابع می‌شود به راحتی با وارد کردن نام آن، دستور جدید را در مورد تابع مورد نظر اجرا کرد در حالی که اگر توابع را نام‌گذاری نکنیم برای استفاده مجدد، مجبور به نوشتن دوباره آن تابع خواهیم بود.

مثال:



اگر دقت کنیم، در این مثال تابعی با نام **a** ایجاد کردیم و برای اعمال عمل ضرب بر روی این تابع، به جای نوشتن کل عبارت **۳+۷**، از نام تابع یعنی **a** استفاده کردیم که باعث راحتی کار در عملیات کد نویسی گردید، البته اهمیت این موضوع در توابع و کدهای پیچیده بیشتر مشهود خواهد بود.

۱-۴- پاک کردن دستورات در پنجره Command Window

برای پاک کردن دستـــورات در محیط **Command Window** از چند دستـــور استفاده می‌کنیم.

✓ **دستور clc**

این دستور تمام دستورات نمایش داده شده در صفحـــه‌ی **Command Window** را پاک می‌کند.

باید توجه کرد که به هنگام استفاده از دستور **clc**، متغیرهای تعریف شده همچنان در پنجره **Command Window** باقی خواهند ماند و آن‌ها تنها از صفحه نمایش پاک خواهند شد، پس برای پاک کردن کلی دستورات و متغیرها باید از دستورهای دیگری استفاده کرد.

✓ **دستور clear**

این دستور علاوه بر صفحه‌ی **Command Window** تمامی متغیرهای تعریف شده را نیز پاک خواهد کرد.

✓ **دستور clear all**

این دستور نیز همانند دستور **clear** عمل خواهد کرد.

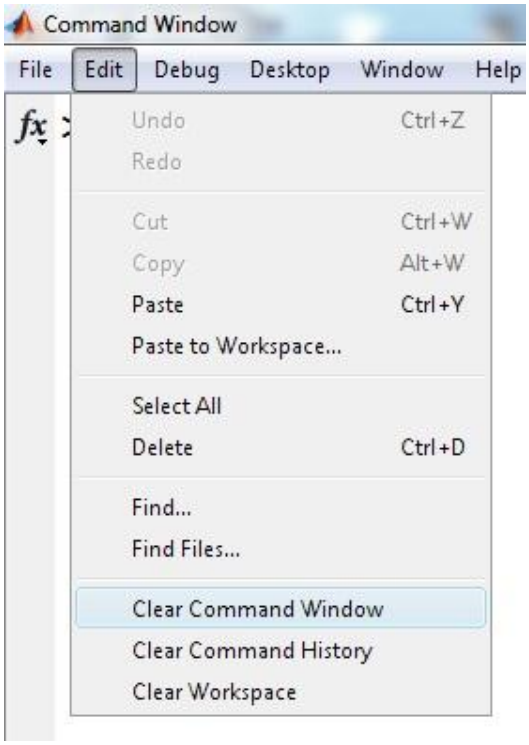
✓ **دستور clear x y**

این دستور متغیرهای **x** و **y** (یا هر متغیر تعریف شده دیگر) را پاک خواهد کرد.

⇦ نکته:

همیشه برای اینکه متغیرهای از قبل تعریف شده، در کد نویسی ما اخلالی ایجاد نکند بهتر است قبل از شروع به نوشتن کد، از دستور **clear** یا **clear all** استفاده گردد. یک کد نویس حرفه‌ای قطعا قبل از نوشتن دستورات و کدهای مورد نظر خود، یکی از دو دستور بیان شده را در ابتدای صفحه خواهد نوشت.

همچنین می‌توان برای پاک کردن صفحه **Command Window** از روش دیگری استفاده کرد. در این روش ابتدا باید روی گزینه **Edit** کلیک کرده و سپس گزینه **Clear Command Window** را انتخاب نمائیم. (طبق شکل زیر)

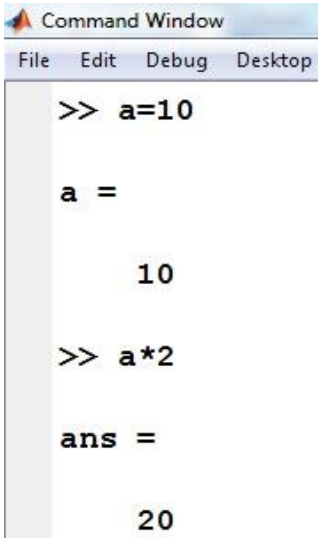


⇦ نکته:

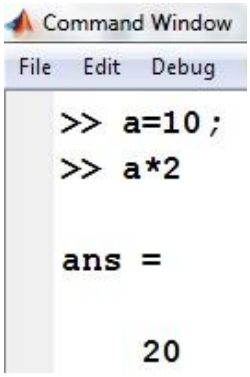
اگر در آخر دستور از **; (سمی‌کالن)** استفاده نمائیم، از نمایش نتیجه‌ی دستور در صفحه **Command Window** جلوگیری می‌کند، البته این به معنای عدم اجرای دستور نیست، بلکه دستور اجرا می‌گردد اما نتیجه دستور در صفحه‌ی **Command Window** نشان داده نمی‌شود که این عمل باعث کاهش سطرهای کد نویسی شده و امکان بروز خطا در هنگام کد نویسی را کاهش می‌دهد.

در مثال زیر، دو دستور را مقایسه نمائید:

مثال:



مثال:



پر واضح است که در مثال اول، به خاطر عدم استفاده از **;** در انتهای دستور، نتیجه **a=10** در صفحه‌ی **Command Window** نمـایش داده شـده و باعـث افـزایش بـی مـورد سـطرهای کدنویسی شده است اما در مثال دوم به دلیل استفاده از **;** در انتهای دستور، از نمایش نتیجه **a=10** جلوگیری شده و باعث کاهش سطرهای کد نویسی گردیده است.

فصل دوم:
دستورات و عملگرهای ریاضـــی

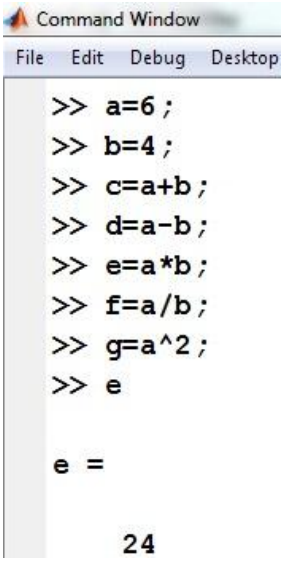
- عملگرهای ریاضی مستقیم
- عملگرهای ریاضی دستوری
- نمادهای پر کاربرد در نرم‌افزار متلب

۲-۱- عملگرهای ریاضی مستقیم

این عملگرها که در فضای کدنویسی متلب کاربرد زیادی دارند به راحتی و با استفاده از کیبورد کامپیوتر و بدون استفاده از دستور خاصی مورد استفاده قرار می‌گیرند، که آن‌ها در اصل همان چهار عملگر اصلی ریاضی به‌علاوه تعدادی از عملگرهای محدود می‌باشند.

جمع	+
تفریق	**-**
ضرب	*
تقسیم	/
توان	^

مثال:



⇦ نکته:

ما با گذاشتن علامت **; (سمی‌کالن)** در مقابل هر دستور از نمایش نتیجه آن جلوگیری کردیم و فقط یک نمونه از جواب‌ها را که همان قسمت **e** می‌باشد نمایش دادیم، در حالت کلی عملیات چهارگانه‌ی ریاضی بعلاوه عمل توان به صورت مثال بالا، نوشته می‌شود.

۲-۲- عملگرهای ریاضی دستوری

در این عملگرها باید برای عمل کردن عملگر، آن‌ها را به صورت دستور نوشت. در ادامه به تعدادی از دستورات مهم اشاره می‌گردد.

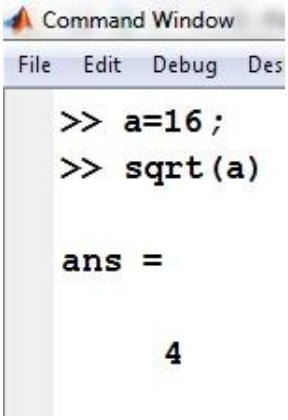
دستورات مهم :

✓ **دستور sqrt(a)**

این دستور، از عبارت **a** با فرجه **۲** جذر می‌گیرد. به عبارتی همان کار **رادیکال** ساده را انجام می‌دهد.

البته مجددا ذکر می‌گردد که **a** تنها یک نام برای عبارت مورد هدف ما است و می‌توان از هر حرف دیگری به جای آن استفاده کرد، که در این نوشته ما برای نام‌گذاری عبارات از حرف **a** استفاده می‌کنیم.

مثال:

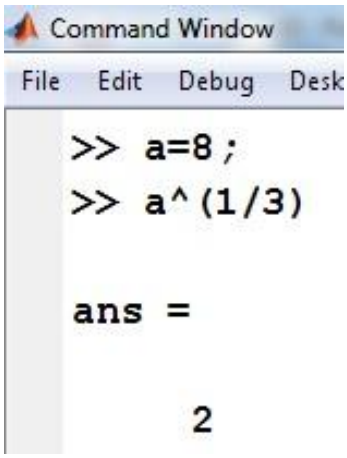


⇦ نکته:

دستور **sqrt** جذر عدد مورد نظر را با فرجه **۲** محاسبه می‌کند، اگر در کد نویسی ما به جذری نیاز داشته باشیم که فرجه آن به غیر از **۲** باشد، دیگر نمی‌توان از دستور **sqrt** استفاده کرد، برای به دست آوردن جذر عدد با فرجه به غیر از **۲** باید از طریق عمل توان محاسبه کرد.

این نکته ریاضی را می‌دانیم که جذر هر عدد با فرجه **x** ، برابر است با آن عدد به توان **1/x**، برای مثال جذر عدد **۱۶** با فرجه **۲** برابر است با **۱۶** به توان **۱/۲**، پس برای به دست آوردن جذر با فرجه بیش از **۲** از همین روش استفاده می‌کنیم. مجدداً برای مثال جذر عدد **۸** با فرجه **۳** برابر خواهد بود، **۸** به توان **۱/۳** در کد نویسی متلب هم برای بدست آوردن جذر با فرجه به غیر از **۲** از همین روش استفاده خواهیم کرد و دستور **sqrt** دیگر کاربردی نخواهد داشت.

مثال:



✓ **دستور log(a)**

این دستور , **Ln(a)** را محاسبه خواهد کرد.

✓ **دستور $log_{10}(a)$**

این دستور نیز، لگاریتم **a** در مبنای **۱۰** را محاسبه خواهد کرد.

مثال:

```
Command Window
File  Edit  Debug  Desktop
>> a=100;
>> b=log(a)

b =

    4.6052

>> c=log10(a)

c =

     2
```

همانطور که در مثال می‌بینیم، با استفاده از دو دستور **log** و **log_{10}**، در قسمت **b** **ln(100)** و در قسمت **c** **log_{10}(100)** را به راحتی محاسبه کردیم.

✓ **دستور round(a)**

این دستور، عدد اعشاری مورد نظر را به نزدیک‌ترین عدد صحیح گرد می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop
>> a=2.9;
>> b=2.3;
>> round(a)

ans =

     3

>> round(b)

ans =

     2
```

✓ دستور fix(a)

این دستور، عدد اعشاری مورد نظر را به سمت صفر گرد می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop
>> a=3.9;
>> b=-4.9;
>> fix(a)

ans =

     3

>> fix(b)

ans =

    -4
```

✓ **دستور ceil(a)**

این دستور، عدد اعشاری را به سمت مثبت بی‌نهایت گرد می‌کند.

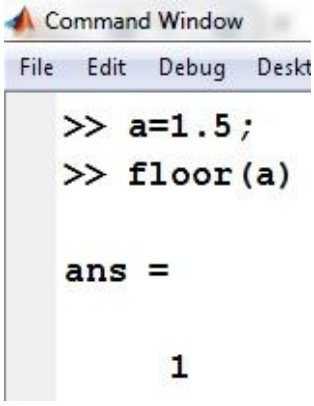
مثال:

```
Command Window
File  Edit  Debug  Des
>> a=1.5;
>> ceil(a)

ans =

     2
```

✓ دستور floor(a)

این دستور عدد اعشاری را به سمت منفی بی‌نهایت گرد می‌کند.

مثال:



✓ دستور format long

در نرم افزار متلب، به صورت پیش فرض اعداد اعشاری تا **۴** رقم اعشار نمایش داده می‌شوند با استفاده از دستور فوق می‌توان ارقام اعداد اعشاری را تا **۱۵** رقم نشان داد.

✓ دستور format short

با استفاده از این دستور، تعداد ارقام اعشاری مجددا تا **۴** رقم نمایش داده خواهد شد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> pi

ans =

    3.1416

>> format long
>> pi

ans =

   3.141592653589793

>> format short
>> pi

ans =

    3.1416
```

⇦ نکته:

عبارت **pi** در فضای نرم‌افزار متلب نشان دهنده‌ی عدد π می‌باشد.

۲-۳- نمادهای پر کاربرد در نرم افزار MATLAB

pi	**عدد پی که برابر است با ۳.۱۴**
inf	**بی‌نهایت**
i	**نشان دهنده‌ی قسمت موهومی عدد مختلط**

مثال:

فصل سوم:
ماتریس‌ها

- تعریف ماتریس در MATLAB
- عملیات ریاضی بر روی ماتریس‌ها و دستورات مربوط به آن

ماتریس‌ها یکی از بخش‌های مهم ریاضی بوده که کاربرد زیادی در مباحث ریاضی و همچنین مباحث فنی، مهندسی دارند. از آنجائی که عملیات ریاضی بر روی ماتریس‌هایی با سطر و ستون زیاد، کار بسیار مشکلی می‌باشد، استفاده از نرم‌افزار متلب می‌تواند جایگزین بسیار مناسبی برای محاسبه و کاربرد ماتریس‌ها در حل مسائل گردد.

لذا در این بخش سعی داریم که نحوه تعریف و ایجاد ماتریس در نرم‌افزار **MATLAB** و کلیه اعمال ریاضی پرکاربرد بر روی ماتریس‌ها را، قدم به قدم آموزش دهیم.

۳-۱- تعریف ماتریس در MATLAB

برای تعریف و یا به عبارتی بهتر ایجاد ماتریس در نرم‌افزار متلب به صورت زیر عمل می‌کنیم:

۱: ابتدا ماتریس را نام‌گذاری می‌کنیم، مانند **a**

۲: سپس درایه‌های ماتریس را مانند زیر می‌نویسیم:

مثال:

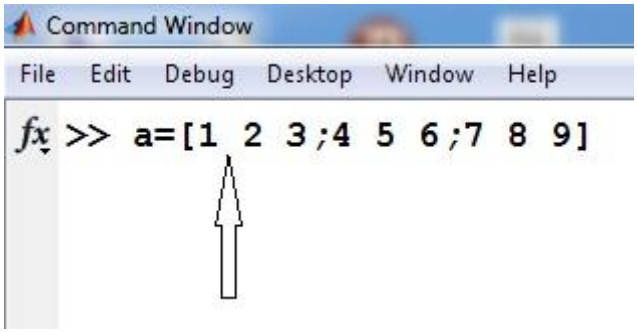
```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[1 2 3;4 5 6;7 8 9]

a =

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

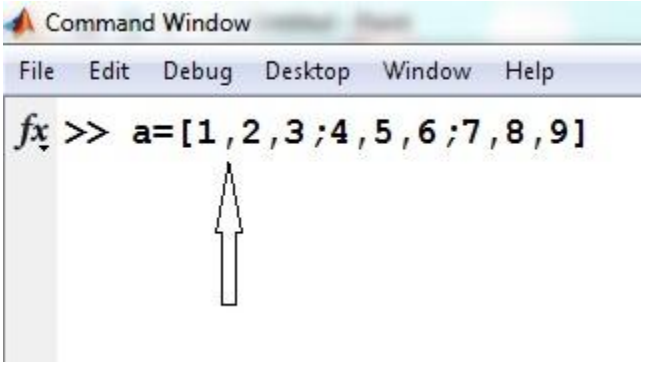
***در نوشتن یک ماتریس چند نکته را باید در نظر گرفت:**

نکته اول: کل ماتریس باید بین **[ ]** قرار گیرد.

```
Command Window
File  Edit  Debug  Desktop  Window  Help
fx >> a=[1 2 3;4 5 6;7 8 9]
```

نکته دوم: برای جدا کردن درایه‌های هر سطر از یک **space** یا **,** (ویرگول) استفاده می‌گردد.

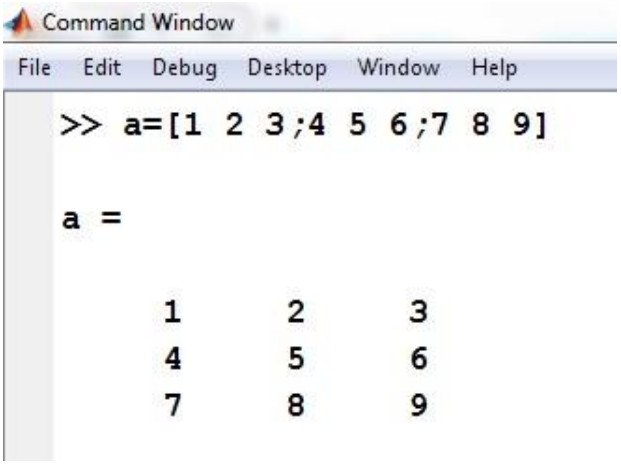




در هر دو حالت، خروجی ماتریس یکسان خواهد بود و استفاده از **Space** یا **,** برای جدا کردن درایه‌ها تنها بستگی به سلیقه شخصی نویسنده ماتریس خواهد داشت.

نکته سوم: برای جدا کردن سطرها از یک **;** استفاده می‌گردد.

و در نهایت ماتریس مورد نظر شکل خواهد گرفت.



ماتریس مثال زده شده یک ماتریس **۳*۳** بود که می‌توان به راحتی هر نوع ماتریس دیگری را ایجاد کرد، مانند ماتریس‌های زیر:

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window
>> b=[2 4;3 5;6 8]

b =

     2     4
     3     5
     6     8
```

در این مثال یک ماتریس **۲*۳** نوشته شده است.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Wind
>> c=[6;8;10;12]

c =

     6
     8
    10
    12
```

و در این مثال نیز یک ماتریس **۱*۴** نوشته شده است.

۳-۲- عملیات ریاضی بر روی ماتریس‌ها و دستورات مربوط به آن

✓ دستور a’

این دستور ترانهاده‌ی ماتریس **a** را محاسبه می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[1 4 7;1 8 9;2 10 5]

a =

     1     4     7
     1     8     9
     2    10     5

>> a'

ans =

     1     1     2
     4     8    10
     7     9     5
```

⇦ نکته:

دوباره ذکر می‌گردد که از نام **a** برای مثال‌ها استفاده شده است و کلمه **a** جزء دستور نیست، نام ماتریس هرچه که انتخاب گردد باید از آن نام برای نوشتن دستور استفاده گردد، یعنی اگر برای ماتریس نام **b** انتخاب کنیم در این صورت از کلمه **b** در دستور استفاده خواهیم کرد. پس از هر نامی که برای ماتریس مورد نظر استفاده کردیم، دستور را هم باید طبق آن نام اجرا کنیم.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> b=[1 2 3;4 5 6;7 8 9]

b =

     1     2     3
     4     5     6
     7     8     9

>> b'

ans =

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

✓ **دستور det(a)**

این دستور دترمینان ماتریس **a** را محاسبه می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> det(a)

ans =

   100
```

✓ دستور inv(a)

این دستور معکوس ماتریس **a** را محاسبه می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> inv(a)

ans =

   -0.5000    0.2000    0.2000
    0.2000   -0.2200    0.0800
    0.2000    0.0800   -0.1200
```

✓ دستور size(a)

این دستور سایز ماتریس **a** را نمایش می‌دهد.

مثال:

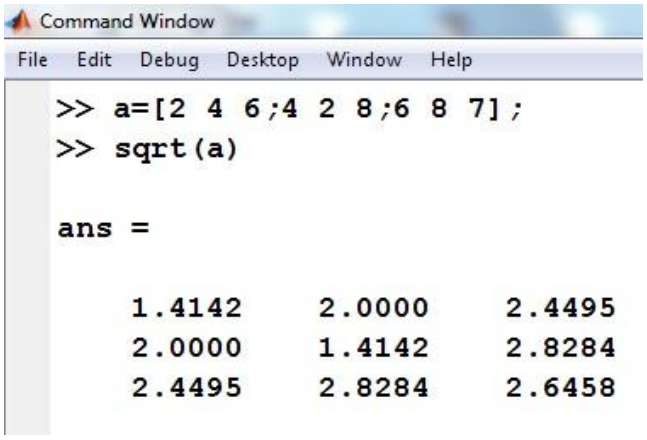
```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> size(a)

ans =

     3     3
```

همانطور که در مثال نشان داده شد، ماتریس **a** یک ماتریس **۳*۳** می‌باشد.

✓ دستور sqrt (a)

این دستور ریشه دوم ماتریس را محاسبه می‌کند. (به عبارتی از آرایه‌های ماتریس **a** جذر می‌گیرد).

مثال:



✓ دستور triu(a)

این دستور یک ماتریس بالا مثلثی از ماتریس **a** را ایجاد می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> triu(a)

ans =

     2     4     6
     0     2     8
     0     0     7
```

✓ **دستور tril(a)**

این دستور یک ماتریس پائین مثلثی از ماتریس **a** را ایجاد می‌کند.

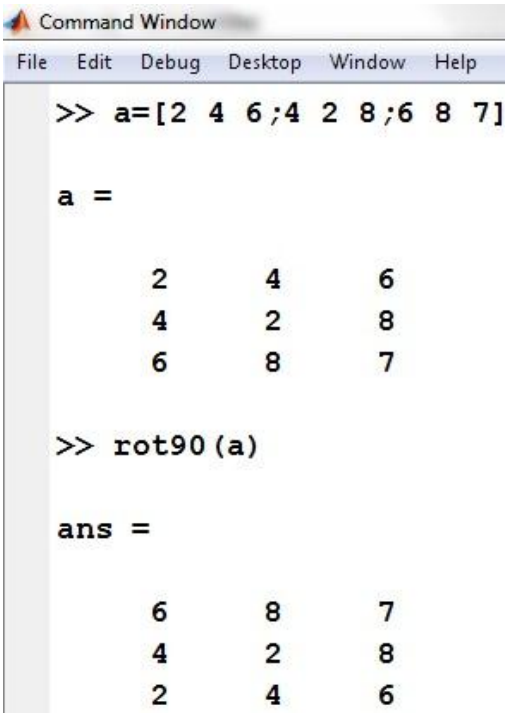
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> tril(a)

ans =

     2     0     0
     4     2     0
     6     8     7
```

✓ دستور rot90(a)

این دستور ماتریس **a** را **۹۰** درجه در جهت عقربه‌های ساعت چرخش می‌دهد.

مثال:



✓ دستور max(a)

این دستور، سطرهایی که مجموع آرایه‌های آن بیش‌ترین باشد را نمایش می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> max(a)

ans =

     6     8     8
```

✓ **دستور min(a)**

این دستور، سطرهایی که مجموع آرایه‌های آن کم‌ترین باشد را نمایش می‌دهد.

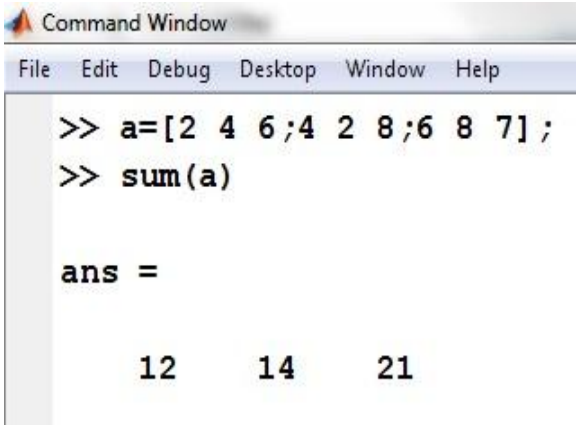
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> min(a)

ans =

     2     2     6
```

✓ دستور sum(a)

این دستور، آرایه‌های سطرها را باهم جمع می‌کند.

مثال:



✓ دستور sum(a,2)

این دستور، آرایه‌های ستون‌ها را باهم جمع می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> sum(a,2)

ans =

    12
    14
    21
```

✓ **دستور mean(a)**

این دستور، عمل میانگین‌گیری را بر روی سطرها انجام می‌دهد.

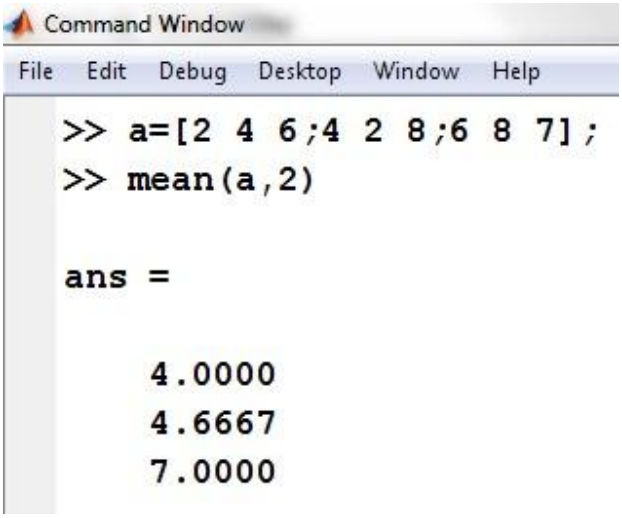
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> mean(a)

ans =

    4.0000    4.6667    7.0000
```

✓ دستور mean(a,2)

این دستور، عمل میانگین‌گیری را بر روی ستون‌ها انجام می‌دهد.

مثال:



✓ دستور sort(a)

این دستور، آرایه‌های ستون‌ها را به صورت صعودی مرتب می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> sort(a)

ans =

     2     2     6
     4     4     7
     6     8     8
```

✓ **دستور sort(a,2)**

این دستور، آرایه‌های سطرها را به صورت صعودی مرتب می‌کند.

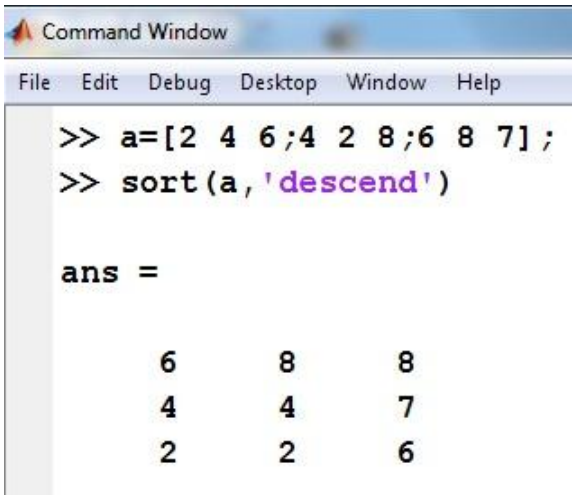
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> sort(a,2)

ans =

     2     4     6
     2     4     8
     6     7     8
```

✓ دستور sort(a,’descend’)

این دستور ستون‌ها را به صورت نزولی مرتب می‌کند.

مثال:



✓ دستور sort(a,2,’descend’)

این دستور سطرها را به صورت نزولی مرتب می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> sort(a,2,'descend')

ans =

     6     4     2
     8     4     2
     8     7     6
```

✓ **دستور trace(a)**

این دستور، مجموع عناصر قطر اصلی ماتریس را نمایش می‌دهد.

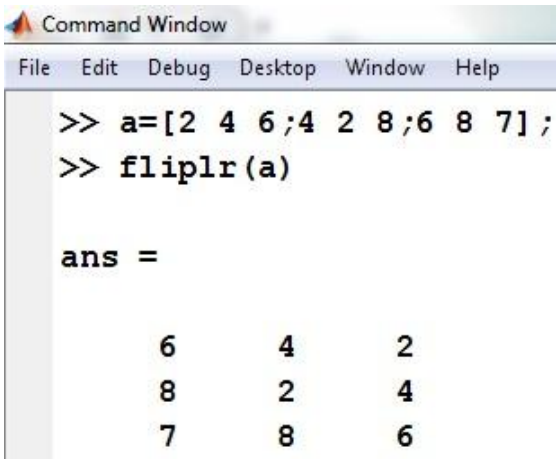
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> trace(a)

ans =

    11
```

✓ دستور fliplr(a)

این دستور چرخش ماتریس را از راست به چپ انجام می‌دهد.

مثال:



✓ دستور flipud(a)

این دستور چرخش ماتریس را از چپ به راست انجام می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> flipud(a)

ans =

     6     8     7
     4     2     8
     2     4     6
```

✓ **دستور diag(a)**

این دستور عناصر قطر اصلی ماتریس **a** را نمایش می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> diag(a)

ans =

     2
     2
     7
```

✓ دستور diag(fliplr(a))

این دستور عناصر قطر فرعی ماتریس **a** را نمایش می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> diag(fliplr(a))

ans =

     6
     2
     6
```

✓ دستور prod(a)

این دستور عمل ضرب را روی ستون‌ها انجام می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> prod(a)

ans =

    48    64   336
```

✓ دستور a(i,j)

این دستور آرایه‌ی سطر **i ام** و ستون **j ام** از ماتریس **a** را نمایش می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a(2,3)

ans =

     8
```

در این مثال دستور، آرایه مورد نظر از سطر دوم و ستون سوم را که برابر با **8** است نمایش داد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a(2,:)

ans =

     4     2     8
```

در این مثال نیز دستور نوشته شده سطر دوم از ماتریس را نشان می‌دهد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a(:,1)

ans =

     2
     4
     6
```

در این مثال نیز، دستور نوشته شده ستون اول ماتریس را نمایش می‌دهد.

با دقت در مثال‌ها می‌توان به روشنی دریافت که در دستور **a(i,j)** , **i** جهت نمایش سطر و **j** جهت نمایش ستون به کار می‌رود، پس در مثال دوم این دستور کاملا مشخص است که منظور دستور نمایش سطر دوم و در مثال سوم این دستور، منظور دستور نمایش ستون اول می‌باشد و برای عدم نمایش سطر یا ستون مورد نظر نیز از **: (دو نقطه)** استفاده شده است.

✓ دستور a(i,:)=[]

این دستور سطر **i ام** ماتریس **a** را حذف می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a(2,:)=[]

a =

     2     4     6
     6     8     7
```

این مثال به راحتی می‌توان متوجه شد که **سطر دوم** از ماتریس **a** حذف شده است.

مثال:

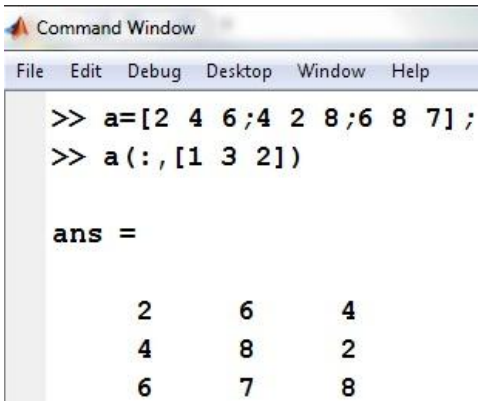
```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a(:,1)=[]

a =

     4     6
     2     8
     8     7
```

این مثال نیز به راحتی می‌توان متوجه شد که **ستون اول** از ماتریس **a** حذف شده است.

✓ **دستور a(:,[1 3 2])**

این دستور، جای **ستون دوم و سوم** ماتریس **a** را عوض می‌کند.

مثال:



با دقت در این مثال به راحتی می‌توان متوجه گردید که این دستور با سطرهای ماتریس **a** کاری ندارد چرا که با گذاشتن علامت **:** (**دو نقطه**) در قسمت **i ام** ماتریس (یعنی همان قسمت سطرها) نشان می‌دهیم هدف جابجایی ما در قسمت سطرها نبوده و با نوشتن **[1 3 2]** در قسمت **j ام** ماتریس (یعنی همان قسمت ستون‌ها)، جابجایی را در قسمت ستون‌ها انجام خواهیم داد، حال در مثال بعدی، جابجایی را در قسمت سطرها انجام می‌دهیم.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a([3 1 2],:)

ans =

     6     8     7
     2     4     6
     4     2     8
```

در مثال فوق نیز با قرار دادن **:** در قسمت **j ام** ماتریس، هیچ گونه جابجایی در ستون‌های این ماتریس انجام نشد ولی با نوشتن **[3 1 2]** در قسمت **i ام** ماتریس (یعنی همان قسمت سطرها) سطرهای ماتریس با توجه به نحوه چیدمان در دستور جابجا گردید.

- ✓ **دستور a*b**

در این دستور ماتریس **a** در ماتریس **b** ضرب خواهد شد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> b=[1 2 10;4 7 9;12 14 5];
>> a*b

ans =

    90   116    86
   108   134    98
   122   166   167
```

✓ دستور a.*b

در این دستور تک تک آرایه‌های ماتریس **a** در تک تک آرایه‌های ماتریس **b** ضرب خواهد شد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> b=[1 2 10;4 7 9;12 14 5];
>> a.*b

ans =

     2     8    60
    16    14    72
    72   112    35
```

با مقایسه دو دستور **a*b** و **a.*b** می‌توان تفاوت این دو دستور را به روشنی درک کرد که در دستور **a*b** دو ماتریس به همدیگر ضرب می‌گردند. اما در دستور **a.*b** آرایه‌های دو ماتریس به صورت تک به تک ضرب می‌گردند، لازم به ذکر است که این قاعده در مورد تقسیم نیز صدق می‌کند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> b=[1 2 10;4 7 9;12 14 5];
>> a/b

ans =

   -0.0103    0.7179   -0.0718
    2.1128   -1.8974    0.7897
    0.1949    0.3590    0.3641

>> a./b

ans =

    2.0000    2.0000    0.6000
    1.0000    0.2857    0.8889
    0.5000    0.5714    1.4000
```

⇦ نکته:

برای دو عملگر + و − نیازی به گذاشتن نقطه وجود ندارد، چرا که در نرم‌افزار متلب، عملگر به صورت پیش‌فرض این دو عمل ریاضی را بر روی تک‌تک آرایه‌های ماتریس اعمال می‌کند.

✓ دستور a^n

این دستور کل ماتریس **a** را به توان **n** می‌رساند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a^2

ans =

    56    64    86
    64    84    96
    86    96   149
```

✓ دستور a.^n

این دستور تک تک آرایه‌های ماتریس **a** را به توان **n** می‌رساند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=[2 4 6;4 2 8;6 8 7];
>> a.^2

ans =

     4    16    36
    16     4    64
    36    64    49
```

و اما در ادامه، دستوراتی را خواهیم آموخت که به کمک آن‌ها، می‌توان ماتریس‌هایی را بدون استفاده از دستور ماتریس نویسی ایجاد کرد.

✓ دستور eye(i,j)

این دستور ماتریس همانی را با **سطر i** و **ستون j** ایجاد خواهد کرد.

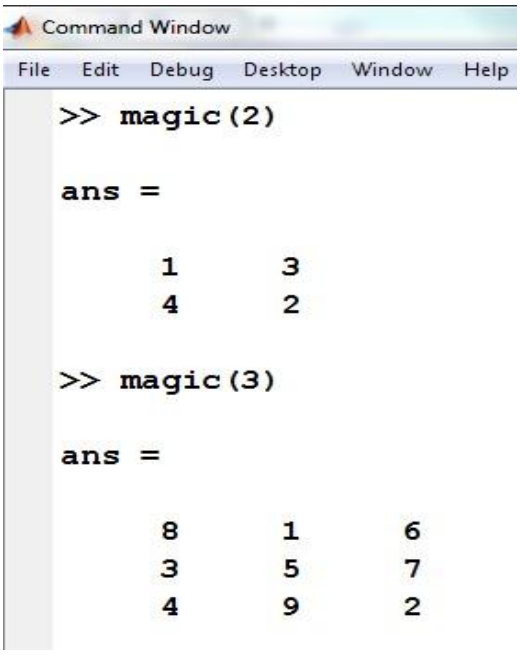
مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> eye(3,3)

ans =

     1     0     0
     0     1     0
     0     0     1

>> eye(4,4)

ans =

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

✓ دستور magic(n)

این دستور یک ماتریس **n*n** تصادفی ایجاد خواهد کرد که به ماتریس جادویی معروف است

مثال:



✓ دستور rand(n)

این دستور ماتریس تصادفی **n*n** ایجاد می‌کند که تمام آرایه‌های **0<n<1** باشد.

البته می‌توان با استفاده از دستور **rand(i,j)** یک ماتریس تصادفی با **سطر i** و **ستون j** با شرط فوق ساخت.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> rand(3)

ans =

    0.6948    0.0344    0.7655
    0.3171    0.4387    0.7952
    0.9502    0.3816    0.1869

>> rand(3,2)

ans =

    0.4898    0.7094
    0.4456    0.7547
    0.6463    0.2760
```

✓ **دستور ones(i,j)**

این دستور یک ماتریس **i*j** تولید می‌کند که همه‌ی آرایه‌های آن **یک** است.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> ones(3,4)

ans =

     1     1     1     1
     1     1     1     1
     1     1     1     1

>> ones(2,5)

ans =

     1     1     1     1     1
     1     1     1     1     1
```

✓ دستور zeros(i,j)

این دستور ماتریسی ایجاد می‌کند که تمام درایه‌های آن **صفر** باشد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> zeros(3,4)

ans =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

فصل چهارم:
بردار ها

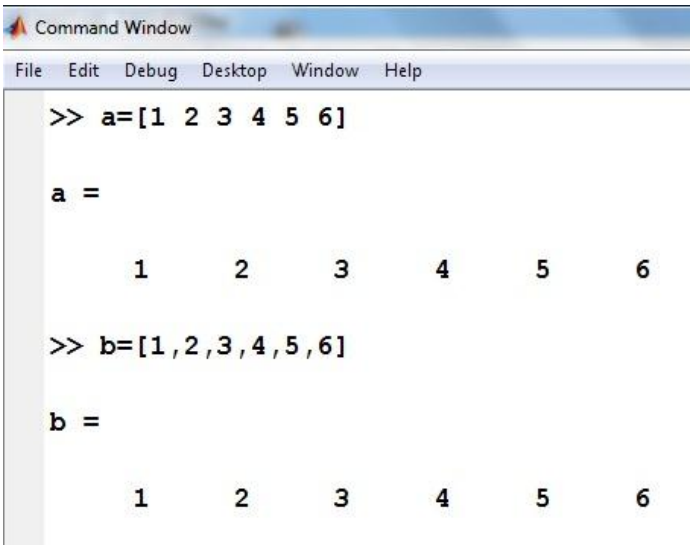
- تعریف بردار در MATLAB
- ایجاد بردار با فواصل خطی
- ایجاد بردار با فواصل لگاریتمی

۴-۱- تعریف بردار در MATLAB

بردار به تعدادی از اعداد اطلاق می‌گردد که پشت سر هم و به صورت زنجیروار آمده‌اند، هر کدام از این اعداد عنصر نام دارد و مانند ماتریس‌ها به دو صورت می‌توان آن‌ها را نوشت.

در روش اول استفاده از **Space** برای جدا کردن عناصر از هم و در روش دوم استفاده از **,** (ویرگول) برای جدا کردن عناصر از هم خواهد بود.

مثال:



در مثال فوق ما از هر دو روش برای ایجاد یک بردار استفاده کردیم.

بهتر است به این نکته نیز اشاره کنیم که برای نوشتن چنین برداری (برداری که عناصر آن به ترتیب) چیده شده‌اند از روش زیر هم می‌توان استفاده کرد.

مثال:

بردار **c** نیز یک بردار منظم مانند بردارهای **a** و **b** می‌باشد اما در این مثال برای کوتاه کردن دستور بردار نویسی، به این شکل نوشته شد و عناصر بردار را از عدد **۱** تا عدد **۶** به ترتیب قرار گرفت. در این دستور علامت **:** به معنای **تا** می‌باشد.

فرض کنید بخواهیم برداری بنویسیم که عناصر آن اعداد **۱** تا **۴۰۰** باشد، طبیعتاً نوشتن چنین برداری به روش **a** و **b** آسان نخواهد بود، ولی می‌توان با استفاده از روش **c** این بردار را به راحتی نوشت.

حال فرض می‌کنیم ما به برداری نیاز داریم که عناصر آن مانند بردارهای **a , b , c** به صورت افزایش یک واحدی نباشند، مثلا قصد داشته باشیم برداری بنویسیم که عناصر آن از **۲** شروع شده و با **۲** واحد افزایش در هر عنصر تا **۱۰** ادامه داشته باشد. در این گونه بردارها به روش زیر عمل می‌کنیم.

✓ **دستور d=[x:y:z]**

x : عنصر ابتدائی بردار

y : مقدار تغییر

z : عنصر انتهائی بردار

مثال:

در مثال فوق ما برداری را ایجاد کردیم که از عدد **۲** شروع شده و با افزایش **۲** واحدی در هر عنصر به عدد **۱۰** ختم می‌گردد

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> e=[0:1.5:6]

e =

         0    1.5000    3.0000    4.5000    6.0000
```

این بردار نیز با عدد **صفر** شروع شده و با افزایش **۱.۵** واحدی در هر عنصر به عدد **۶** ختم می‌گردد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> f=[6:-2:-4]

f =

     6     4     2     0    -2    -4
```

این بردار نیز با عدد **۶** شروع شده و با کاهش **۲** واحدی در هر عنصر به عـــدد **۴-** ختم می‌گردد.

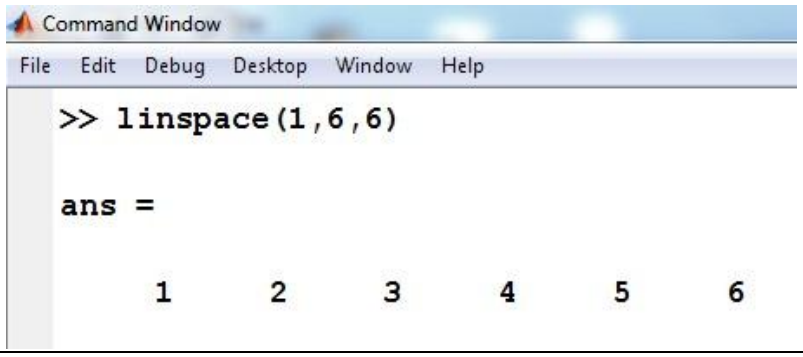
۴-۲- ایجاد بردار با فواصل خطی

برای ایجاد یک بردار با فواصل خطی از دستور زیر استفاده می‌گردد.

✓ **دستور linspace(a,b,c)**

در این دستور **a** ابتدای بردار، **b** انتهای بردار و **c** تعداد نقاط بردار است.

مثال:



۴-۳- ایجاد بردار با فواصل لگاریتمی

برای ایجاد یک بردار با فواصل لگاریتمی از دستور زیر استفاده می‌گردد.

✓ **دستور logspace(a,b,c)**

در این دستور**10^a** ابتدای بردار، **10^b** انتهای بردار و **c** تعداد نقاط بردار است.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> logspace(1,2,5)

ans =

   10.0000   17.7828   31.6228   56.2341  100.0000
```

در این مثال بردار لگاریتمی بین **10^1** و **10^2** پنج عدد انتخاب می‌گردد.

✓ **دستور randperm(n)**

این دستور برداری را ایجاد می‌کند که در آن، اعداد صحیح از **۱** تا **n** به صورت تصادفی قرار گرفته‌اند.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> a=randperm(7)

a =

     6     3     7     5     1     2     4
```

فصل پنجم:
توابع و چند جمله‌ای‌ها

- تعریف توابع و چند جمله‌ای‌ها در MATLAB
- مقدار دهی به متغیرها در توابع و چند جمله‌ای‌ها

۵-۱- تعریف توابع و چند جمله‌ای‌ها در MATLAB

شکل کلی توابع و چند جمله‌ای‌ها در توابع ریاضی به شکل زیر است

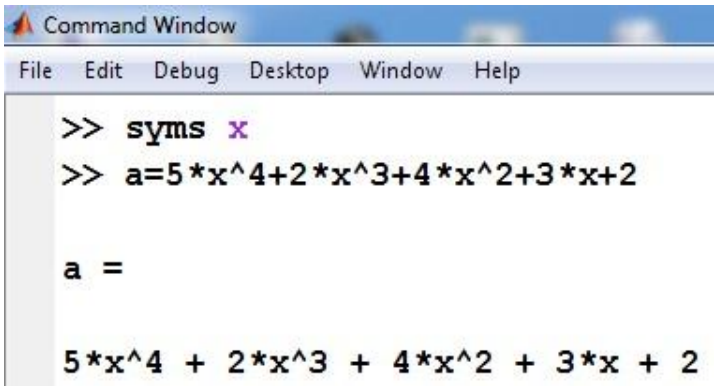
$$\boldsymbol{a = f(x) = ax^n + bx^{n-1} + \ \dots + dx + c.}$$

مثال:

$$\boldsymbol{.a = f(x) = 5x^4 + 2x^3 + 4x^2 + 3x + 2}$$

در این مثال، یک چند جمله‌ای درجه چهار نوشته شده است.

حال مثال فوق را در نرم‌افزار متلب می‌نویسیم.



***در نوشتن یک چند جمله‌ای باید به سه نکته دقت کرد.**

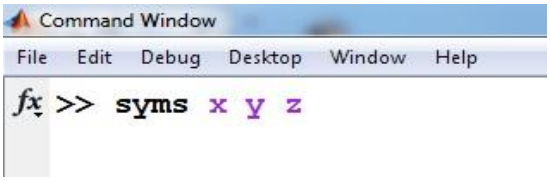
⇦ نکته اول:

✓ **دستور syms**

برای نوشتن چند جمله‌ای در اولین گام باید متغیر چند جمله‌ای را با استفاده از دستور **syms** تعریف کرد که در این مثال چند جمله‌ای ما دارای یک متغیر به نام **x** می‌باشد.

باید توجه کرد، در صورتی که چند جمله‌ای ما دارای چند متغیر باشد، باید همه متغیرها را با استفاده از دستور **syms** تعریف کرد، همچنین اشاره می‌گردد که ترتیب نوشتن متغیرها در دستور **syms** دارای اهمیت نیست و می‌توان با ترتیب دلخواه متغیرها را تعریف کرد و فقط این نکته حائز اهمیت است که باید ما بین متغیرها حتما از **space** استفاده گردد.

مثال:



⇦ نکته دوم:

عبارت ax باید به صورت $a * x$ نوشته گردد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window
>> syms x
>> 3*x

ans =

3*x
```

⇦ نکته سوم:

عبارت x^n باید به صورت x^n نوشته گردد.

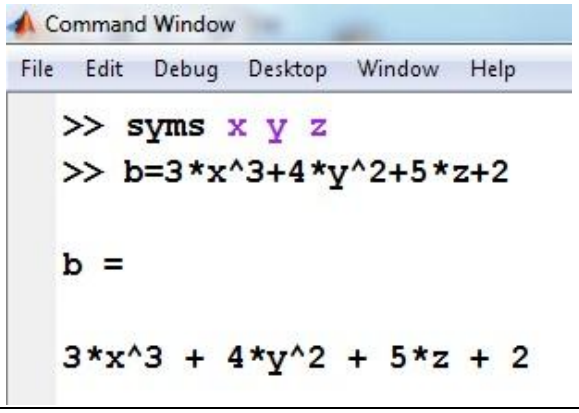
مثال:

```
Command Window
File  Edit  Debug  Desktop
>> syms x
>> x^3

ans =

x^3
```

اکنون با استفاده از نکات بیان شده چند جمله‌ای سه متغیره زیر را می‌نویسیم.

مثال:

$$.b = 3x^3 + 4y^2 + 5z + 2$$



۵-۲- مقدار دهی به متغیرها در توابع و چند جمله‌ای‌ها

✓ **دستور subs**

برای اینکه بتوان به هر کدام از متغیرهای موجود در یک تابع یا چند جمله‌ای مقدار داد، از دستور **subs** استفاده می‌گردد.

مثال:

برای مثال در تابع زیر، مقادیر **x ,y ,z** را جایگذاری می‌کنیم.

$$.a = 4x^3 + 3y^2 + 3z$$

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> syms x y z
>> a=4*x^3+3*y^2+3*z;
>> b=subs(a,x,4)

b =

3*y^2 + 3*z + 256

>> c=subs(a,y,6)

c =

4*x^3 + 3*z + 108

>> d=subs(a,z,2)

d =

4*x^3 + 3*y^2 + 6
```

با دقت در مثال فوق می‌توان به روشنی دریافت که با استفاده از دستور **subs**:

در دستور قسمت **b**: در تابع **a** به جای متغیر **x** مقدار **۴** را جایگذاری کرده‌ایم.

در دستور قسمت **c**: در تابع **a** به جای متغیر **y** مقدار **۶** را جایگذاری کرده‌ایم.

در دستور قسمت **d**: در تابع **a** به جای متغیر **z** مقدار **۲** را جایگذاری کرده‌ایم.

فصل ششم:
مشتـــق

- مشتق در **MATLAB**
- مشتق مرتبه‌ی **n** ام
- مشتق نسبت به متغیر مورد نظر

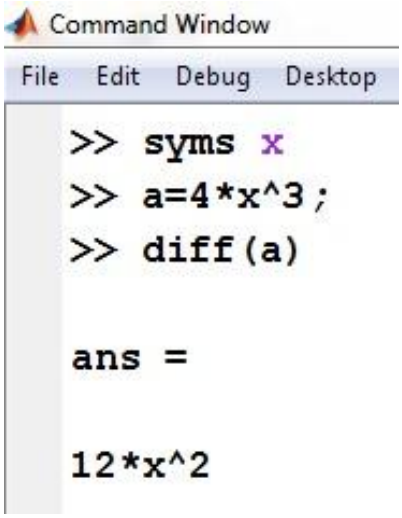
۶-۱- مشتق در MATLAB

✓ دستور diff(a)

از این دستور برای انجام عمل مشتق‌گیری در نرم‌افزار متلب استفاده می‌گردد.

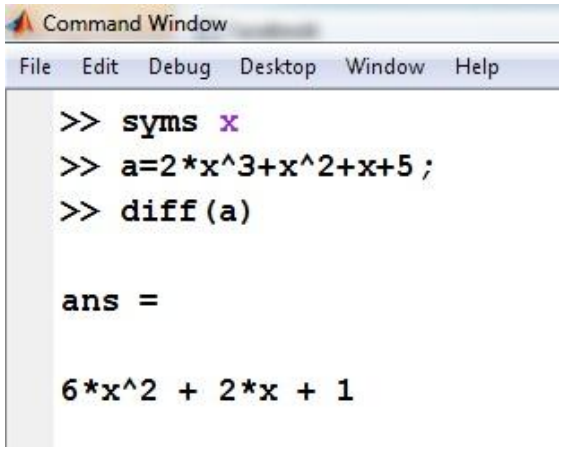
مثال:

$$.a = 4x^3$$



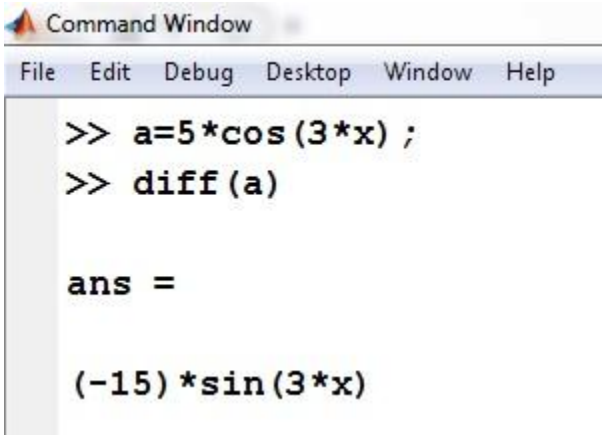
مثال:

$$.a = 2x^3 + x^2 + x + 5$$



مثال:

$$.a = 5cos(3x)$$



۶-۲- مشتق مرتبه n ام

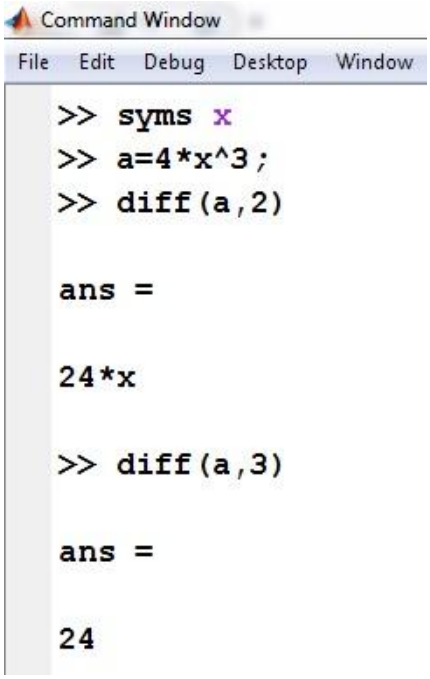
برای مشتق‌گیری در مراتب بالاتر از یک، از دستور زیر استفاده می‌کنیم.

✓ دستور diff(a,n)

این دستور از تابع **a** مشتق مرتبه‌ی **n ام** را محاسبه می‌کند.

مثال:

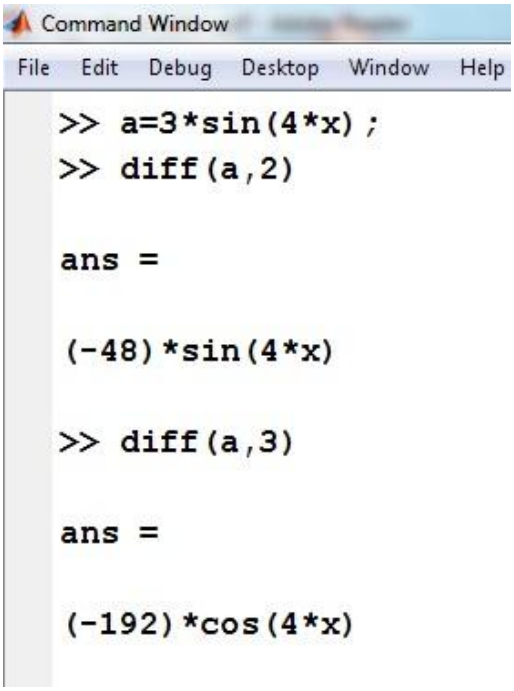
$$.a = 4x^3$$



در این مثال از تابع **a** یک‌بار **مشتق مرتبه دوم** و یک‌بار هم **مشتق مرتبه سوم** گرفته شده است.

مثال:

$$.a = 3\sin(4x)$$



۶-۳- مشتق نسبت به متغیر مورد نظر

در توابع چند جمله‌ای که دارای چند متغیر می‌باشند باید مشخص کرد که از کدام متغیر قصد انجام عمل مشتق‌گیری را داریم، لذا با استفاده از دستور زیر عمل می‌کنیم.

✓ دستور diff(a,x)

برای مثال در این دستور از تابع **a** نسبت به متغیر **x** عمل مشتق‌گیری انجام خواهد شد.

مثال:

$$.a = 3x^3 + 2y^2 + 3z^2 + 2$$

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> syms x y z
>> a=3*x^3+2*y^2+3*z^2+2;
>> diff(a,x)

ans =

9*x^2

>> diff(a,y)

ans =

4*y

>> diff(a,z)

ans =

6*z
```

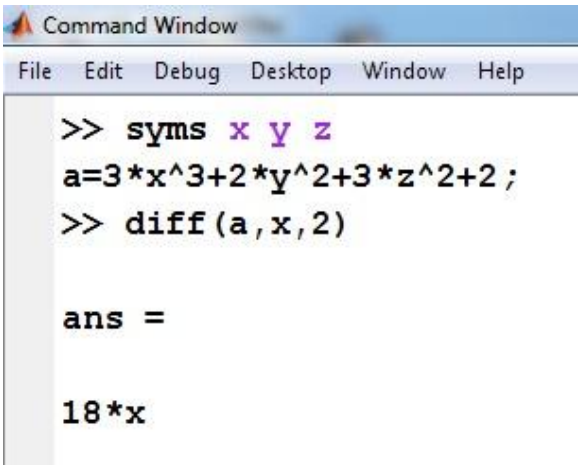
با دقت در مثال فوق می‌توان دریافت که، یک‌بار از تابع **a** نسبت به **x** و یک‌بار نسبت به **y** و یک‌بار هم نسبت به **z** عمل مشتق‌گیری انجام شده است.

اکنون قصد داریم از تابع **a** نسبت به یکی از متغیرها مشتق مرتبه‌ی **n ام** بگیریم.

برای مثال می‌خواهیم از تابع **a** نسبت به **x** مشتق مرتبه‌ی **دوم** بگیریم

مثال:

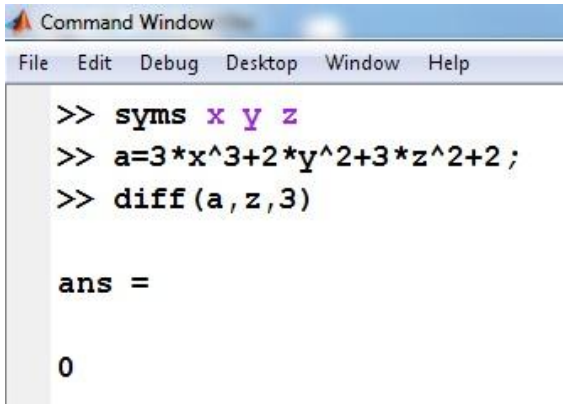
$$.a = 3x^3 + 2y^2 + 3z^2 + 2$$



در مثال بعدی، از تابع **a** نسبت به **z** مشتق مرتبه‌ی **سوم** را محاسبه می‌کنیم.

مثال:

$$.a = 3x^3 + 2y^2 + 3z^2 + 2$$



فصل هفتم:
انتگرال

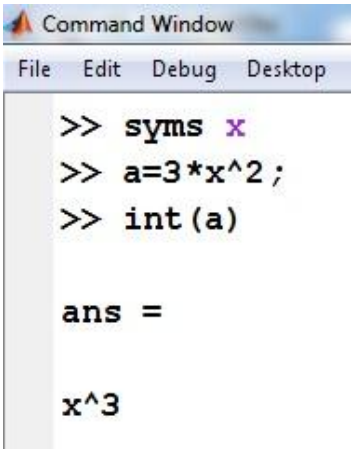
- انتگرال‌گیری در MATLAB
- انتگرال‌های معین
- انتگرال توابع چند متغیره
- انتگرال‌های چندگانه
- انتگرال‌های چندگانه معین

۷-۱- انتگرال‌گیری در MATLAB

✓ دستور int(a)

از این دستور برای انتگرال‌گیری از توابع در نرم‌افزار متلب استفاده می‌گردد.
برای مثال، می‌خواهیم انتگرال $\int 3x^2 dx$ را محاسبه نمائیم.

مثال:



مثال:

در این مثال نیز می‌خواهیم، انتگرال تابع زیر را محاسبه نمائیم.

$$.a = \int 4x^3 dx$$

```
>> syms x
>> a=4*x^3;
>> int(a)

ans =

x^4
```

مثال:

انتگرال تابع زیر را محاسبه می‌کنیم.

$$.a = \int cos\ 3x\ \ dx$$

```
>> a=cos(3*x);
>> int(a)

ans =

sin(3*x)/3
```

۷-۲- انتگرال‌های معین

مثال‌های حل شده‌ی قبلی در مبحث انتگرال‌گیری در مورد انتگرال‌های نامعین بود، اما در این قسمت سعی می‌کنیم بیاموزیم که چگونه می‌توان انتگرال‌های معین را نیز محاسبه کرد.

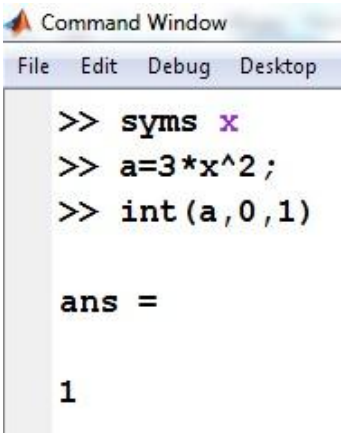
برای مثال انتگرال زیر را در نظر بگیرید:

$$.a = \int_0^1 3x^2 dx$$

در این انتگرال، حد بالا و پائین انتگرال مشخص گردیده است، به همین خاطر از آن به عنوان انتگرال معین نام می‌بریم.

اکنون، انتگرال زیر را در محیط **Command Window** نرم‌افزار متلب محاسبه می‌کنیم.

مثال:



با دقت در قسمت دستور **int(a,0,1)** متوجه خواهیم شد که، **a** تابع مورد نظر ما بوده و ۰ حد پائین انتگرال و **۱** حد بالای انتگرال می‌باشد، به عبارتی بهتر دستور نوشته شده، این مفهوم را می‌رساند که باید از تابع **a** با حدود ۰ تا **۱** عمل انتگرال‌گیری انجام گیرد.

۷-۳- انتگرال توابع چند متغیره

تابع زیر را در نظر بگیرید.

$$.a = f(x, y) = 3x^3 + 4y^2$$

اکنون قصد داریم از تابع بالا، یعنی همان تابع **a** نسبت به دو متغیر **x** و **y** به طور جداگانه انتگرال‌گیری کنیم.

مثال:

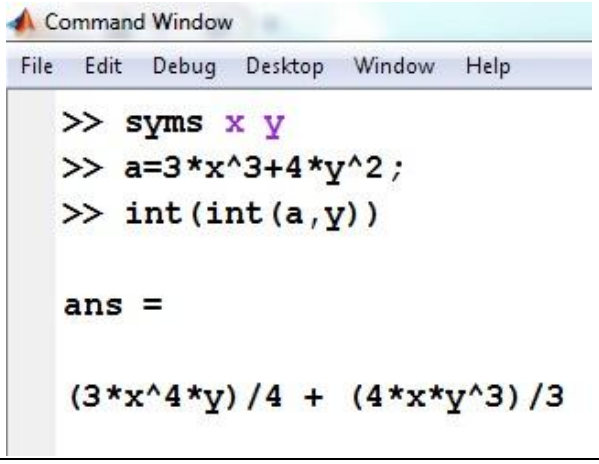
```
Command Window
File  Edit  Debug  Desktop  Window
>> syms x y
>> a=3*x^3+4*y^2;
>> int(a,x)

ans =

(3*x^4)/4 + 4*x*y^2

>> int(a,y)

ans =

3*x^3*y + (4*y^3)/3
```

۷-۴- انتگرال‌های چند گانه

برای محاسبه انتگرال‌های چندگانه باید از دستور **int** به صورت تو در تو استفاده کرد، برای مثال، انتگرال دوگانه زیر را از طریق دستور **int** به صورت تو در تو محاسبه می‌کنیم.

مثال:

$$\mathbf{a}=\iint (3x^3+4y^2)\,dxdy.$$



۷-۵- انتگرال‌های چند گانه معین

انتگرال زیر را در نظر بگیرید.

$$\boldsymbol{a = \int_0^1 \int_0^1 (3x^3 + 4y^2)\, dxdy.}$$

در این نوع از انتگرال‌ها هم مانند مثال بالا از دستور **int** تو در تو استفاده می‌کنیم با این تفاوت که دو حد بالا و پائین انتگرال را نیز برای محاسبه وارد دستور می‌کنیم.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> syms x y
>> a=3*x^3+4*y^2;
>> int(int(a,y,0,1),0,1)

ans =

25/12
```

با دقت در دستور نوشته شده در مثال، می‌توان مفهوم دستور را به روشنی درک کرد و آن عبارت است از:

گام اول دستور: از تابع **a** نسبت به **y** با حدود ۰ تا **۱** انتگرال گیری انجام گیرد.

گام دوم دستور: از نتیجه حاصل شده از گام اول، با حدود ۰ تا **۱** مجددا انتگرال‌گیری انجام گیرد.

⇦ نکته:

احتمال دارد این سوال پیش آید که چرا در گام دوم دستور، مشخص نگردید که از تابع نسبت به کدام متغیر باید عمل انتگرال گیری انجام گردد؟!

پاسخ بدین گونه است که در گام اول دستور با توجه به ترتیب انتگرال‌گیری مشخص گردید که از تابع **a** نسبت به **y** با حدود ذکر شده عمل انتگرال‌گیری انجام گیرد، پس بعد از انتگرال‌گیری با حدود ذکر شده در گام اول، متغیر **y** از بین رفته و تبدیل به عدد می‌گردد که در این صورت تنها متغیر باقی مانده در تابع فقط **x** خواهد بود که در این صورت نیازی به مشخص کردن متغیر برای نرم‌افزار متلب نبوده و نرم‌افزار به خودی خود از تنها متغیر باقی مانده که همان **x** است با حدود ذکر شده انتگرال‌گیری خواهد کرد.

فصل هشتم:
توابع حدی

- حد در MATLAB
- تابع حدی با حدود چپ و راست

۸-۱- حد در MATLA

✓ دستور limit

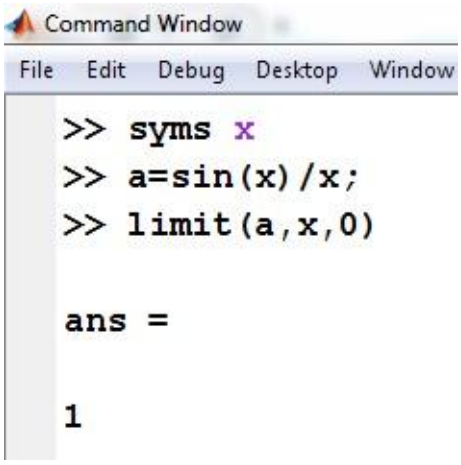
برای محاسبه توابع حدی در نرم‌افزار متلب از دستور **limit** استفاده می‌گردد.

مثال:

تابع حدی زیر را در نظر بگیرید.

$$.a = \lim_{x\to 0}\left(\frac{\sin x}{x}\right)$$

اکنون این تابع را با استفاده از دستور **limit** محاسبه خواهیم کرد.



با دقت در این مثال متوجه خواهیم شد که در دستور **limit(a,x,0)** , **a** تابع حدی مورد نظر، **x** متغیر حدی و ۰ نیز مقداری است که **x** به سمت آن میل می‌کند.

۸-۲- توابع حدی با حدود چپ و راست

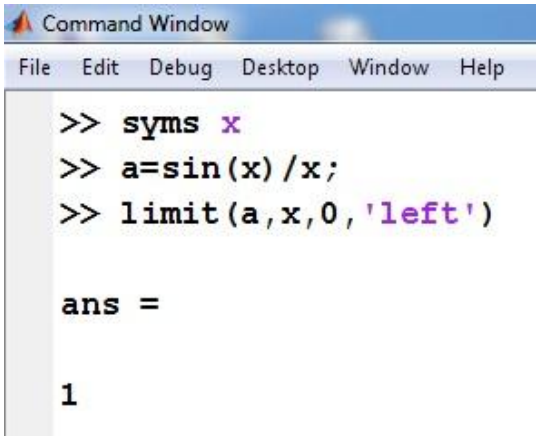
در برخی از توابع حدی، حد راست یا حد چپ تابع نیز مورد محاسبه قرار می‌گیرد، در این نوع از توابع به شکل زیر عمل می‌گردد.

تابع زیر را در نظر بگیرید.

$$.a = \lim_{x\to 0^-}\left(\frac{\sin x}{x}\right)$$

حال باید حد تابع **a** در زمانی که **x** به سمت 0^- میل می‌کند را محاسبه کرد.

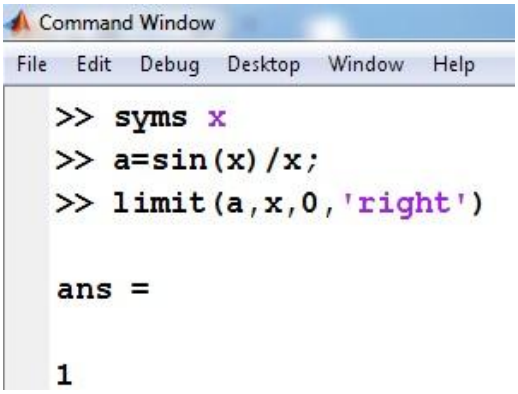
مثال:



با دقت در این مثال متوجه خواهیم شد که در دستور **limit(a,x,0,'left')** , **a** تابع حدی مورد نظر، **x** متغیر حدی و ۰ نیز مقداری است که **x** به سمت آن میل می‌کند و عبارت **left** نیز سمت حد مورد نظر را مشخص می‌کند.

برای تمرین، مثال فوق را زمانی که **x** به سمت 0^+ میل می‌کند را دوباره محاسبه می‌کنیم.

مثال:

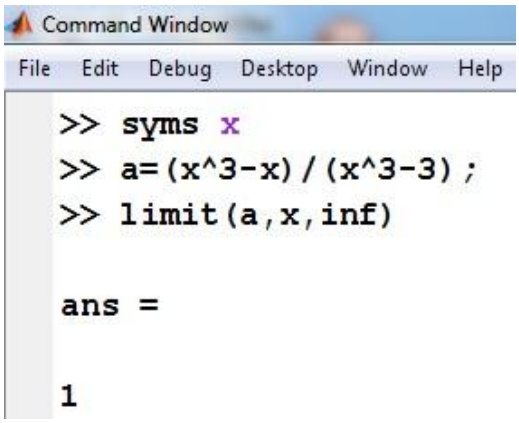


⇦نکته:

برای نوشتن **بی‌نهایت** در توابع حدی یا ... از عبارت **inf** استفاده می‌گردد.

مثال:

$$.\mathbf{a} = \mathbf{lim}_{x\to\infty}\left(\frac{x^3-x}{x^3-3}\right)$$



فصل نهم:
لاپلاس

- تبدیل لاپلاس در **MATLAB**
- عکس تبدیل لاپلاس در **MATLAB**

مبحث لاپلاس یکی از مباحث مهم در مباحث ریاضی و همچنین فنی، مهندسی می‌باشد. به همین دلیل یادگیری مبحث لاپلاس در فضای نرم‌افزار متلب نیز دارای اهمیت می‌باشد.

۹-۱- تبدیل لاپلاس در MATLAB

✓ دستور laplace

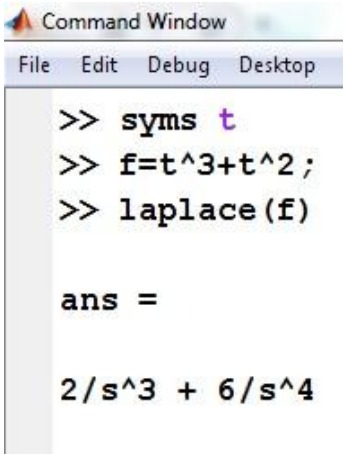
این دستور در نرم‌افزار متلب برای تبدیل تابع از حوزه‌ی زمان **(t)** به حوزه‌ی فرکانس **(f)** استفاده می‌گردد. (تبدیل لاپلاس).

تابع زیر را در نظر بگیرید.

مثال:

$$.f(t) = t^3 + t^2$$

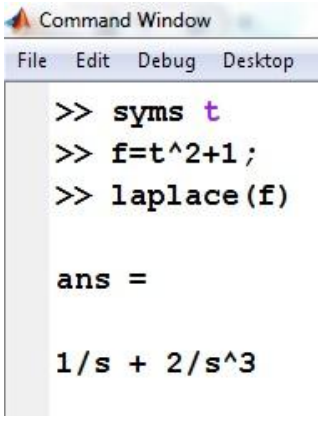
حال با دستور **laplace** از تابع فوق تبدیل لاپلاس می‌گیریم.



برای تمرین بیشتر چند تبدیل لاپلاس دیگر را انجام می‌دهیم.

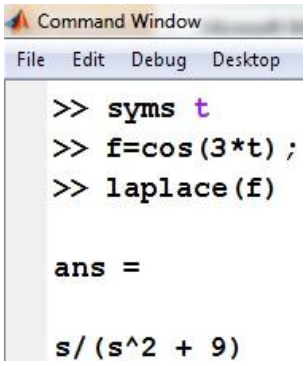
مثال:

$$.f(t) = t^2 + 1$$



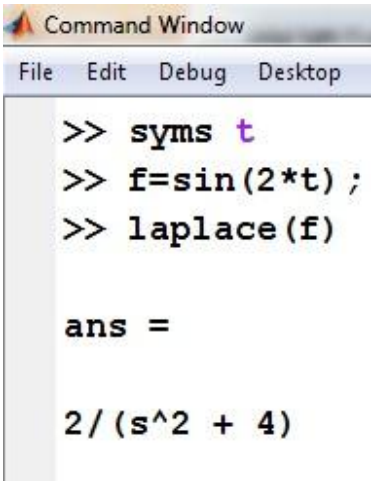
مثال:

$$.f(t) = cos(3t)$$



مثال:

$$.f(t) = sin(2t)$$



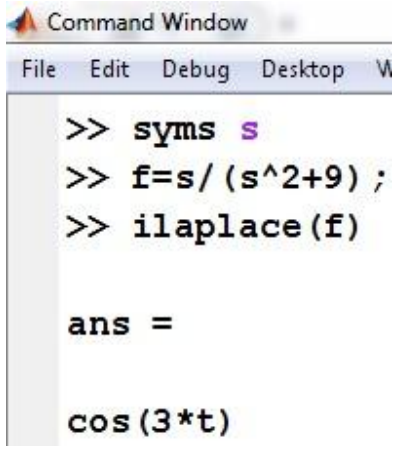
۹-۲- عکس تبدیل لاپلاس در MATLAB

✓ دستور ilaplace

این دستور در نرم‌افزار متلب برای تبدیل تابع از حوزه‌ی فرکانس **(f)** به حوزه‌ی زمان **(t)** استفاده می‌گردد. (عکس تبدیل لاپلاس).

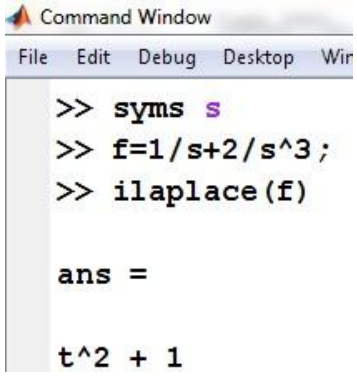
مثال:

$$.f(s) = \frac{s}{s^2+9}$$



مثال:

$$.f(s) = \frac{1}{s} + \frac{2}{s^3}$$



فصل دهم:
اعداد مختلط

- مختصات دکارتی
- مختصات قطبی

همانطور که می‌دانیم اعداد مختلط در مباحث ریاضی و همچنین مباحث فنی و مهندسی در دو نوع از مختصات نوشته می‌شود.

حالت اول: اعداد مختلط در مختصات دکارتی بوده که به صورت $\boldsymbol{x+yi}$ نوشته می‌گردد که **x** قسمت حقیقی و **y** قسمت موهومی عدد مختلط است.

حالت دوم: اعداد مختلط در مختصات قطبی بوده که به صورت $\mathbf{r}\measuredangle\boldsymbol{\theta}$ نوشته می‌گردد که **r** مقدار بردار و $\boldsymbol{\theta}$ زاویه عدد مختلط می‌باشد.

اکنون چگونگی نوشته شدن هر دو حالت عدد مختلط در نرم‌افزار متلب بیان می‌گردد.

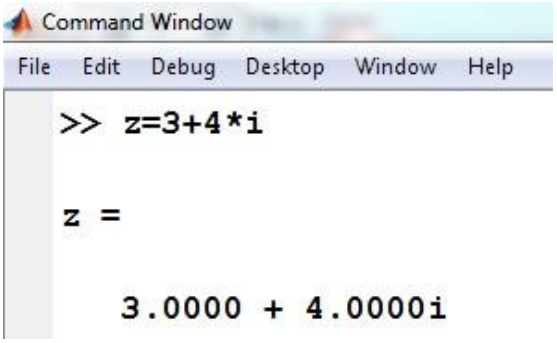
۱۰-۱- مختصات دکارتی

برای نوشتن عدد مختلط در مختصات دکارتی از رابطه‌ی زیر استفاده می‌کنیم.

$$\boldsymbol{.z = x + y * i}$$

مثال:

$$.z = 3 + 4i$$



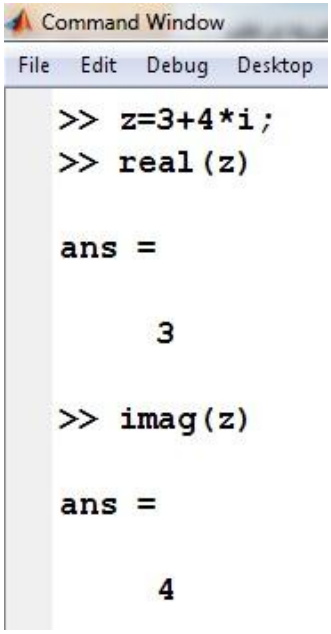
✓ **دستور real(z)**

این دستور قسمت **حقیقی** عدد مختلط را نمایش می‌دهد.

✓ **دستور imag(z)**

این دستور نیز , قسمت **موهومی** عدد مختلط را نمایش می‌دهد.

مثال:



۱۰-۲- مختصات قطبی

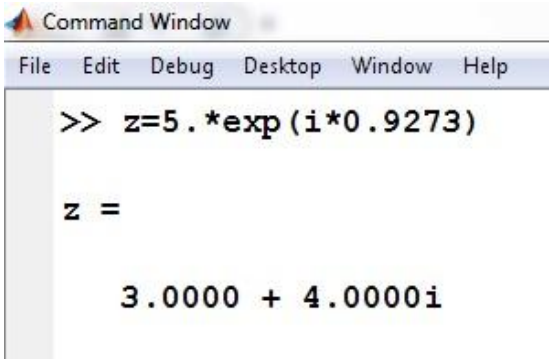
برای نوشتن عدد مختلط در مختصات قطبی از رابطه زیر استفاده می‌کنیم.

$$.z = r.* \exp(i * \theta)$$

r: عبارت است از مقدار عدد مختلط در مختصات قطبی

θ: عبارت است از زاویه عدد مختلط در مختصات قطبی

مثال:



اکنون برای تبدیل عدد مختلط فوق از حالت دکارتی به حالت قطبی به دو فاکتور , اندازه و زاویه‌ی عدد مختلط نیاز داریم.

برای بدست آوردن این دو فاکتور از دستورات زیر استفاده می‌کنیم.

- ✓ **دستور abs(z)**

از این دستور برای بدست آوردن **مقدار عدد مختلط** استفاده می‌کنیم.

- ✓ **دستور angle(z)**

از این دستور نیز برای بدست آوردن **زاویه عدد مختلط** استفاده می‌کنیم.

مثال:

```
Command Window
File  Edit  Debug  Desktop
>> z=3+4*i;
>> abs(z)

ans =

     5

>> angle(z)

ans =

    0.9273
```

⇦نکته:

کلیه زوایا در نرم‌افزار متلب بر حسب **رادیان** می‌باشند، پس در این مثال نیز زاویه‌ی بدست آمده برحسب رادیان است.

فصل یازدهم:
معادلات چند مجهولی

- حل معادلات چند مجهولی

۱۱-۱- حل معادلات چند مجهولی

برای حل معادلات دو روش وجود دارد.

روش اول: روش ماتریس

بسیاری از معادلات یا به عبارت ساده‌تر چند معادله و چند مجهول را می‌توان با استفاده از روش ماتریس‌ها به راحتی حل کرد.

مثال:

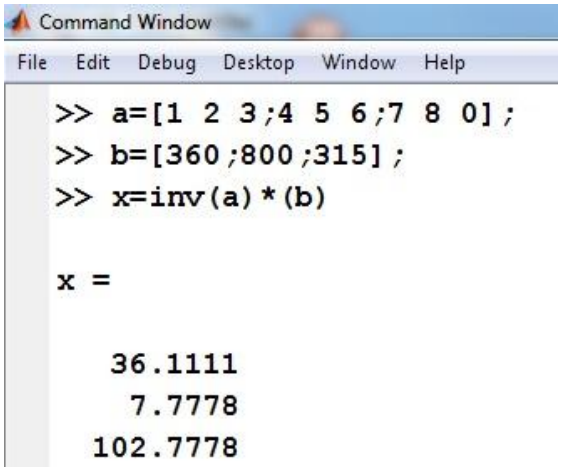
برای مثال، دستگاه معادلات زیر را در نظر بگیرید.

$$.\begin{cases} x_1 + 2x_2 + 3x_3 = 360 \\ 4x_1 + 5x_2 + 6x_{30} = 800 \\ 7x_1 + 8x_2 + 0x_3 = 315 \end{cases}$$

دستگاه معادلات فوق را می‌توان به شکل زیر نوشت:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 360 \\ 800 \\ 315 \end{bmatrix}.$$

اکنون برای حل این دستگاه به روش مثال زیر عمل می‌کنیم:



با توجه به مثال فوق x_1 , x_2 و x_3 به ترتیب **۳۶.۱۱۱۱** , **۷.۷۷۷۸** و **۱۰۲.۷۷۷۸** خواهد بود.

روش دوم: استفاده از دستور زیر:

✓ **دستور solve**

اکنون با استفاده از دستور **solve** دستگاه دو معادله و دو مجهولی زیر رو حل می‌نمائیم.

مثال:

$$\begin{cases} x - 2y = -7 \\ -x + y = -1 \end{cases}.$$

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> s=solve('x-2*y=-7','-x+y=-1')

s =

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

>> s=[s.x s.y]

s =

[ 9, 8]
```

⇦نکته اول:

هنگام استفاده از دستور **solve** به شیوه نوشته شده در مثال بالا , نیازی به استفاده از دستور **syms** برای تعریف متغیر نمی‌باشد.

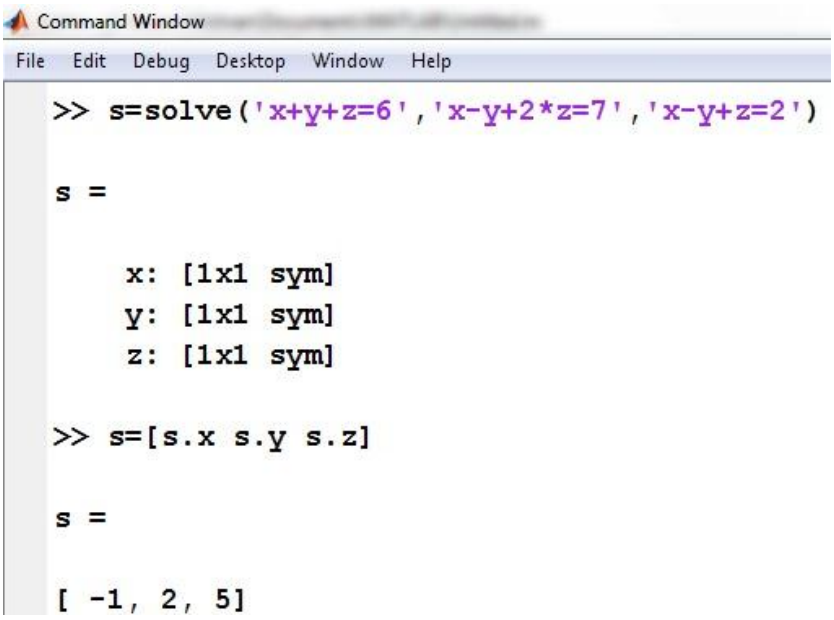
⇦نکته دوم:

هر معادله باید مابین دو علامت **'** قرار گیرد و معادلات با علامت **,** از هم جدا گردند.

برای تمرین بیشتر، دستگاه سه معادله و سه مجهولی زیر را حل می‌نمائیم.

مثال:

$$.\begin{cases} x+y+z=6 \\ x-y+2z=7 \\ x-y+z=2 \end{cases}$$



فصل دوازدهم:
M - File

- تعریف M-File
- ایجاد یک M-File در MATLAB

۱۲-۱- تعریف M-File

کدنویسی در فضای **Command Window** همیشه مطلوب نیست و دارای ایرادهایی است، مانند اینکه اگر در فضای **Command Widow** در کدنویسی دچار اشتباه گردیم، برای تصحیح این اشتباه باید کل دستورات نوشته شده در صفحه **Command Window** را پاک نموده و دوباره دستورات را به شکل صحیح در این صفحه بنویسیم. به عبارتی بهتر صفحه‌ی **Command Window** غیر قابل ویرایش می‌باشد. همچنین این صفحه برای نوشتن کدهای پیچیده و طولانی به هیچ وجه مناسب نیست، به خاطر همین به فضایی نیازمندیم که در آن بتوانیم بدون مشکلات بیان شده، به انجام عمل کدنویسی بپردازیم. این فضا مناسب برای کدنویسی در نرم‌افزار متلب **M-File** نام دارد.

به عبارتی بهتر **M-File** صفحه‌ای است که ما می‌توانیم عمل کدنویسی را در آن به طور مطلوب‌تری انجام دهیم و بدون برخورد با مشکلات بیان شده در صفحه‌ی **Command Window** عمل کدنویسی را به صورت بهتر و حرفه‌ای تر انجام دهیم.

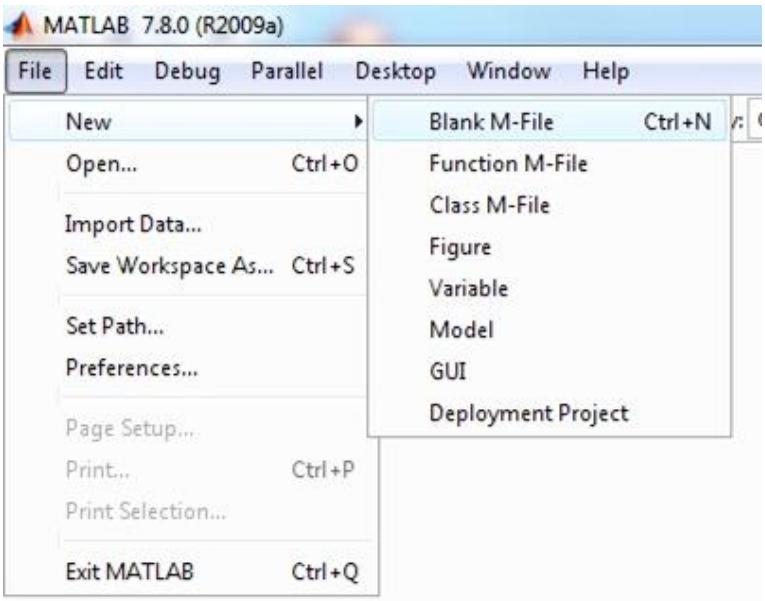
۱۲-۲- ایجاد یک M-File در نرم‌افزار MATLAB

برای ایجاد یک **M-File** در نرم‌افزار متلب چهار روش وجود دارد:

روش اول: در قسمت بالای پنجره اصلی به ترتیب زیر عمل می‌کنیم.

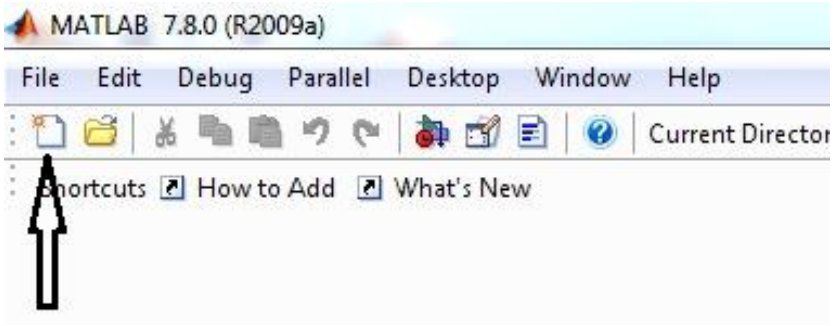
File → New → M-File

مانند تصویر زیر:



روش دوم: در قسمت نوار ابزار از طریق گزینه **New M-File**

در تصویر زیر این گزینه نمایش داده شده است:



روش سوم: با استفاده از کلیدهای ترکیبی **Ctrl+N**

روش چهارم: با تایپ دستور **edit** در صفحه‌ی **Command Window**

با استفاده از یکی از دستورات ذکر شده، به راحتی می‌توانیم یک صفحه‌ی **M-File** باز کرده و عمل کدنویسی را در آن انجام دهیم.

***چند نکته بسیار حائز اهمیت در کد نویسی و اجرای دستور در M-File**

⇦نکته اول:

توصیه می‌گردد در اول صفحه‌ی **M-File** از دستور **clear all** استفاده گردد. در این صورت تمامی متغیرهای تعریف شده از قبل، پاک شده و اخلالی در انجام برنامه ایجاد نخواهد شد. همانگونه که قبلا هم ذکر گردید، یک کدنویس حرفه‌ای، حتما در اول عمل کدنویسی خود، از دستور **clear all** استفاده خواهد کرد.

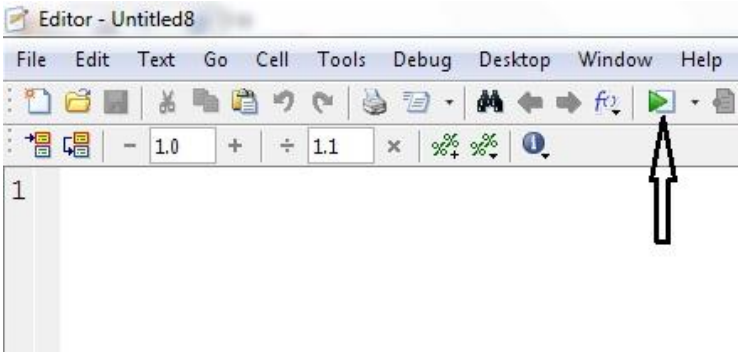
⇦نکته دوم:

برای ذخیره دستورات کدنویسی از دستور **Save** موجـــود در منوی **File** صفحه‌ی **M-File** استفاده می‌کنیم.

⇦نکته سوم:

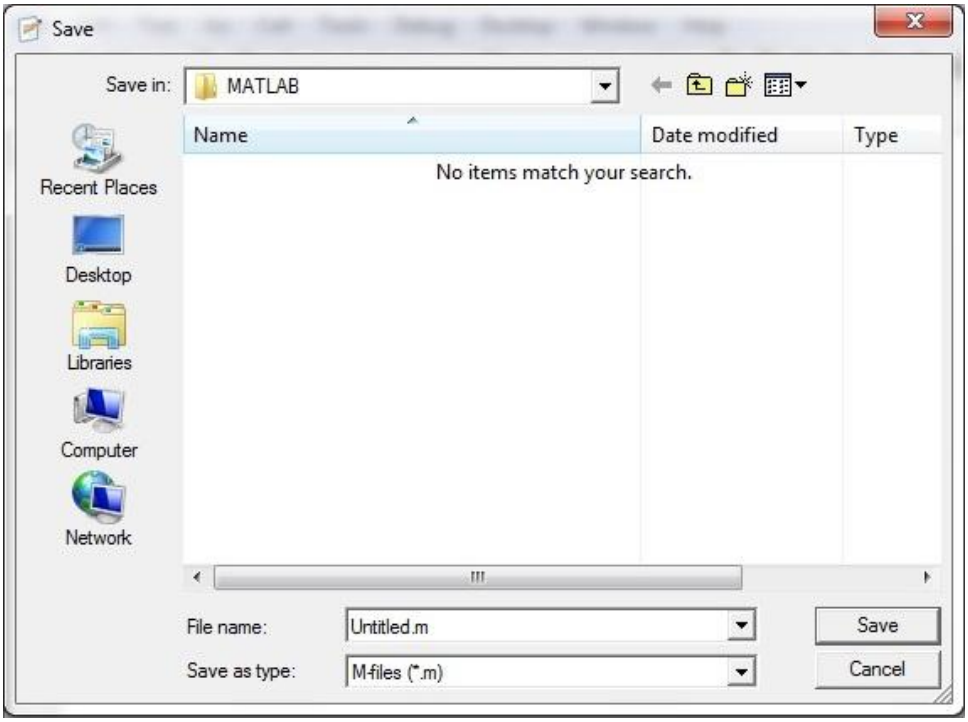
برای اجرای دستور از گزینه **Save and run** که به صورت آماده در قسمت نوار ابزار صفحه‌ی **M-File** قرار دارد استفاده می‌کنیم.

در تصویر زیر این گزینه نمایش داده شده است:



همانگونه که از نام این گزینه معلوم است، این گزینه عمل ذخیره و اجرای دستور را به صورت همزمان انجام می‌دهد.

هنگام اجرای این دستور با پنجره نمایش داده شده در زیر روبرو خواهیم شد:

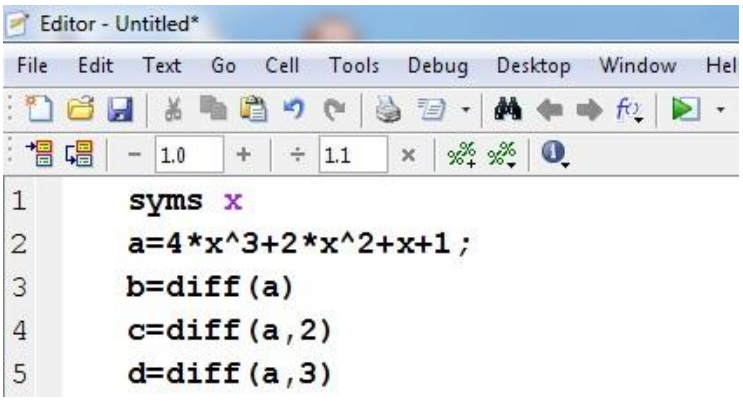


همانگونه که در این تصویر مشهود است، در این پنجره می‌توانیم محل ذخیره **M-File** و همچنین نام **M-File** مورد نظر را تعیین کنیم.

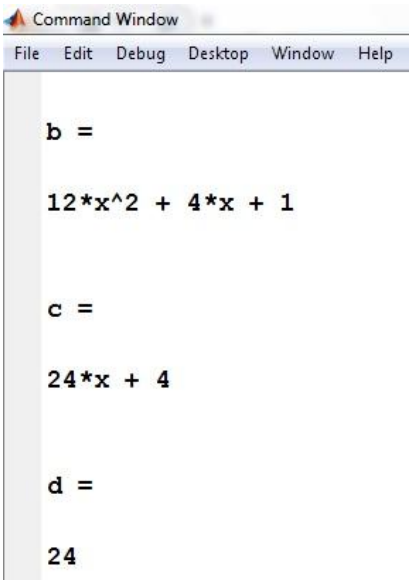
همچنین قابل ذکر است که **M-File** های مورد نظر دارای پسوند **m** خواهد بود.

بعد از اجرا کردن برنامه، نتیجه برنامه نوشته شده در صفحه‌ی **Command Window** نمایش داده خواهد شد.

مثال:



در تصویر بالا، دستورات نوشته شده در یک **M-File** نشان داده شده است و همچنین در تصویر پائین نیز، پس از اجرای دستور در صفحه‌ی **M-File**، نتیجه حاصل شده از اجرای دستور در صفحه‌ی **Command Window** نمایش داده شده است.



⇦نکته:

یکی دیگر از ویژگی‌های مطلوب کدنویسی در **M-File** این است که می‌توان فایل کدنویسی شده را ذخیره کرد و در صورت نیاز آن را دوباره اجرا و یا ویرایش کرد، حال اینکه ویژگی بیان شده در صفحه‌ی**Command Window** موجود نیست.

✓ استفاده از % در کدنویسی

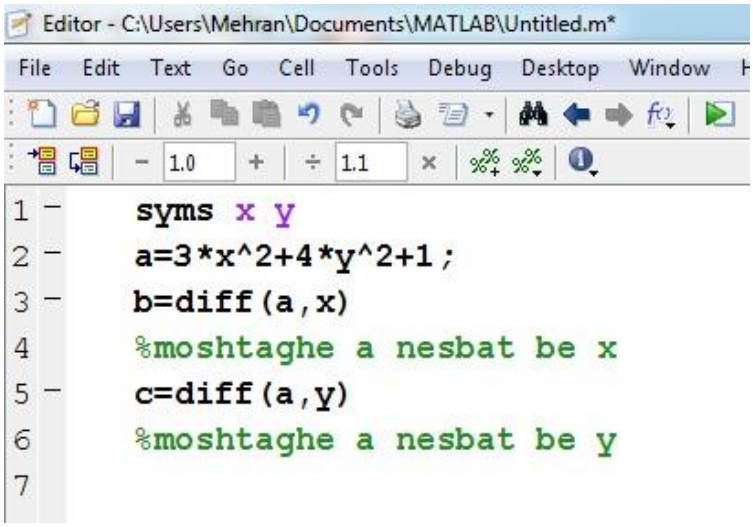
یک نکته بسیار مهم و حائز اهمیت در کدنویسی، مخصوصا کدهای پیچیده و طولانی، نوشتن توضیحات در کنار دستورات اصلی می‌باشد. بدین صورت که به دلیل حجم زیاد دستورات و طولانی بودن پروسه کدنویسی، گاهی ممکن است که نویسنده کد، علت و

چرایی دستورات را فراموش کند و برای یاد آوری مجدد آن مجبور گردد که بارها کدهای نوشته شده را مطالعه کرده و در نهایت به مفهوم آن‌ها پی ببرد که این کار باعث افزایش زمان کدنویسی و همچنان باعث بروز مشغله ذهنی برای نویسنده کد می‌گردد.

برای اجتناب از برخورد با چنین مشکلی، از نوشتن توضیحات در کنار برخی از دستورات استفاده می‌گردد و برای انجام این عمل از دستور **%** استفاده می‌گردد.

شایان ذکر است که از این روش هم در صفحه‌ی **Command Window** و هم در صفحه‌ی **M-File** می‌توان استفاده کرد.

مثال:

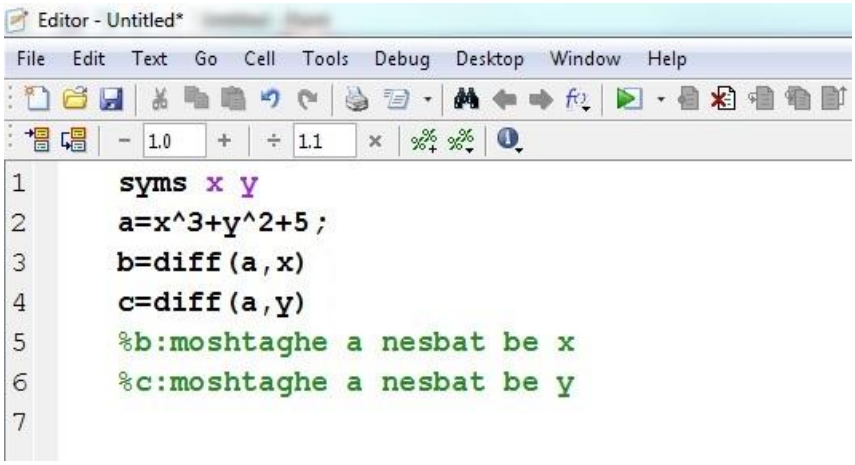


همانطور که مشخص است، توضیحاتی که بعد از علامت **%** نوشته می‌شوند، نرم افزار متلب آن‌ها را به عنوان توضیح ثبت کرده و در نتیجه دستورات اصلی دخالت نمی‌دهد.

⇦نکته:

البته باید این موضوع را نیز به خاطر داشت که فقط می‌توان در سطر مقابل علامت **%** توضیحات را ثبت کرد و اگر نیاز باشد که در سطر بعدی هم توضیحاتی نوشته شود، حتما باید در ابتدای سطر بعدی نیز از علامت **%** استفاده گردد.

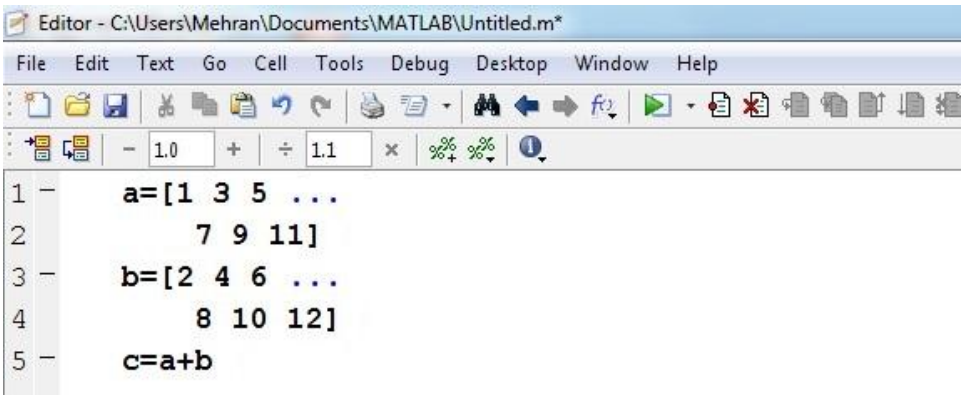
مثال:

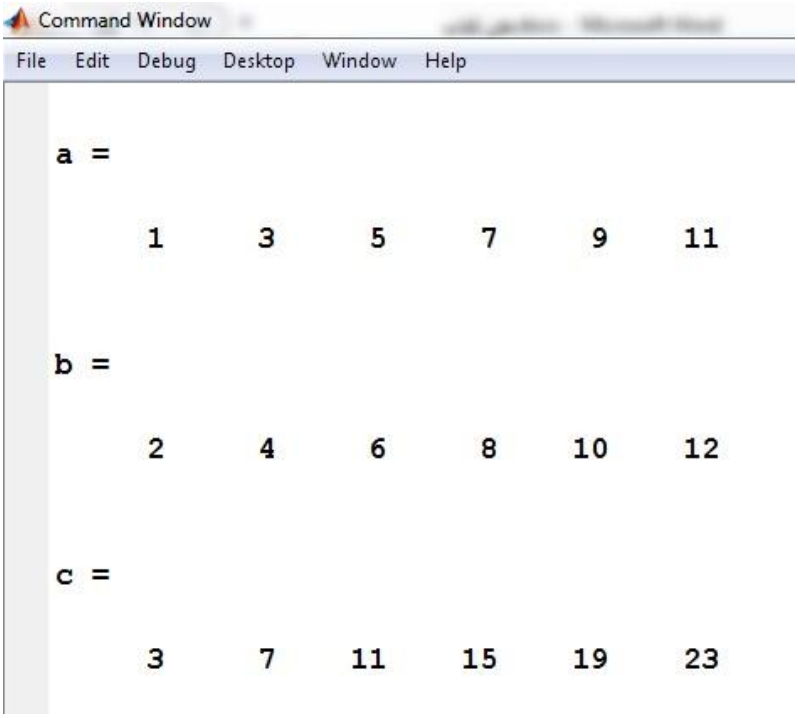


✓ استفاده از ... (سه نقطه) در کدنویسی

گاهی در کدنویسی با سطرهایی از دستور برخورد می‌کنیم که طولانی هستند، این سطرها به علت طولانـــی بودن فضای زیادی را اشغال کرده و باعث ایجاد بی‌نظمی در کدنویسی می‌گردند، برای اجتناب از طولانی شدن سطر، سعی می‌کنیم به جای نوشتن تمام دستور در یک سطر، قسمتی از دستور را در سطر بعدی بنویسیم، برای انجام این عمل از ... در انتهای سطر استفاده می‌کنیم.

مثال:




Command Window
File Edit Debug Desktop Window Help
a =
1 3 5 7 9 11
b =
2 4 6 8 10 12
c =
3 7 11 15 19 23

فصل سیزدهم:
ترسیم گرافیکی توابع

- ترسیم توابع با دستورات
- نحوه‌ی نمایش نمودار با شیوه‌های متفاوت
- ترسیم سه بعدی توابع در MATLAB

۱۳-۱- ترسیم گرافیکی توابع

در این قسمت از بحث، به ترسیم گرافیکی توابع خواهیم پرداخت، به همین منظور با تعدادی از دستورات رایج در این بحث آشنا خواهیم شد.

✓ دستور plot

از این دستور برای ترسیم بردارها و توابع گسسته استفاده می‌گردد.

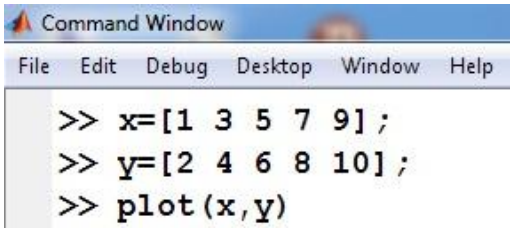
در واقع دستور **plot** مقادیر گسسته را که هر کدام به صورت یک نقطه هستند به هم وصل کرده و به صورت یک شکل پیوسته نشان می‌دهد.

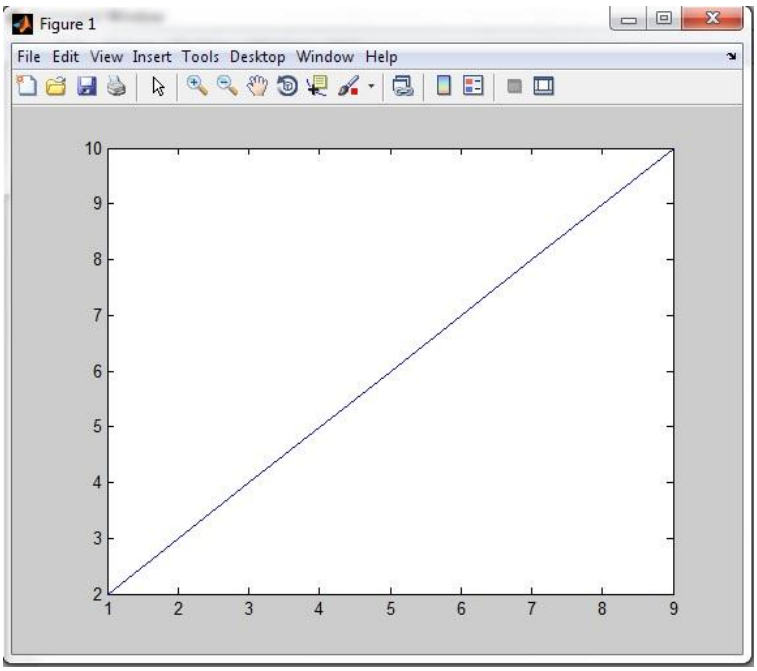
مثال:

بردارهای زیر را در نظر بگیرید.

$$.\boldsymbol{x} = [\mathbf{1\ 3\ 5\ 7\ 9}]$$
$$.\boldsymbol{y} = [\mathbf{2\ 4\ 6\ 8\ 10}]$$

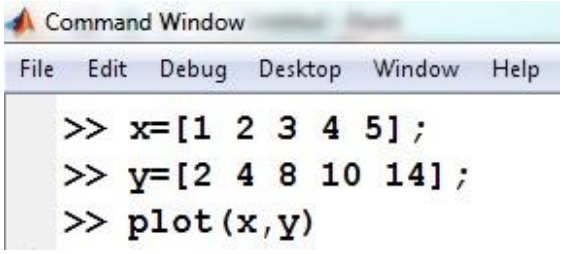
اکنون با استفاده از دستور **plot** شکل این دو بردار را ترسیم می‌نمائیم.





مثال:

$$.x = [1\ 2\ 3\ 4\ 5]$$
$$.y = [2\ 4\ 8\ 10\ 14]$$


Command Window
File Edit Debug Desktop Window Help
>> x=[1 2 3 4 5];
>> y=[2 4 8 10 14];
>> plot(x,y)

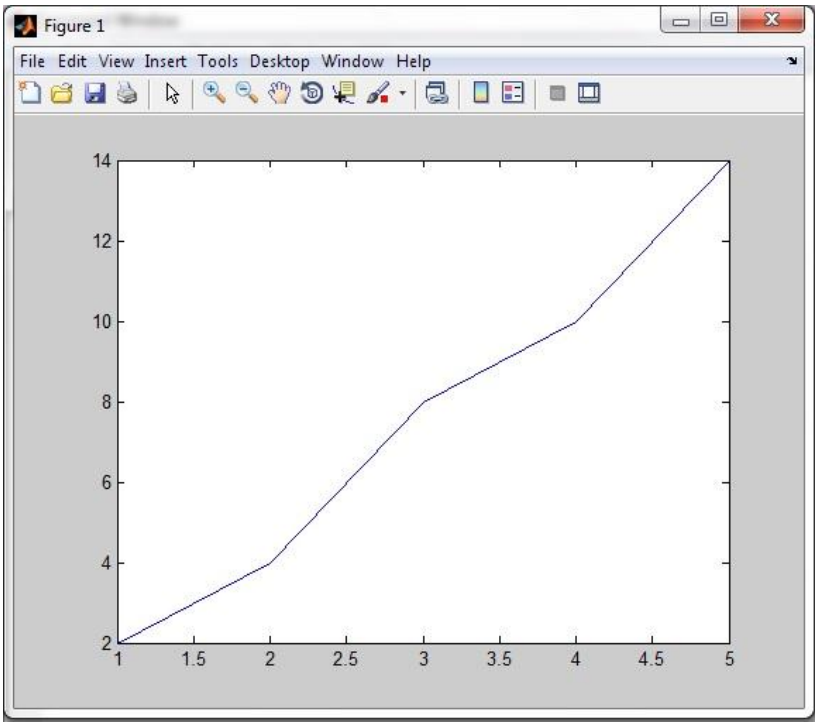


Figure 1
File Edit View Insert Tools Desktop Window Help
14
12
10
8
6
4
2
1
1.5
2
2.5
3
3.5
4
4.5
5

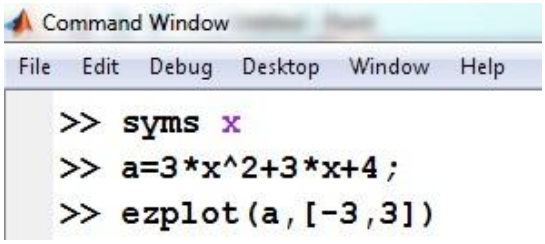
✓ دستور ezplot

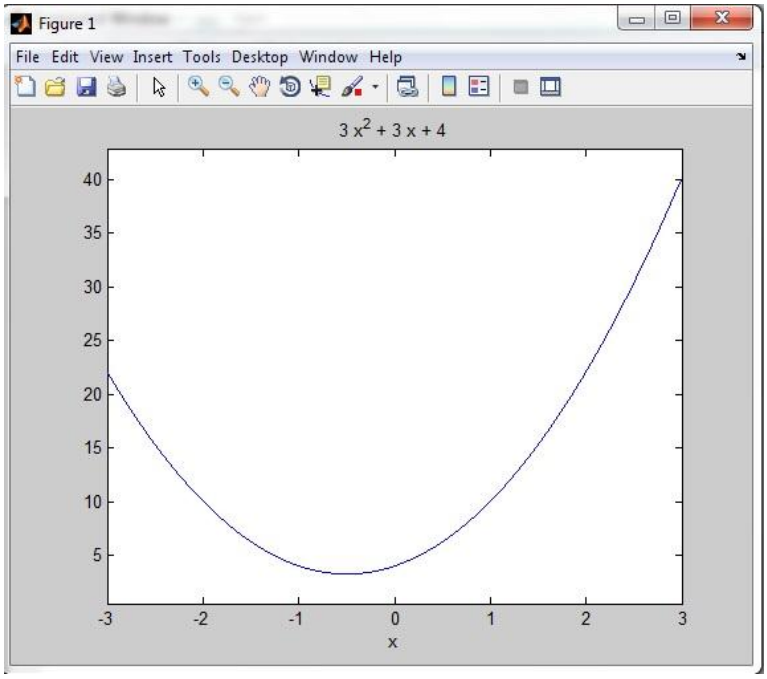
از این دستور برای ترسیم توابع پیوسته به صورت گرافیکی استفاده می‌کنیم.

مثال:

تابع زیر را در نظر گرفته و با استفاده از دستور **ezplot** شکل خروجی آن را در بازه **[-3,3]** ترسیم می‌نمائیم.

$$.a = 3x^2 + 3x + 4$$





با دقت در مثال فوق می‌توان به روشنی دریافت که **a** تابع مثال و **[-3,3]** نیز محدوده‌ی رسم تابع می‌باشد.

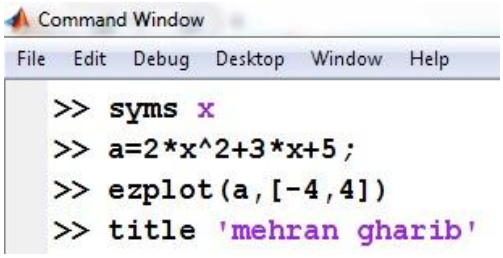
✓ دستور title

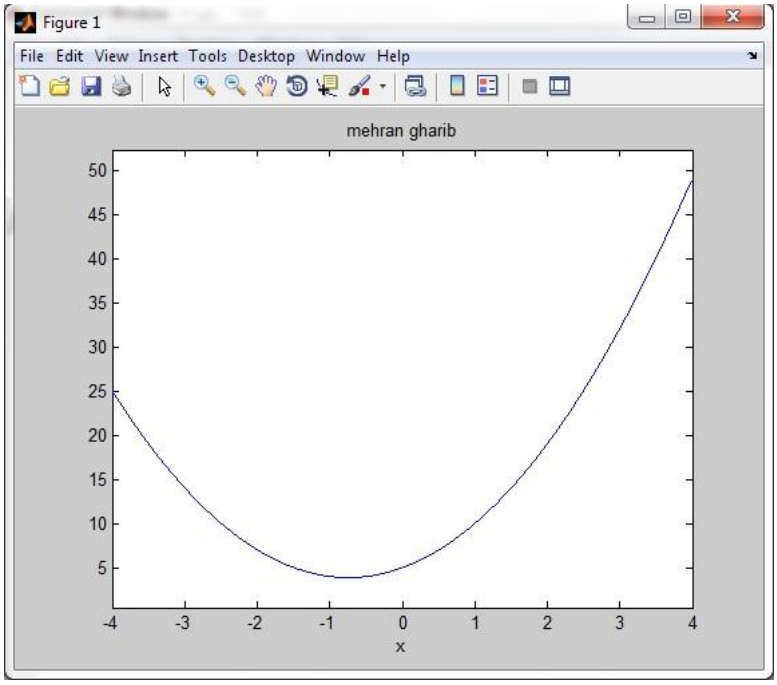
اغلب اوقات برای اینکه بتوان، ترسیمات مختلف را در بین فایل‌ها و تصاویر مختلف تشخیص داد، از نام گذاری ترسیمات استفاده می‌گردد، برای انجام این عمل از دستور **title** استفاده می‌نمائیم.

مثال:

تابع زیر را در نظر گرفته و با استفاده از دستور **ezplot** شکل خروجی آن را در بازه **[-4,4]** ترسیم می‌نمائیم.

$$.a = 2x^2 + 3x + 5$$





⇦نکته:

دستور **title** در مورد همه نوع از ترسیمات صدق می‌کند و همچنین باید دقت گردد که نام انتخاب شده بعد از دستور **title** حتما ما بین دو ' قرار گیرد.

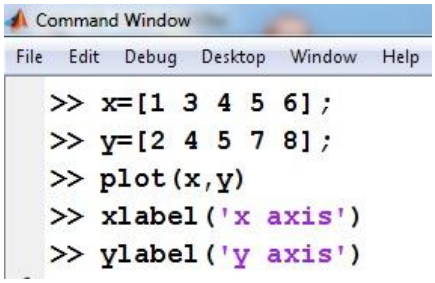
✓ **دستور xlabel**

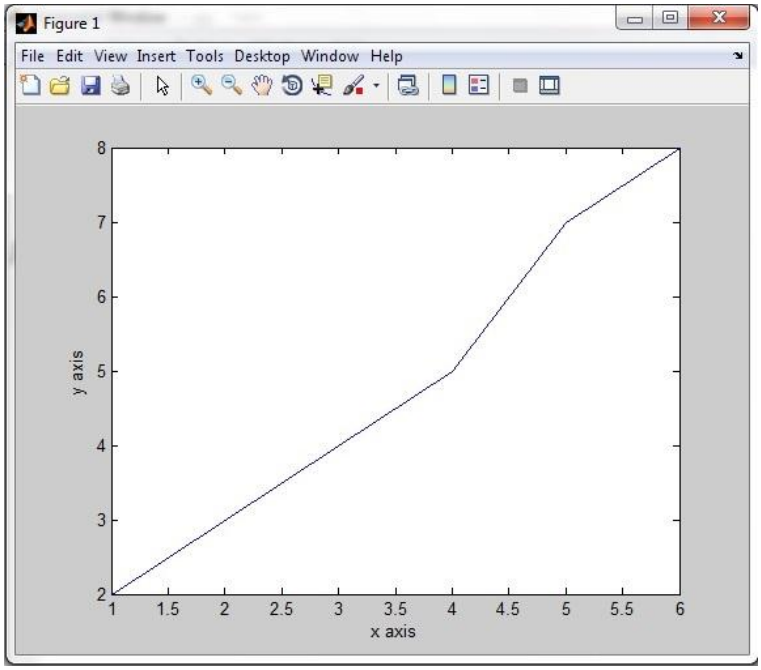
از این دستور برای عنوان‌گذاری محور افقی استفاده می‌گردد.

✓ **دستور ylabel**

از این دستور نیز برای عنوان‌گذاری محور عمودی استفاده می‌کنیم.

مثال:

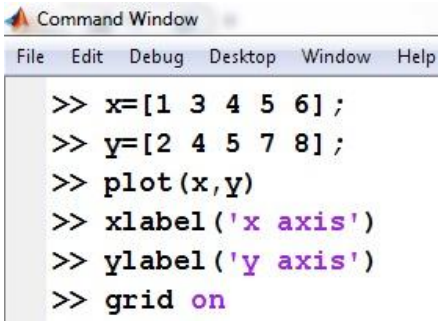


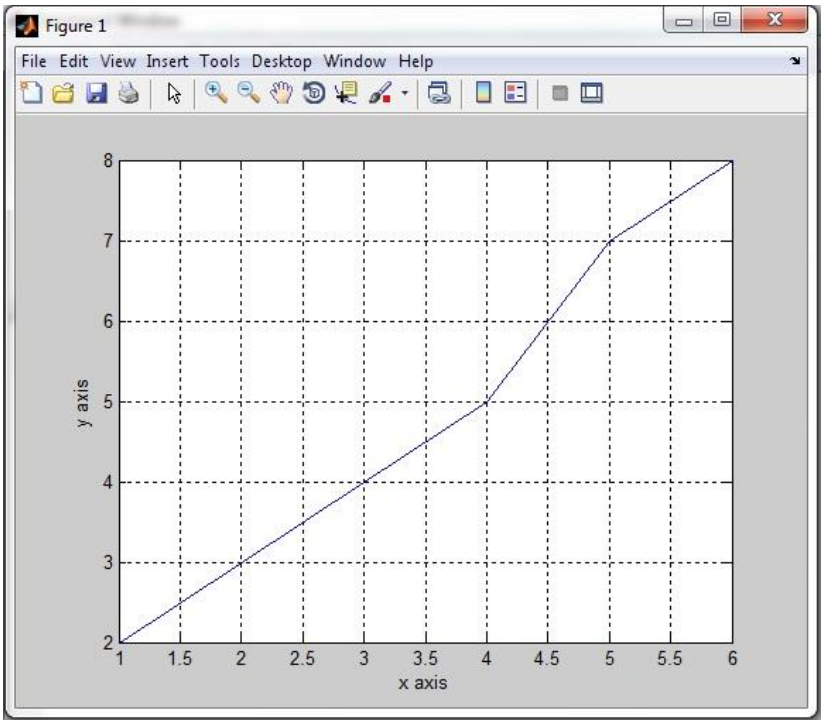


✓ دستور grid on

با این دستور می‌توان صفحه نمایش نمودار را شبکه بندی کرد.

مثال:





✓ **دستور grid off**

با این دستور می‌توان صفحه نمایش نمودار را از حالت شبکه بندی خارج کرد.

مثال:

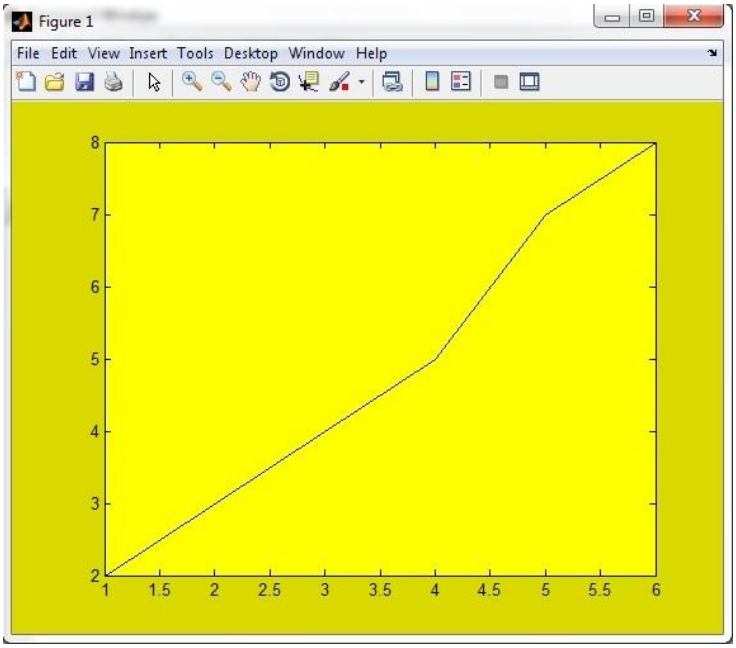
```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> x=[1 3 4 5 6];
>> y=[2 4 5 7 8];
>> plot(x,y)
>> xlabel('x axis')
>> ylabel('y axis')
>> grid on
>> grid off
```

✓ دستور whitebg

در نرم افزار متلب، رنگ پس‌زمینه شکل‌ها به صورت پیش فرض سفید قرار داده شده است، اگر بخواهیم در مواردی رنگ پس زمینه را تغییر بدهیم، از این دستور استفاده می‌کنیم.

مثال:

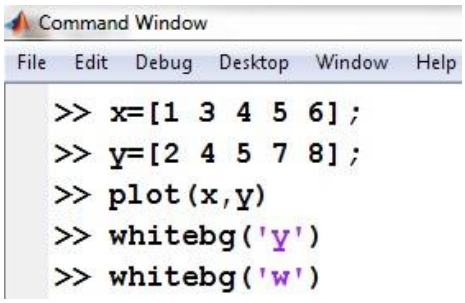
```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> x=[1 3 4 5 6];
>> y=[2 4 5 7 8];
>> plot(x,y)
>> whitebg('y')
```



⇦نکته:

برای بازگرداندن رنگ پس زمینه شکل به رنگ سفید مانند مثال زیر عمل می‌کنیم.

مثال:



⇦نکته:

کد بندی رنگ‌ها در نرم‌افزار متلب در جدول زیر ارائه گردیده است.

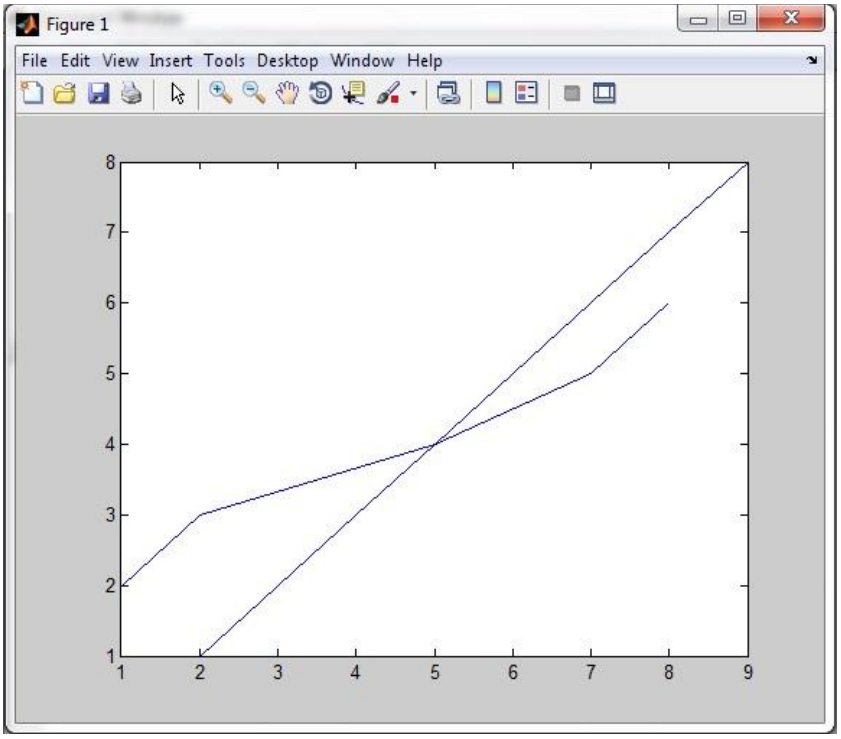
رنگ	کد رنگ
قرمز	r
سبز	g
آبی	b
آبی مایل سبز	c
صورتی مایل به قرمز	m
زرد	y
سیاه	k
سفید	w

✓ دستور hold on

از این دستور برای ترسیم چند نمودار در یک شکل استفاده می‌گردد، گاهی لازم است تا چند نمودار در کنار هم و در یک شکل رسم گردند تا بر روی آن‌ها مقایسه‌ی بهتری صورت گیرد، در این موارد از این دستور استفاده می‌گردد.

مثال:

```
>> x=[1 2 5 6 8 9];
>> y=[2 3 4 5 7 8];
>> z=[1 2 3 4 5 6];
>> plot(x,y)
>> hold on
>> plot(y,z)
>> hold on
```

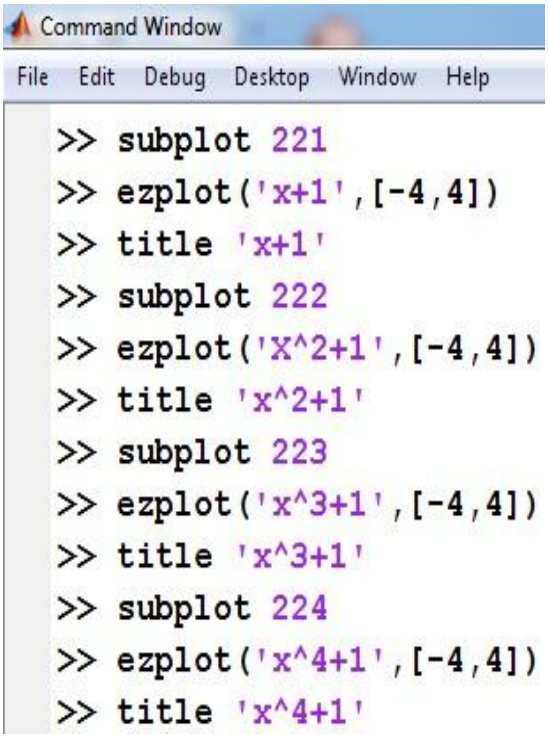


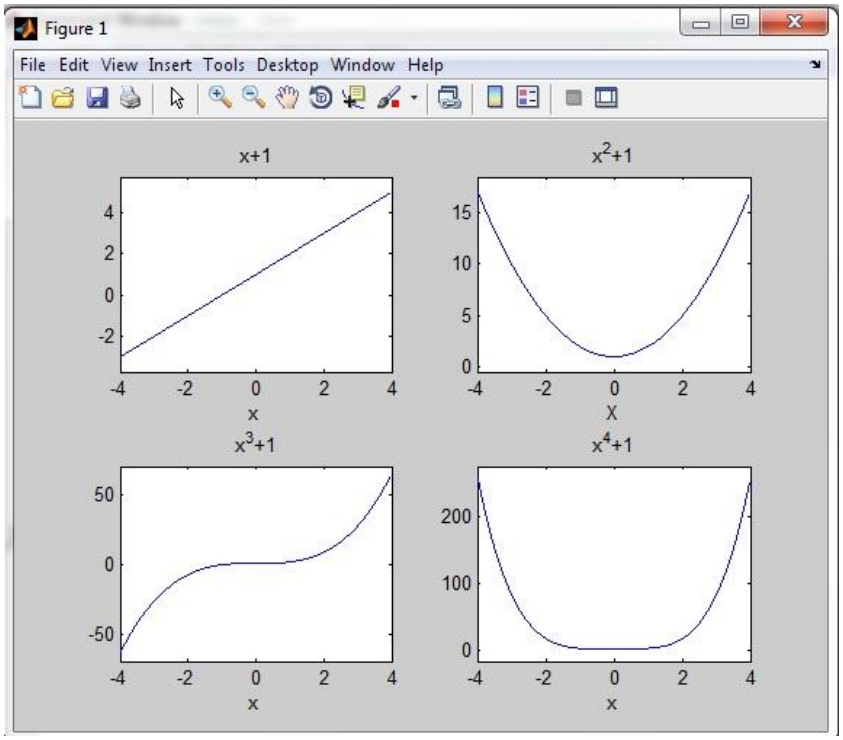
✓ دستور subplot

از این دستور برای ترسیم چندین شکل در کنار هم مورد استفاده قرار می‌گردد.

البته باید دقت کرد که تفاوت دستور **hold on** و دستور **subplot** در این است که دستور **hold on** چند نمودار را در یک شکل واحد رسم می‌کند اما در دستور **subplot** چند نمودار را در چند شکل کنار هم رسم می‌گردد.

مثال:





⇦نکته:

با دقت در دستورات نوشته شده می‌توان دریافت که برای هر شکل و نمودار در سه سطر دستور نوشته شده است، که برای مثال سه سطر دستور نوشته شده برای شکل و نمودار اول را بررسی می‌کنیم.

سطر اول دستور: subplot 221

در این سطر، از دستور **subplot** برای ترسیم نمودار در شکل اول استفاده می‌کنیم، عبارت **۲۲۱** در مقابل دستور **subplot** این را می‌رساند که شکل و نمودار ترسیم شده به عنوان شکل اول قرار گیرد.

سطر دوم دستور: ezplot('X^2+1',[-4,4])

در این سطر , از دستور **ezplot** برای ترسیم نمودار استفاده می‌کنیم.

سطر سوم دستور: title 'x^2+1'

در این سطر , شکل مورد نظر را با استفاده از دستور **title** نام‌گذاری می‌کنیم.

⇦نکته:

با دقت در دستورات نوشته شده می‌توان دریافت که در سطر اول هر چهار دستور در مقابل دستور **subplot** از اعداد , **۲۲۱** , **۲۲۲** , **۲۲۳** , **۲۲۴** استفاده شده است، از این اعداد در مقابل دستور **subplot** به این منظور استفاده می‌گردد که نرم‌افزار متلب متوجه گردد که هر شکل و نمودار به عنوان چندمین شکل و نمودار ترسیم گردد.

subplot 221: این مفهوم را می‌رساند که شکل و نمودار ترسیم شده برای این تابع به عنوان شکل و نمودار اول ترسیم گردد.

subplot 222: این مفهوم را می‌رساند که شکل و نمودار ترسیم شده برای این تابع به عنوان شکل و نمودار دوم ترسیم گردد.

subplot 223: این مفهوم را می‌رساند که شکل و نمودار ترسیم شده برای این تابع به عنوان شکل و نمودار سوم ترسیم گردد.

subplot 224: این مفهوم را می‌رساند که شکل و نمودار ترسیم شده برای این تابع به عنوان شکل و نمودار چهارم ترسیم گردد.

⇦نکته:

همچنین از روش زیر نیز در دستور **subplot** می‌توان برای تعیین مکان دستور استفاده کرد.

- ✓ **subplot(2,2,1)**

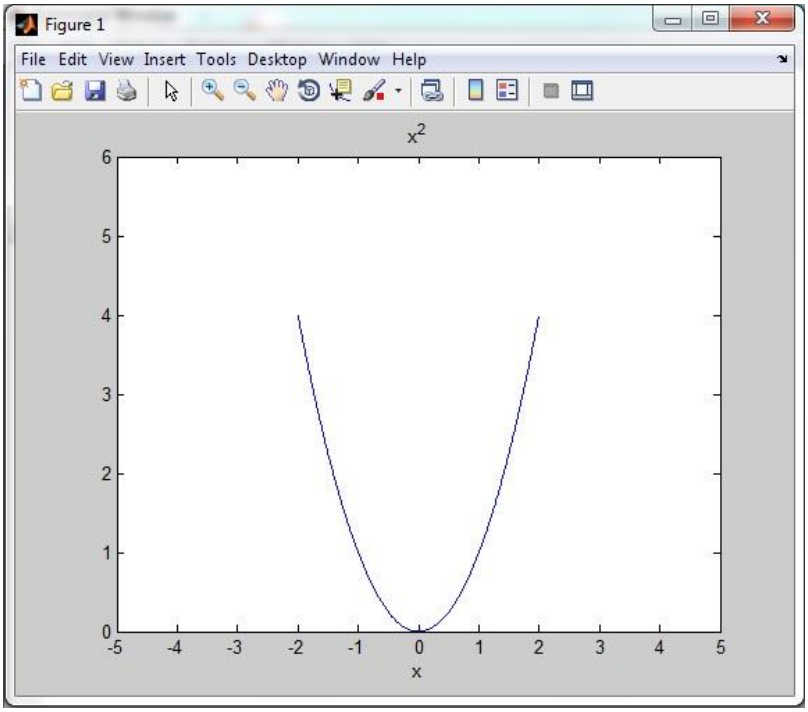
به عبارتی بهتر هر دو دستور **subplot 221** و **subplot(2,2,1)** یک عمل یکسان را انجام خواهند داد.

✓ دستور axis(a b c d)

از این دستور برای رنج بندی محور افقی و عمودی شکل استفاده می‌گردد، به نحوی که **a** تا **b** رنج محور افقی و **c** تا **d** نیز رنج محور عمودی را مشخص می‌کند.

مثال:

```
>> syms x
>> a=x^2;
>> ezplot(a,[-2,2])
>> axis([-5 5 0 6])
```



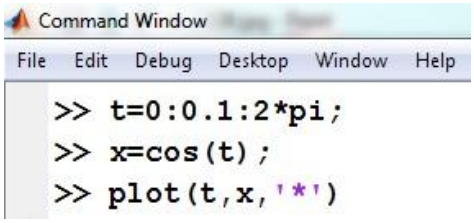
با دقت در این مثال می‌توان دریافت که محور افقی از **۵**- تا **۵** و همچنین محور عمودی از **۰** تا **۶** رنج بندی شده است.

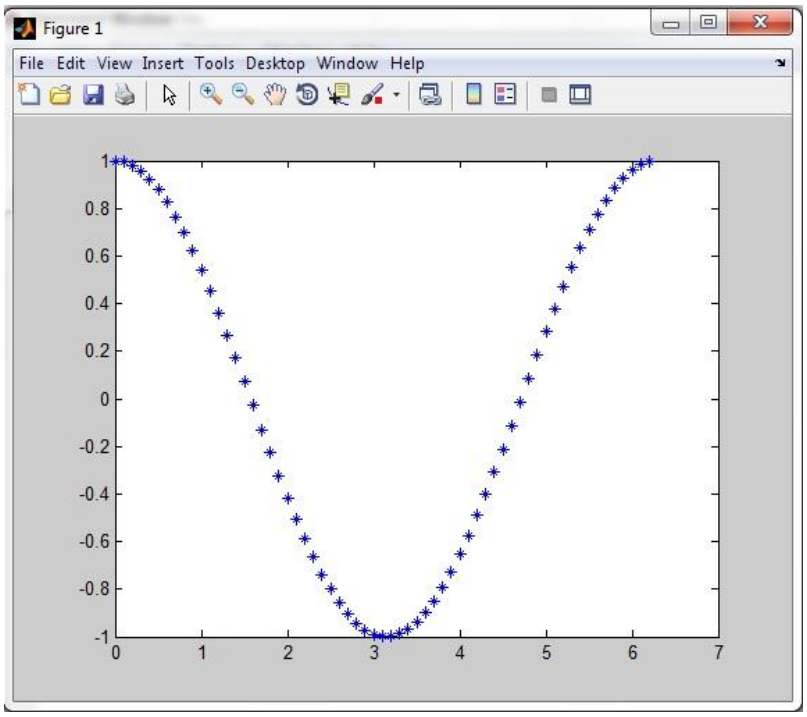
۱۳-۲- نحوه نمایش نمودار با شیوه‌های متفاوت

نمودارهای رسم شده در نرم‌افزار متلب را می‌توان به شیوه‌های گوناگون رسم کرد.

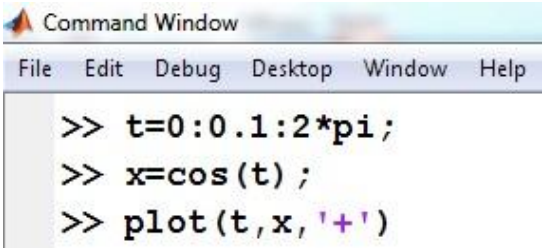
به مثال‌های زیر دقت کنید.

مثال:





مثال:


Command Window
File Edit Debug Desktop Window Help
>> t=0:0.1:2*pi;
>> x=cos(t);
>> plot(t,x,'+')

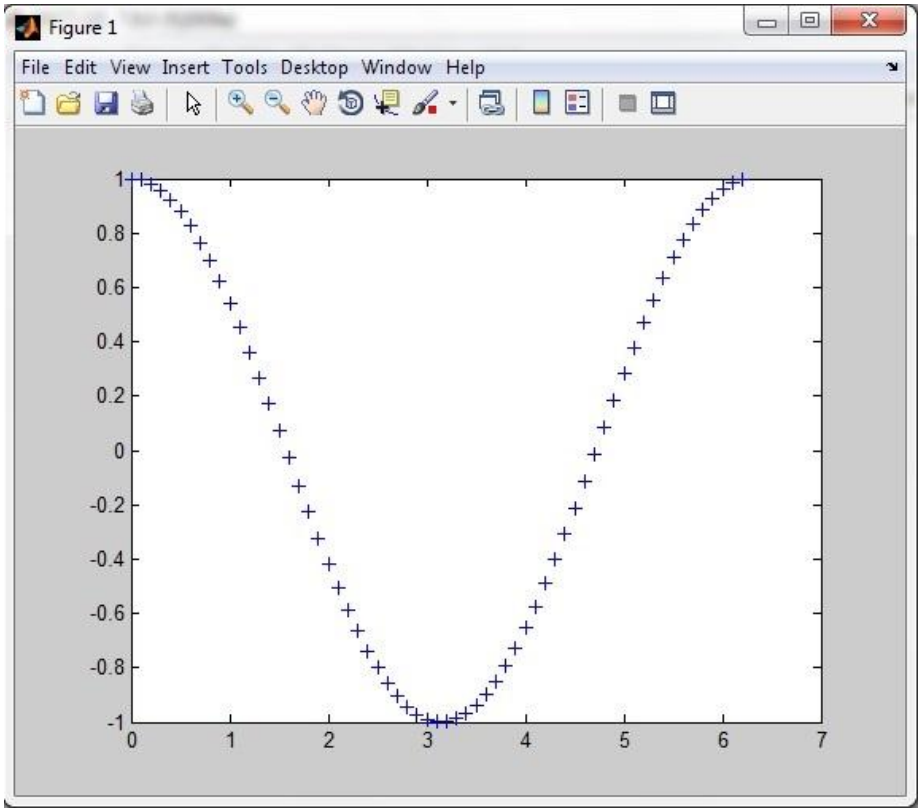


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 1 2 3 4 5 6 7

سطر اول دستور: این سطر از دستور مربوط به تعریف بازه‌ی زمانی نمودار می‌باشد که بدین صورت تعریف می‌گردد.

t= a:b:c که در این دستور:

a: ابتدای بازه‌ی زمانی نمودار می‌باشد

c: انتهای بازه‌ی زمانی نمودار می‌باشد که در این سه مثال برابر بار **۲π** است.

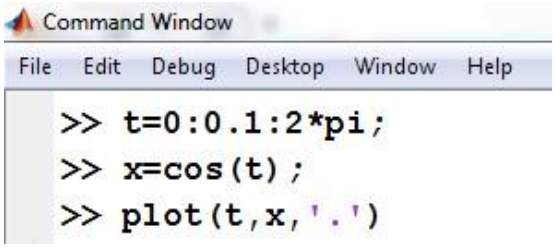
b: نیز فاصله‌ی نقاط نمودار از **a** تا **c** می‌باشدکه در این مثال‌ها برابر با **0.1** است.

سطر دوم دستور: این سطر از دستور نیز، نوع تابعی که رسم خواهد شد را نشان می‌دهد که در این مثال برابر با **cos(t)** می‌باشد.

سطر سوم دستور: در این سطر از دستور نیز، دستور رسم تابع و همچنین شیوه رسم نمودار بیان می‌گردد.

برای تمرین بیشتر، از چند مثال دیگر نیز استفاده می‌کنیم.

مثال:


Command Window
File Edit Debug Desktop Window Help
>> t=0:0.1:2*pi;
>> x=cos(t);
>> plot(t,x,'.')

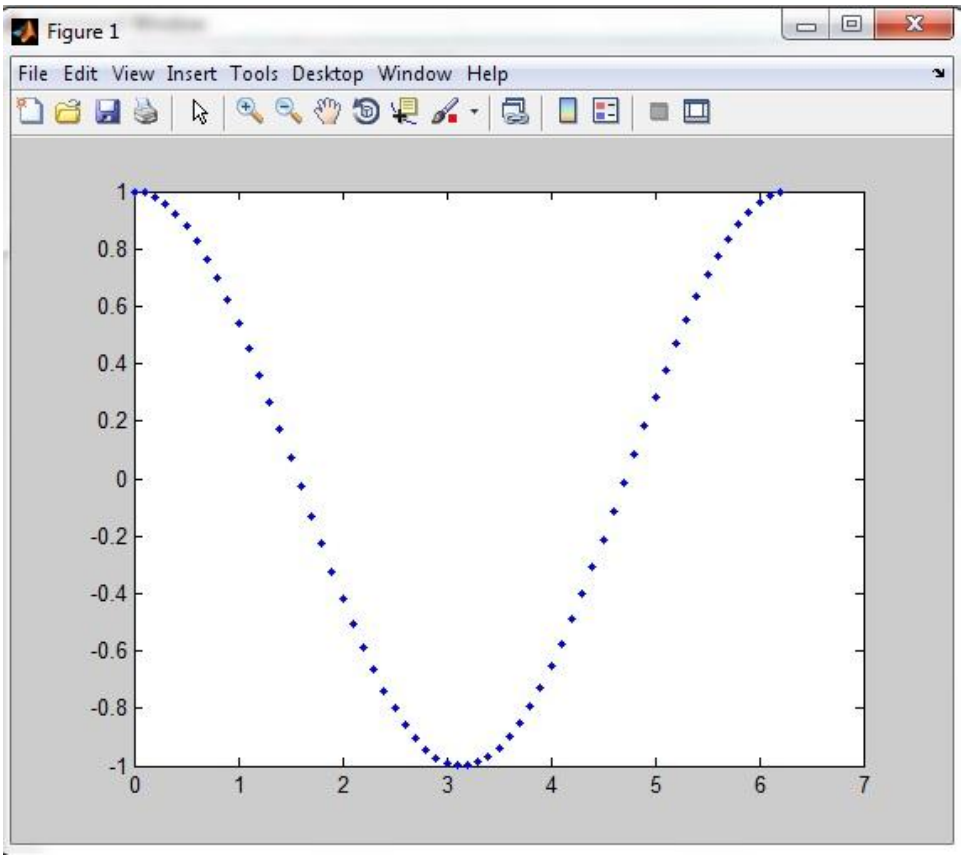
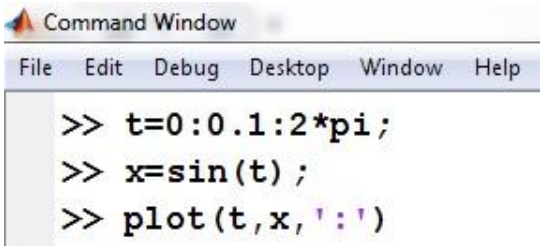


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 1 2 3 4 5 6 7

مثال:


Command Window
File Edit Debug Desktop Window Help
>> t=0:0.1:2*pi;
>> x=sin(t);
>> plot(t,x,':')

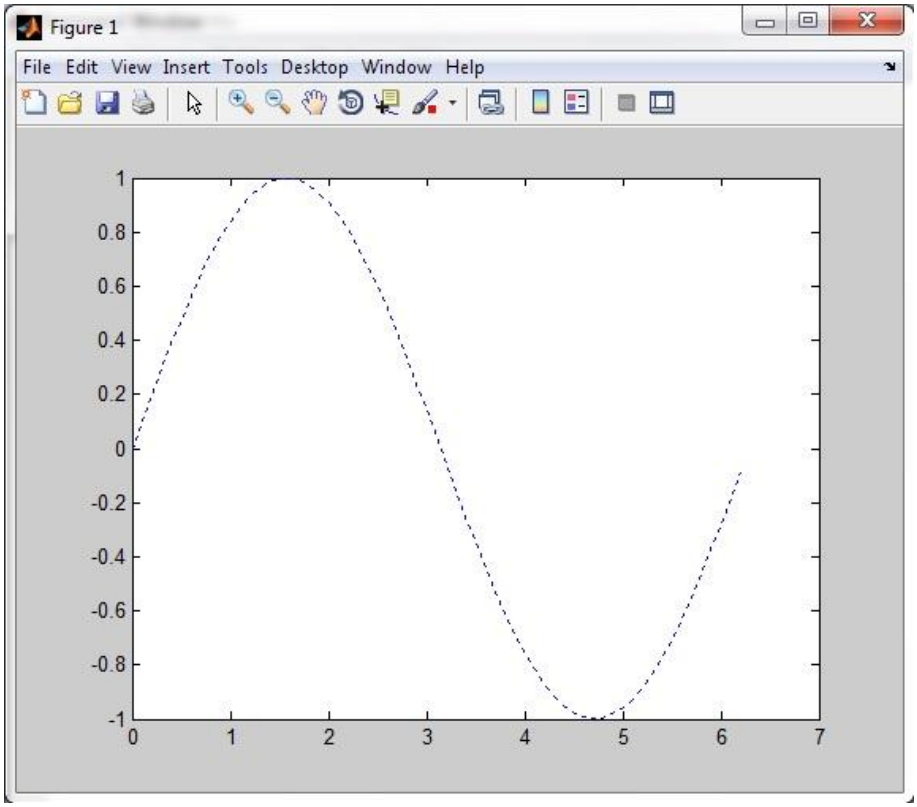
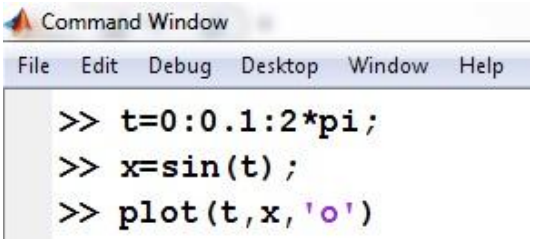


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 1 2 3 4 5 6 7

مثال:


Command Window
File Edit Debug Desktop Window Help
>> t=0:0.1:2*pi;
>> x=sin(t);
>> plot(t,x,'o')

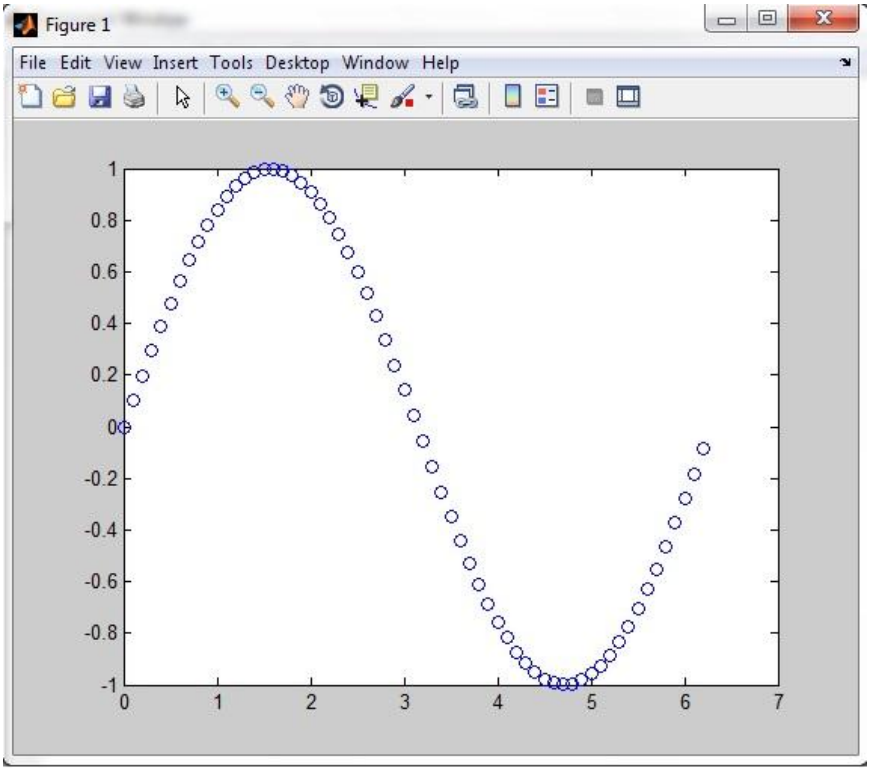


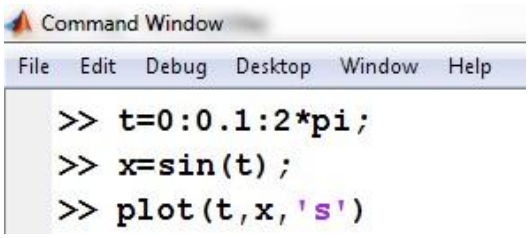
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 1 2 3 4 5 6 7

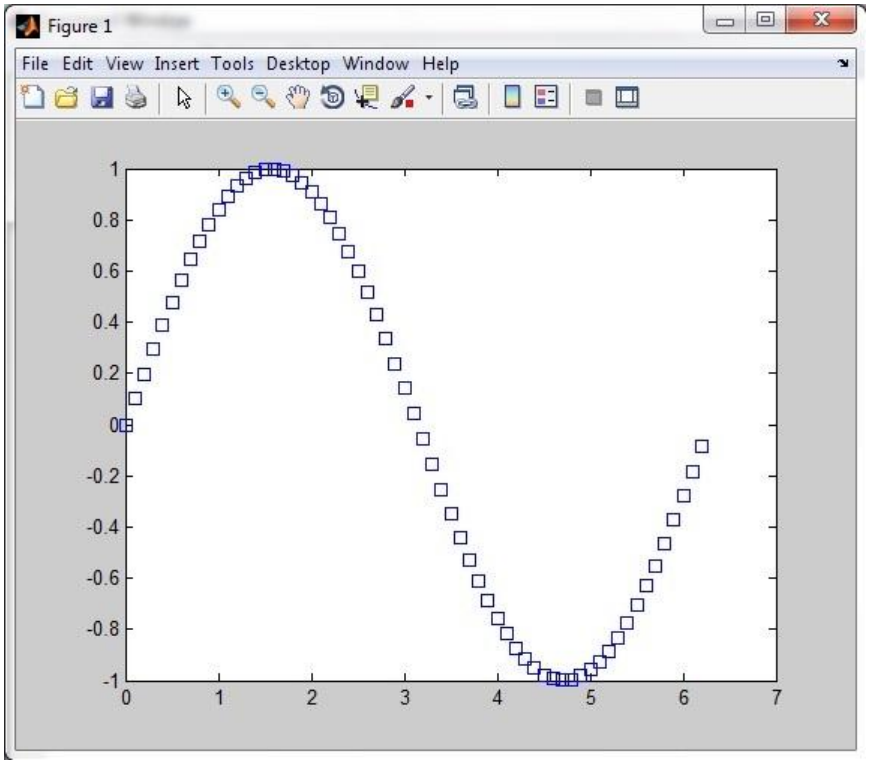
⇦نکته:

علاوه بر اشکال استفاده شده در مثال‌های بالا, از کدهای دستوری نیز می‌توان طبق جدول زیر استفاده کرد.

مربع	**square**	**s**
لوزی	**diamond**	**d**
ستاره‌ی پنج رأسی	**pentagram**	**p**
ستاره‌ی شش رأسی	**hexagram**	**h**

مثال:





⇦نکته:

در این مثال **s** حرف اول عبارت **square** به معنی مربع می‌باشد.

⇦نکته:

در ترسیم توابع گرافیکی، علاوه بر تعیین شکل نمودار، می‌توان رنگ نمودار را نیز تعیین کرد. برای این امر کافی است، در دستور ترسیم نمودار، کد رنگ دلخواه را در کنار شکل یا کد نمودار دلخواه وارد کرد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> t=0:0.1:2*pi;
>> x=sin(t);
>> plot(t,x,'*r')
```

در این دستور، نمودار تابع سینوسی، با شکل ستاره و رنگ قرمز رسم خواهد شد.

مثال:

```
Command Window
File  Edit  Debug  Desktop  Window  Help
>> t=0:0.1:2*pi;
>> x=sin(t);
>> plot(t,x,'dg')
```

در این دستور نیز، نمودار تابع سینوسی، با شکل لوزی و رنگ سبز رسم خواهد شد.

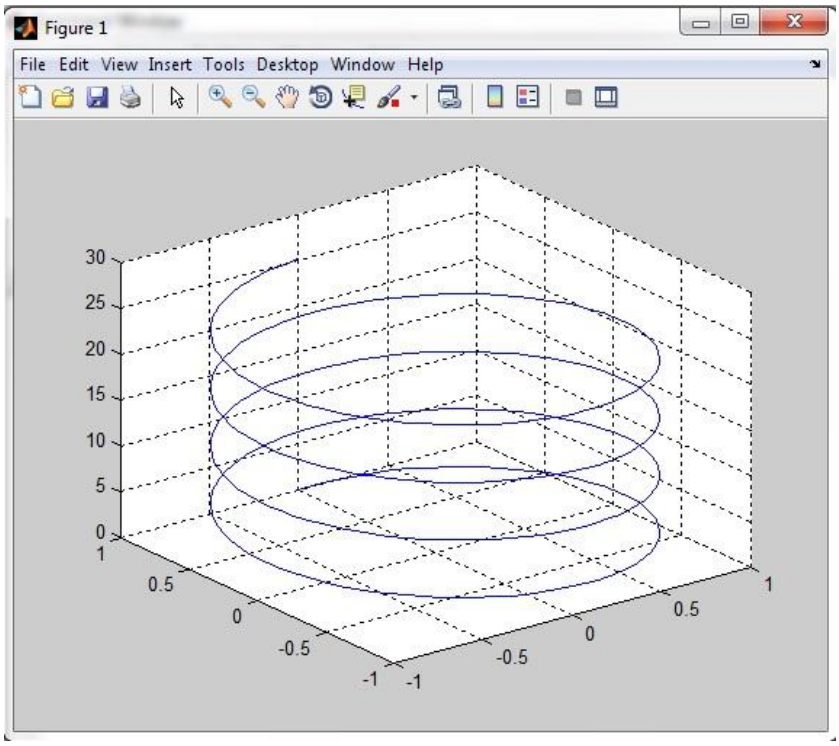
۱۳-۳- ترسیم سه بعدی توابع در MATLAB

✓ دستور plot3

از این دستور برای ترسیم سه بعدی توابع استفاده می‌گردد.

مثال:

```
>> t=0:pi/40:8*pi;
>> x=sin(t);
>> y=cos(t);
>> plot3(x,y,t)
>> grid on
```



پایان این کتاب ...