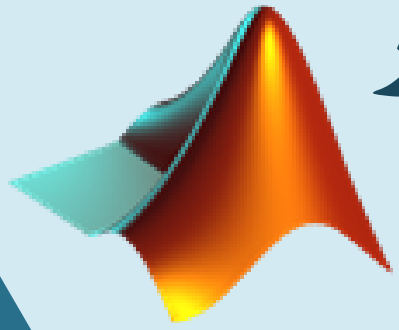
بسمه‌تعالی

بسته‌ی آموزشی ۱



آموزش نرم‌افزار MATLAB

آزمایشگاه مخابرات دیجیتال

استاد: دکتر علی الفت

ویرایش ۱/۱

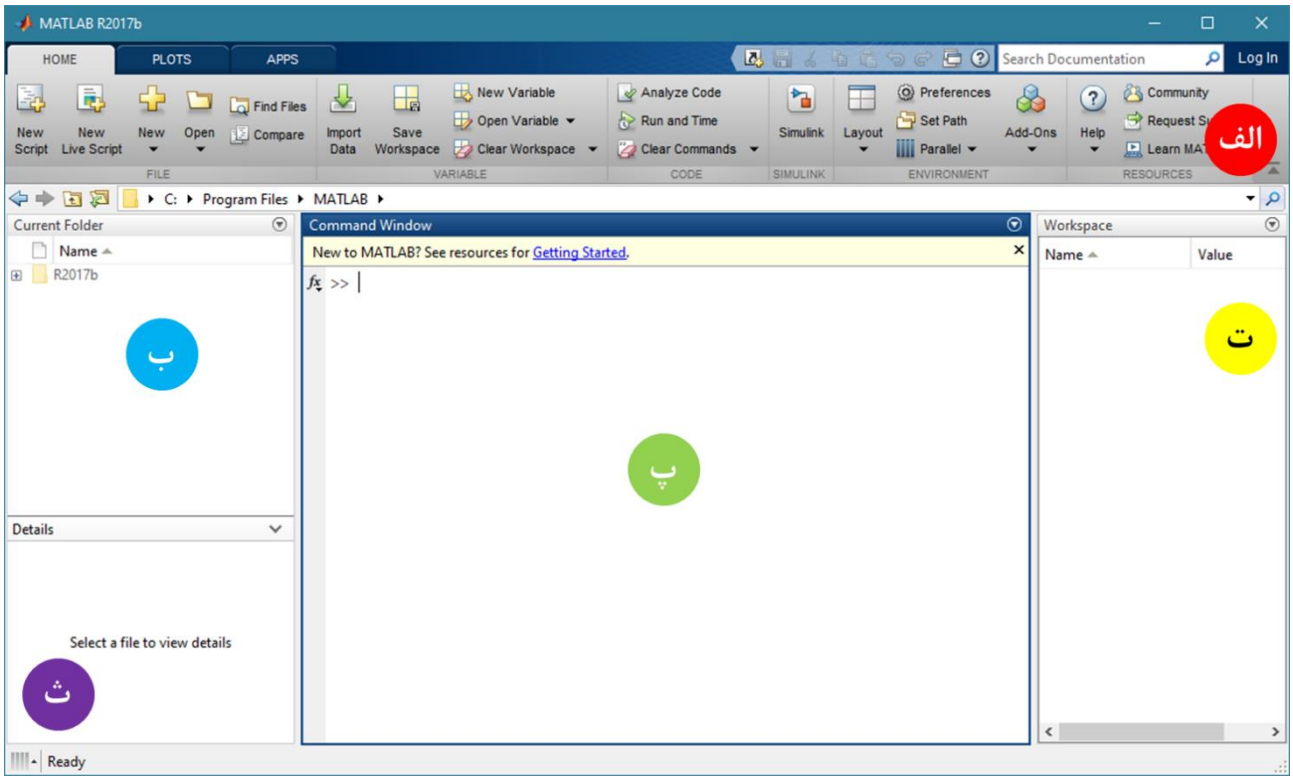
فهرست مطالب

بسته‌ی آموزشی ۱

آشنایی با نرم‌افزار MATLAB

آشنایی با محیط نرم‌افزار MATLAB

با اجرای نرم‌افزار MATLAB پنجره‌ای مشابه پنجره‌ی شکل ۱ نمایش داده خواهد شد. اجزای اصلی این پنجره شامل منوی نواری[1]، پوشه‌ی فعلی[2]، پنجره‌ی دستور، فضای کار[3] و جزییات فایل[4] می‌باشد. با فعال شدن هر یک از این بخش‌ها نوار عنوان بالای هر بخش آبی رنگ می‌شود.



شکل ۱ واسط کاربری نرم‌افزار MATLAB

الف: منوی نواری منوی نواری امکان کار با فایل‌ها و متغیرها، دسترسی به کمک و پشتیبانی (Help and Support)، تعیین تنظیمات نرم‌افزار (Preferences)، تنظیم چیدمان نرم‌افزار(Layout)،اضافه کردن افزونه‌های نرم‌افزاری و سخت‌افزاری (Add-Ons) و ایجاد فایل‌های برنامه‌نویسی مختلف را فراهم می‌آورد.

ب: پوشه‌ی فعلی در بخش پوشه‌ی فعلی، فایل‌های موجود در پوشه‌ای فعلی را نشان داده می‌شود. در این‌جا می‌توان فایل‌های را باز، جابه‌جا، تغییر نام و حذف نمود.

1 Menu Ribbon
2 Current Folder
3 Workspace
4 Details

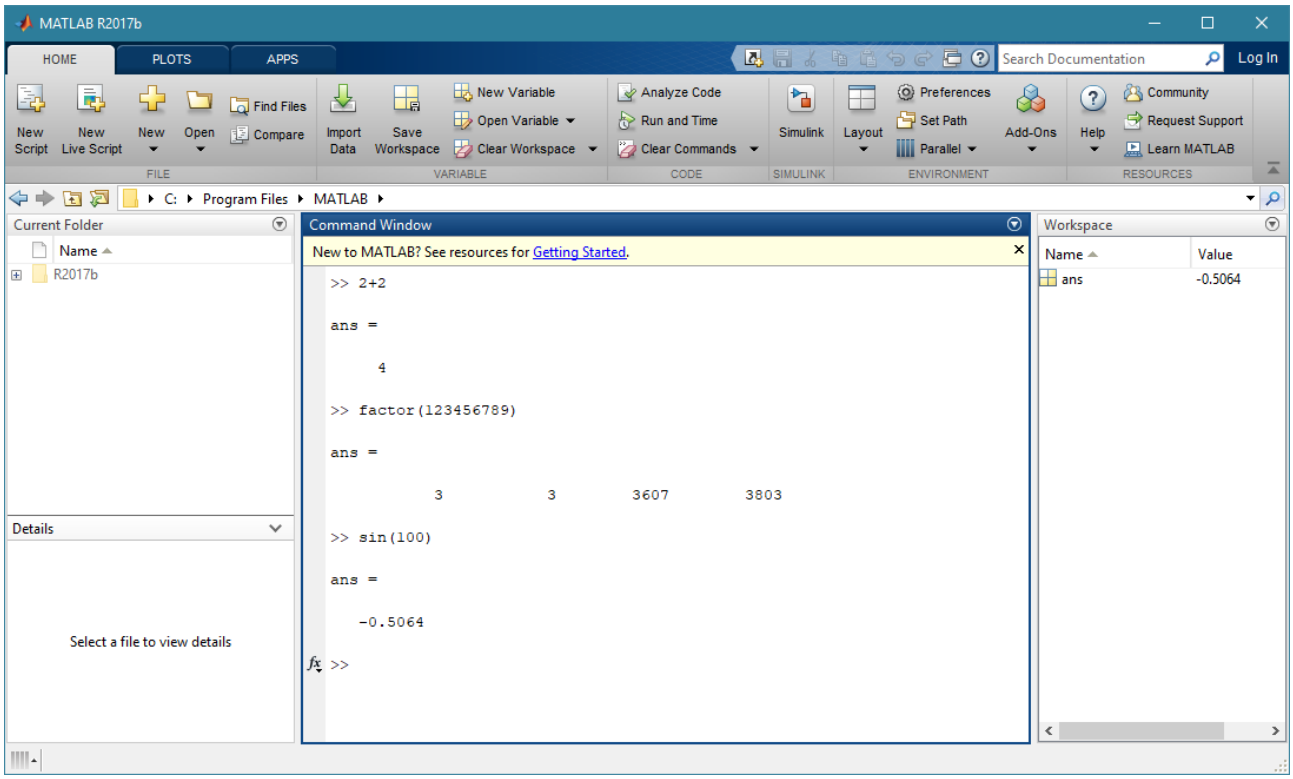
پ: پنجره‌ی دستور در این پنجره می‌توان دستورهای نرم‌افزار MATLAB شامل عبارت‌های کوتاه و چندخطی کوتاه، اجرای نگاره[۵]ها و فراخوانی توابع را به صورت مستقیم وارد نمود. با زدن دکمه‌ی Enter و رفتن به سطر بعد، دستور اجرا می‌شود.

ت: فضای کاری فضای کاری همه‌ی متغیرهایی (اسکالرها، آرایه‌ها، ماتریس‌ها، رشته‌ها، ساختارها و ...) که درون حافظه نگه داشته شده است را نمایش می‌دهد. این متغیرها را می‌توان با این پنجره مشاهده و تغییر داد.

ث: جزییات فایل این بخش خلاصه‌ای از فایل انتخاب شده در پوشه‌ی فعلی را نمایش می‌دهد. این خلاصه شامل تاریخچه‌ی نسخه‌ها، توصیف یا پیش‌نمایش فایل می‌باشد.

نوشتن در پنجره‌ی دستور

با کلیک بر روی پنجره‌ی دستور می‌توان آن را فعال نمود. وقتی یک پنجره فعال می‌شود نوار عنوان آن تیره می‌شود و یک نشان‌گر چشمک‌زن درون پنجره‌ی دستور ظاهر می‌شود. حال می‌توان دستورات را وارد نمود. با وارد کردن دستور و فشردن دکمه‌ی Enter بعد از آن، دستور اجرا می‌شود. نتیجه‌ی وارد کردن دستورهای **2+2**، **factor(123456789)** و **sin(100)** در شکل ۲ آمده است.



شکل ۲ محیط نرم‌افزار MATLAB با چند دستور اجرا شده

بخش راهنمایی (Help)

نرم‌افزار MATLAB دارای یک راهنمای بسیار گسترده است. برای دسترسی به این بخش روش‌های مختلفی وجود دارد. با وارد کردن عبارت **help** در پنجره‌ی دستور، یک لیست طولانی از موضوعاتی که برای آن راهنمایی وجود دارد، ظاهر می‌شود. با وارد کردن **help general** می‌توان به لیستی از دستورهای عمومی نرم‌افزار دست یافت. حال اگر بنا باشد اطلاعاتی در مورد یک دستور یا تابع مشخص نظیر **factor** به دست آید می‌بایست عبارت **help factor** وارد شود. نتیجه‌ی اجرای این دستور به صورت زیر می‌باشد.

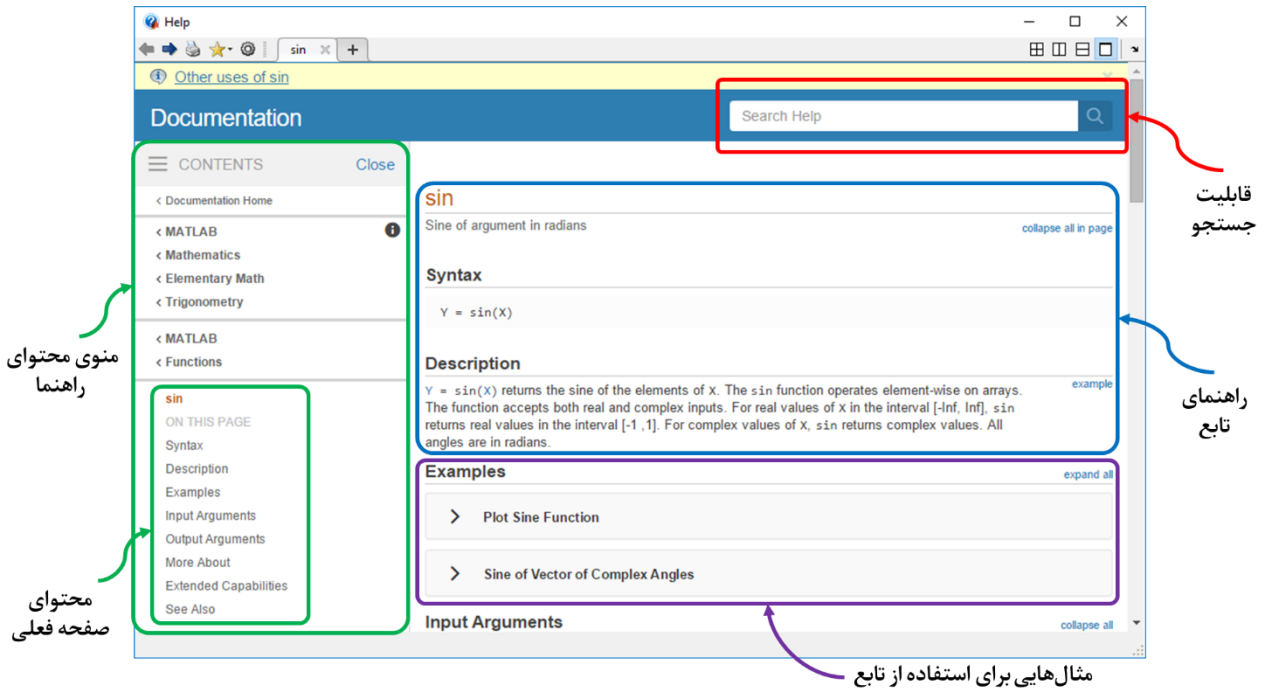
```
>> help factor
 factor Prime factors.
    factor(N) returns a vector containing the prime factors of N.

    This function uses the simple sieve approach. It may require large
    memory allocation if the number given is too big. Technically it is
    possible to improve this algorithm, allocating less memory for most
```

5 Script

```
cases and resulting in a faster execution time. However, it will still
have problems in the worst case.

Class support for input N:
   float: double, single
   integer: uint8, int8, uint16, int16, uint32, int32, uint64, int64

See also primes, isprime.

Reference page for factor
Other functions named factor
```

دستور **lookfor** در خط اول همه‌ی فایل‌های راهنمای MATLAB رشته‌ی مشخصی را جست‌جو می‌نماید. برای جستجو در همه‌ی خطوط از **lookfor -all** استفاده می‌شود. به عنوان مثال به منظور دیدن فهرستی از تمام دستورهای MATLAB که در آن کلمه‌ی «factor» استفاده شده است، می‌توان دستور **lookfor factor** را وارد نمود. اگر دستور مورد نظر در لیست مشاهده شد، آن‌گاه می‌توان با استفاده از **help** اطلاعات بیشتری در مورد این دستور آموخت.

در حالی که دستور **help** برای به دست آوردن اطلاعات سریع یک دستور مشخص مفید است، با استفاده از مرورگر راهنمای MATLAB می‌توان به اطلاعات بیشتری دست یافت. می‌توان به روش‌های مختلف این مرورگر را مشاهده نمود. با وارد کردن عبارت **doc** در پنجره‌ی دستور، فشردن دکمه‌ی Help در منوی نواری و سربرگ Home می‌توان وارد مرورگر راهنما شد. روش بسیار مفید دیگر برای استفاده از مرورگر راهنما نوشتن دستوری مانند **doc sin** می‌باشد. این دستور مرورگر راهنما را باز می‌کند و صفحه‌ی مربوط به **sin** را نمایش می‌دهد. این صفحه به طور کلی مشابه داده‌های خروجی دستور **help** است و بعضی اوقات اطلاعات بیشتری را در بر دارد. شکل ۳ نمایی از مرورگر راهنمای نرم‌افزار MATLAB را نشان می‌دهد.



شکل ۳ چیدمان مرورگر راهنمای MATLAB

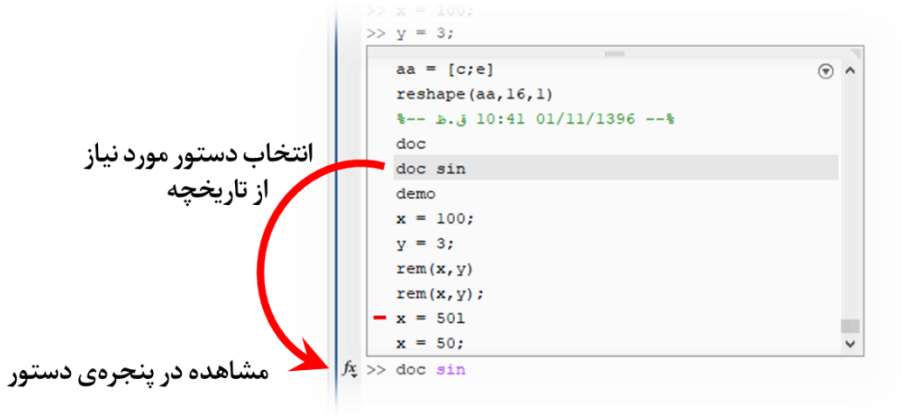
با استفاده از دستور **demo** مثال‌های بسیار جذابی در نرم‌افزار MATLAB را مشاهده خواهید نمود.

استفاده از بخش تاریخچه

نرم‌افزار MATLAB دستورهایی را که در پنجره‌ی دستور اجرا می‌شود را درون حافظه‌ی خود نگه می‌دارد. از این امر می‌توان جهت مشاهده دستورهایی که اخیراً اجرا شده‌اند و یا اجرای مجدد دستور قبل (چه به طور مستقیم و چه با تغییر) به صورت سریع و آسان استفاده نمود. پس از کلیک در پنجره‌ی دستور و مشاهده‌ی نشان‌گر چشمک‌زن، با فشردن دکمه‌ی بالای صفحه‌کلید لیستی از دستورهای قبلی

(تاریخچه‌ی دستورها) نمایش داده می‌شود. حال با استفاده از دکمه‌های بالا و پایین می‌توان دستور مورد نظر را انتخاب نمود و آن را در پنجره‌ی دستور مشاهده نمود. شکل ۴ گویای این مسأله می‌باشد.

اگر دکمه‌ی Enter فشرده شود نرم‌افزار کد را اجرا می‌نماید. با کلیک بر روی پنجره‌ی دستور و یا استفاده از دکمه‌های راست یا چپ صفحه‌کلید می‌توان کد را تغییر داد و سپس برای اجرای کد دکمه‌ی Enter را فشار داد.



شکل ۴ استفاده از تاریخچه‌ی دستورها

اگر هدف یافتن دستور مشخصی با عبارتی خاص در ابتدای آن باشد، ابتدا می‌توان این عبارت را وارد کرد و سپس دکمه‌ی بالای صفحه‌کلید را فشار داد. در این حالت تنها بین دستورهایی که با این عبارت آغاز می‌شود می‌توان جستجو نمود.

محاسبات ریاضی در نرم‌افزار MATLAB

با استفاده از نرم‌افزار MATLAB می‌توان مانند یک ماشین حساب، محاسبات ریاضی را انجام داد. عمل جمع با **+**، عمل تفریق را با **-**، عمل ضرب را با *****، عمل تقسیم را با **/** و عمل توان را با **^** انجام می‌شود. در زیر مثالی آمده است.

```
>> 3^2 - (5 + 4)/2 + 6*3
ans =
   22.5000
```

نرم‌افزار MATLAB جواب را چاپ می‌نماید و حاصل را به متغیری با نام **ans** تخصیص می‌دهد. به جای بازنویسی پاسخ می‌توان از متغیر **ans** برای محاسبات بعدی استفاده نمود. به عنوان مثال اگر بناست مجموع توان دوم و مجذور پاسخ قبل به دست آید به صورت زیر عمل می‌شود.

```
>> ans^2 + sqrt(ans)
ans =
  510.9934
```

همان‌طور که مشاهده می‌شود، MATLAB با هر محاسبه، مقدار جدیدی به **ans** اختصاص می‌دهد. برای محاسبات پیچیده‌تر می‌توان مقادیر محاسبه شده را به متغیرهای دلخواه اختصاص داد. در زیر مثالی از این کار آمده است.

```
>> u = cos(10)
u =
   -0.8391
>> v = sin(10)
v =
   -0.5440
>> u^2 + v^2
ans =
   1
```

نرم‌افزار MATLAB از محاسبات اعشاری با دقت دوبرابر[6] استفاده می‌کند که تقریباً تا ۱۵ رقم، صحیح می‌باشد. با این وجود نرم‌افزار MATLAB به صورت پیش‌فرض تنها ۵ رقم را نمایش می‌دهد. برای نمایش رقم‌های بیشتر می‌توان **format long** را وارد نمود. پس از این دستور تمامی خروجی‌ها رقمی جدید با ۱۵ رقم نشان داده می‌شود. با نوشتن **format short** می‌توان به نمایش ۵ رقمی بازگشت.

6 Double-Precision Floating-Point

محاسبات جبری یا سمبلیک

تفاوت MATLAB با یک ماشین‌حساب در انجام محاسبات دقیق ریاضی است. به عنوان مثال می‌توان کسرهای 1/2 و 1/3 را به صورت سمبلیک با یکدیگر جمع نمود و کسر صحیح 5/6 را به دست آورد. با استفاده از جعبه‌ابزار Symbolic Math نرم‌افزار می‌توان محاسبات جبری و سمبلیک نظیر فاکتورگیری از چندجمله‌ای‌ها یا حل معادلات جبری را انجام داد. برای اطمینان از نصب این جعبه‌ابزار می‌توان عبارت **help symbolic** را وارد نمود.

برای انجام محاسبات سمبلیک، می‌بایست برای تعریف متغیرهایی که بناست به عنوان متغیر سمبلیک استفاده شوند از دستور **syms** استفاده نمود. مجموعه دستورهای زیر به درک بیشتر این امر کمک می‌نماید.

```
>> syms x y
>> (x - y)*(x - y)^2
ans =
(x-y)^3
>> expand(ans)
ans =
x^3-3*x^2*y+3*x*y^2-y^3
>> factor(ans)
ans =
(x-y)^3
```

MATLAB معمولاً ساده‌سازی‌های جزیی بر روی عبارات وارد شده انجام می‌دهد و تا زمانی که به نرم‌افزار گفته نشود، تغییرات عمده‌ای اعمال نمی‌نماید. دستور **expand** باعث می‌شود که عبارت‌ها در هم ضرب شوند و دستور **factor** از عبارت‌ها فاکتورگیری می‌نماید. MATLAB دستوری با نام **simplify** دارد که از آن می‌توان برای بیان یک فرمول به ساده‌ترین شکل ممکن استفاده نمود. مثالی از این دستور در زیر آمده است.

```
>> simplify((x^3 - y^3)/(x - y))
ans =
x^2+x*y+y^2
```

جایگذاری در عبارت‌های سمبلیک

زمان استفاده از عبارت‌های سمبلیک، معمولاً لازم می‌شود یک مقدار عددی یا حتی یک عبارت سمبلیک دیگر را جایگزین یک یا چند متغیر اولیه در عبارت کرد. این امر با استفاده از دستور **subs** انجام می‌شود. به عنوان مثال با فرض این‌که عبارت سمبلیک **w** تعریف شده است و شامل متغیر سمبلیک **u** می‌باشد، می‌توان با دستور **subs(w, u, 2)** مقدار 2 را جایگزین متغیر **u** در عبارت **w** می‌نماید. برای این امر مثالی ارایه می‌شود.

```
>> d = 1, syms u v
d =
   1
>> w = u^2 - v^2
w =
u^2-v^2
>> subs(w, u, 2)
ans =
4-v^2
>> subs(w, v, d)
ans =
u^2-1
>> subs(w, v, u + v)
ans =
u^2-(u+v)^2
>> simplify(ans)
ans =
-2*u*v-v^2
```

محاسبات عبارت‌های سمبلیک، دقت متغیر و محاسبات دقیق

برای درک بهتر محاسبات اعشاری و سمبلیک MATLAB مثال زیر بیان می‌شود.

```
>> cos(pi/2)
ans =
    6.1232e-17
```

پاسخ به صورت یک عدد اعشار نوشته شده است و برابر 6.1232×10^{-17} می‌باشد. با این وجود مشخص است که $\cos(\pi/2)$ برابر با 0 است. علت این تفاوت آن است که با نوشتن **pi**، نرم‌افزار MATLAB به جای مقدار دقیق، تقریبی از عدد π را تا ۱۵ رقم به دست می‌دهد. برای محاسبه جواب دقیق می‌توان از بیان سمبلیک $\pi/2$ به صورت **str2sym('pi/2')** استفاده نمود. حال اگر کسینوس این عبارت را حساب نمایید جواب دقیق و مورد انتظار به دست خواهد آمد.

```
>> cos(str2sym('pi/2'))
ans =
0
```

پریم‌های اطراف **pi/2** در عبارت **str2sym('pi/2')** یک رشته شامل کاراکترهای **pi/2** را تولید می‌کند و مانع از محاسبه‌ی **pi/2** به صورت یک عدد اعشاری می‌شود. دستور **str2sym** این رشته را به یک عبارت سمبلیک تبدیل می‌نماید. در این‌جا دو عدد 1/2 و 1/3 را می‌توان به صورت سمبلیک با یکدیگر جمع نمود.

```
>> str2sym('1/2') + str2sym('1/3')
ans =
5/6
```

برای انجام محاسبات با دقت متغیر می‌توان از دستور **vpa** استفاده نمود. به عنوان مثال برای چاپ عدد $\sqrt{2}$ تا ۵۰ رقم اعشار می‌توان به صورت زیر عمل نمود.

```
>> vpa('sqrt(2)', 50))
ans =
1.4142135623730950488016887242096980785696718753769
```

اگر برای تعداد رقم‌ها، عددی مشخص نشود به صورت پیش‌فرض عدد ۳۲ در نظر گرفته می‌شود. این مقدار پیش‌فرض را می‌توان با دستور **digits** تغییر داد.

محاسبات مختلط

نرم‌افزار MATLAB بیشتر محاسبات خود را به صورت مختلط انجام می‌دهد. عدد مختلط i در MATLAB با 1i بیان می‌شود. بعضی از چند جمله‌ای‌ها با ضرایب حقیقی، دارای جواب مختلط هستند و هم‌چنین وابسته به آرگومان بعضی از تابع‌ها نیز خروجی مختلط دارند.

```
>> log(-1)
ans =
   0.0000 + 3.1416i
```

با استفاده از نرم‌افزار MATLAB می‌توان با وارد کردن اعدادی به صورت a + b*1i محاسبات مختلط را انجام داد.

```
>> (2 + 3*1i)*(4 - 1i)
ans =
  11.0000 +10.0000i
```

محاسبات مختلط قابلیت قدرت‌مند و باارزشی است. حتی اگر بنا نیست از اعداد مختلط استفاده شود می‌بایست توجه داشت که جواب‌های نرم‌افزار MATLAB می‌تواند به صورت مختلط باشد.

بردارها و ماتریس‌ها

نرم‌افزار MATLAB در ابتدا برای این منظور نوشته شده بود که ریاضی‌دانان، دانشمندان و مهندسان به راحتی بتوانند از ابزارهای جبرخطی مانند بردارها و ماتریس‌ها بهره ببرند. در ادامه این مفاهیم معرفی می‌شود.

بردارها

بردار، یک لیست دارای ترتیب از اعداد می‌باشد. در MATLAB می‌توان یک بردار با طول دلخواه را با نوشتن اعدادی که با درنگ‌نما(,) و یا فاصله جدا شده و درون کروشه ([]) قرار گرفته، ایجاد نمود. در ادامه مثالی آمده است.

```
>> Z = [2, 4, 6, 8]
Z =
     2     4     6     8
```

```
>> Y = [4 -3 5 -2 8 1]
Z =
     4     -3      5     -2      8      1
```

اگر بنا باشد برداری از اعداد ۱ تا ۹ ایجاد شود، به جای ورود تک‌تک اعداد می‌توان به صورت زیر عمل نمود.

```
>> X = 1:9
X =
     1     2     3     4     5     6     7     8     9
```

عبارت **1:9** یک بردار از اعداد ۱ تا ۹ با گام‌های ۱ ایجاد می‌نماید. می‌توان گام را به صورت زیر مشخص کرد.

```
>> X = 0:2:10
X =
     0     2     4     6     8    10
```

گام می‌تواند به صورت کسری و یا منفی نیز باشد. به عنوان مثال می‌توان به **0:0.1:1** و **100:-1:0** اشاره نمود.

درایه‌های برداری مانند **X** را می‌توان به صورت **X(1)**، **X(2)** و ... به دست آورد. به عنوان نمونه داریم.

```
>> X(3)
ans =
     4
```

برای تبدیل بردار سطری **X** به یک بردار ستونی از یک پریم (**'**) بعد از **X** استفاده می‌شود.

```
>> X'
ans =
     0
     2
     4
     6
     8
    10
```

بر روی بردارها می‌توان عملیات ریاضی انجام داد. به عنوان مثال می‌توان همه‌ی درایه‌های بردار **X** را به توان دو رساند.

```
>> X.^2
ans =
     0     4    16    36    64   100
```

دقت شود که نقطه در این عبارت اهمیت بسیار زیادی دارد. این نقطه بیان می‌کند اعداد درون **X** بایستی به صورت درایه به درایه به توان ۲ برسند. به طور مشابه برای ضرب یا تقسیم دو بردار به صورت درایه به درایه می‌بایست از **.*** یا **./** استفاده نمود. به عنوان مثال برای ضرب المان‌های بردار **X** در المان‌های متناظر بردار **Y** می‌بایست به صورت زیر عمل کرد.

```
>> X.*Y
ans =
     0    -6    20   -12    64    10
```

بیشتر عملگرهای MATLAB به صورت پیش‌فرض، درایه به درایه عمل می‌نمایند. به عنوان نمونه لازم نیست قبل از عملگر جمع و تفریق از نقطه استفاده نمایید. هم‌چنین می‌توان برای محاسبه خروجی تابع نمایی برای هر درایه‌ی **X** از **exp(X)** استفاده نمود.

ماتریس‌ها

ماتریس 4×3 زیر را در نظر بگیرید.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

این ماتریس را می‌توان به صورت زیر در نرم‌افزار MATLAB وارد نمود.

```
>> A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

توجه داشته باشید که درایه‌های ماتریس در هر ردیف با درنگ‌نما(**,**) و ردیف‌ها با نقطه‌درنگ‌نما (**;**) جدا می‌شوند. المان‌های هر ردیف می‌توانند با فاصله نیز از یکدیگر جدا شوند.

اگر دو ماتریس **A** و **B** هم‌اندازه باشند، جمع درایه به درایه‌ی آن‌ها را می‌توان با وارد کردن **A + B** انجام داد. هم‌چنین می‌توان یک عدد اسکالر را به ماتریس اضافه نمود. **A + c** عدد **c** را به همه‌ی درایه‌های ماتریس **A** اضافه می‌نماید. به طور مشابه **A – B** اختلاف دو ماتریس **A** و **B** را محاسبه کرده و **A – c** عدد **c** را از همه‌ی درایه‌های **A** کم می‌کند. اگر ماتریس **A** و **B** را بتوان در هم ضرب نمود، حاصل این ضرب را می‌توان به صورت **A*B** به دست آورد. ضرب عدد **c** در ماتریس **A** به صورت **c*A** می‌باشد و **A'** بیان‌کننده‌ی ترانهاده‌ی مختلط ماتریس **A** می‌باشد.

یک مثال ساده از ضرب ماتریسی را می‌توان با ضرب ماتریس **A** با ابعاد 4×3 و بردار ستونی **Z'** با اندازه‌ی 1×4 به صورت زیر نشان داد که منجر به یک ماتریس ستونی 1×3 می‌شود.

```
>> A*Z'
ans =
     60
    140
    220
```

اگر **f** یکی از توابع ریاضی داخلی نرم‌افزار MATLAB باشد، **f(A)** ماتریسی است که با اعمال **f** به تک تک درایه‌های **A** به دست می‌آید. به عنوان مثال حاصل **sqrt(A)** ماتریسی با ابعاد 4×3 خواهد بود که درایه‌های آن جذر درایه‌های متناظر ماتریس **A** است.

می‌دانیم که **x(3)** المان سوم بردار **x** را برمی‌گرداند. به صورت مشابه **A(2,3)** بیان‌گر درایه‌ی سطر دوم و ستون سوم ماتریس **A** می‌باشد. می‌توان از همین طریق زیرماتریس‌های یک ماتریس را نیز به دست آورد. با نوشتن **A(2,[2 4])** دومین و چهارمین درایه‌ی سطر دوم ماتریس **A** حاصل می‌شود. برای انتخاب المان‌های دوم، سوم و چهارم این سطر می‌توان **A(2,2:4)** را وارد نمود. زیرماتریس شامل درایه‌های سطر ۲ و ۳ و ستون ۲، ۳ و ۴ با نوشتن **A(2:3,2:4)** به دست می‌آید. علامت دو نقطه به تنهایی مشخص‌کننده‌ی یک سطر یا ستون کامل می‌باشد. به عنوان مثال **A(:,2)** نشان دهنده‌ی ستون دوم ماتریس **A** و **A(3,:)** نشان‌دهنده‌ی سطر سوم ماتریس **A** می‌باشد.

نرم‌افزار MATLAB دستورهای زیادی برای ایجاد ماتریس‌های خاص دارد. دستورهای **zeros(n,m)** و **ones(n,m)** به ترتیب ماتریس تمام‌صفر و تمام‌یک با ابعاد $n \times m$ تولید می‌نماید. همچنین **eye(n)** بیان‌گر یک ماتریس همانی $n \times n$ است.

عدم چاپ خروجی

بعضی از دستورهای MATLAB خروجی‌های تولید می‌نماید که ممکن است زاید باشد. به عنوان مثال زمانی که به یک متغیر مقداری تخصیص داده می‌شود، MATLAB این مقدار را چاپ می‌نماید. با نوشتن یک نقطه‌درنگ‌نما (;) در انتهای یک خط ورودی، می‌توان از چاپ خروجی دستور MATLAB ممانعت نمود. در زیر مثالی آورده شده است.

```
>> syms x
>> y = x + 7
y =
x + 7

>> z = x + 7;
>> z
z =
x + 7
```

علامت نقطه‌درنگ‌نما به صورت داخلی هیچ تأثیری بر روی پردازش MATLAB ندارد. این امر با مشاهده پاسخ نرم‌افزار به دستور **z** قابل مشاهده می‌باشد.

نقطه‌درنگ‌نما به طور کلی باید در هنگام تعریف بردارها یا ماتریس‌های بزرگ استفاده شود. هم‌چنین می‌توان هر زمان که نیازی به نمایش خروجی MATLAB نیست از این علامت استفاده نمود. این علامت می‌تواند چندین دستور پشت سر هم را از هم جدا نماید و تنها خروجی دستور آخر را نمایش دهد. درنگ‌نما به تنهایی نیز می‌تواند بدون ممانعت از چاپ خروجی دستورها را از هم جدا نماید. اگر از نقطه‌درنگ‌نما بعد از یک دستور گرافیکی استفاده شود، مانع از ایجاد نمودار نمی‌شود.

ممکن است شخصی بخواهد MATLAB نام متغیر و علامت را در خروجی دستور چاپ ننماید. دستور **disp** برای این امر طراحی شده است. نوشتن **disp(x)** مقدار متغیر **x** را بدون نام متغیر و علامت مساوی چاپ می‌نماید.

```
>> x = 7; disp(x)
     7
```

```
>> disp(solve(str2sym('x + tan(y) = 5'), 'y'))
-atan(x-5)
```

تابع‌ها

در نرم‌افزار MATLAB می‌توان از تابع‌های داخلی و تابع‌های شخصی استفاده نمود.

تابع‌های داخلی

MATLAB دارای توابع داخلی بسیاری است. از جمله این توابع می‌توان به **sqrt** (جذر)، **cos** (کسینوس)، **sin**(سینوس)، **tan**(تانژانت)، **log**(لگاریتم طبیعی)، **exp** (تابع نمایی) و **atan** (معکوس تانژانت) می‌باشد. علاوه بر این تابع‌های خاص ریاضی نظیر **gamma**، **erf** و **besselj** را نیز در بر دارد. نرم‌افزار MATLAB ثابت‌های داخلی زیادی نیز نظیر **pi** (عدد π)، 1i (عدد مختلط $i = \sqrt{-1}$) و **inf** (∞) دارد. مثال‌هایی از این موارد در زیر آمده است.

```
>> log(exp(3))
ans =
     3
```

تابع **log** در واقع لگاریتم طبیعی است که در بیشتر کتاب‌ها با ln شناخته می‌شود.

```
>> sin(2*pi/3)
ans =
    0.8660
```

برای به دست آوردن جواب دقیق می‌بایست از آرگومان سمبلیک استفاده نمود.

```
>> sin(str2sym('2*pi/3'))
ans =
1/2*3^(1/2)
```

تابع‌های تعریف‌شده توسط کاربر

در این‌جا روشی برای تعریف تابع شخصی در MATLAB بیان می‌شود. در این روش از عملگر @ می‌شود و تابعی که با این عملگر ایجاد می‌شود، «تابع ناشناس» خوانده می‌شود. تابع‌ها می‌توانند در فایل‌های جداگانه‌ای با نام M-file نیز نوشته شوند. در ادامه با استفاده از عملگر @ تابع $f(x) = x^2$ تعریف می‌شود.

```
>> f = @(x) x^2
f =
    @(x) x^2
```

زمانی که این تابع تعریف شد می‌توان به صورت زیر آن را محاسبه نمود.

```
>> f(4)
ans =
     16
```

همان‌طور که قبل مشاهده شد، اکثر توابع MATLAB می‌توانند علاوه بر اسکالرها بر روی بردارها نیز اعمال شوند. برای اطمینان از این‌که تابع تعریف‌شده توسط کاربر بتواند بر روی بردارها عمل نماید، قبل از عملگرهای ریاضی *، / و ^ می‌بایست نقطه‌ای اضافه شود. بنابراین برای به دست آوردن نسخه برداری تابع $f(x) = x^2$ به صورت زیر عمل می‌شود.

```
>> f = @(x) x.^2
```

حال می‌توان این تابع را بر روی یک بردار نیز اعمال نمود.

```
>> f(1:5)
ans =
     1     4     9    16    25
```

تابع‌ها را می‌توان با دو یا تعداد بیشتری متغیر تعریف نمود. تابع زیر بر روی بردارها نیز عمل می‌نماید.

```
>> g = @(x, y) x.^2 + y.^2;
>> g(1, 2)
ans =
     5
>> g([1 2], [3 4])
ans =
     10     20
```

آخرین عبارت در بالا مقدار تابع را در نقاط (1,3) و (2,4) برمی‌گرداند.

کار با متغیرها

در نرم‌افزار MATLAB می‌توان از علامت مساوی برای اختصاص یک عدد به یک متغیر استفاده نمود. به عنوان مثال داریم.

```
>> x = 7
x =
     7
```

این عبارت از این به بعد به متغیر **x** مقدار 7 را نسبت می‌دهد. از این به بعد هر زمان MATLAB حرف **x** را ببیند آن را با 7 جایگزین می‌نماید. به عنوان مثال اگر **y** یک متغیر سمبلیک باشد، خواهیم داشت.

```
>> x^2 - 2*x*y + y
ans =
49-13*y
```

نام یک متغیر یا تابع می‌تواند هر رشته‌ای از حروف، رقم و خط زیرین باشد و می‌بایست با حرف شروع شود. نمی‌توان در این نام از علامت‌های نشانه‌گذاری استفاده نمود. نرم‌افزار MATLAB بین حروف بزرگ و کوچک تمایز قایل می‌شود. اسامی را می‌بایست به نحوی انتخاب شود که به راحتی به خاطر سپرده شود. معمولا از حروف کوچک به همراه خط زیرین برای این امر استفاده می‌شود.

در یک برنامه‌ی بزرگ MATLAB به سختی می‌توان نام همه‌ی متغیرها و نوع آن‌ها را به خاطر سپرد. دستور **whos** خلاصه‌ای از نام و نوع متغیرهای تعریف شده را مشخص می‌نماید. پنجره‌ی فضای کار لیستی از متغیرها را به صورت گرافیکی انجام می‌دهد.

به منظور پاک کردن همه‌ی متغیرهای تعریف شده می‌توان با وارد کردن **clear** یا **clear all** این کار را انجام داد. برای حذف متغیر **x** و **y** عبارت **clear x y** را می‌توان وارد کرد. می‌بایست معمولاً متغیرها را قبل از شروع محاسبات جدید حذف نمود. در غیر این صورت مقادیر محاسبات قبل می‌تواند به صورت تصادفی در محاسبات جدید دخیل شود.

حل معادله و دستگاه معادله

حل غیر ماتریسی

می‌توان معادلات شامل متغیرها را با استفاده از دستورهای **solve** و **fzero** حل نمود. به عنوان مثال برای یافتن جواب معادله $x^2 - 2x - 4 = 0$ به صورت زیر عمل می‌شود که در نسخه‌ی جدید نرم‌افزار MATLAB به این طریق پیشنهاد شده است.

```
>> syms x; solve(x^2 - 2*x - 4 == 0, x)
ans =
5^(1/2)+1
1-5^(1/2)
```

جواب فوق شامل جواب‌های سمبلیک دقیق $1 \pm \sqrt{5}$ است. برای رسیدن به جواب‌های عددی می‌توان از **double(ans)** یا برای داشتن تعداد رقم دلخواه از **vpa(ans)** استفاده نمود.

دستور **solve** می‌توان معادلات چندجمله‌ای مرتبه‌ی بالاتر و هم‌چنین معادله‌های دیگری را نیز حل می‌نماید. هم‌چنین با این دستور می‌توان معادله‌های دارای بیش از یک متغیر را حل نمود. اگر تعداد معادلات کمتر از تعداد متغیرهاست می‌بایست متغیر یا متغیرهایی که بناست پاسخ بر اساس آن به دست آید را نوشت. به عنوان مثال با نوشتن **solve(2*x - log(y) == 1, y)** می‌توان در $2x - \log y = 1$ مقدار y را بر حسب x به دست آورد. می‌توان معادلات بیشتری را نیز تعریف نمود. در زیر مثالی ارایه می‌شود.

```
>> [x y] = solve(x^2 - y == 2, y - 2*x == 5)
x =
 1 - 2*2^(1/2)
 2*2^(1/2) + 1
y =
 7 - 4*2^(1/2)
 4*2^(1/2) + 7
```

این دستگاه معادلات دارای دو پاسخ است. MATLAB پاسخ‌ها را با ارایه‌ی دو مقدار برای x و دو مقدار برای y این جواب‌ها را نشان می‌دهد. اولین جواب شامل اولین مقدار x و اولین مقدار y می‌باشد.

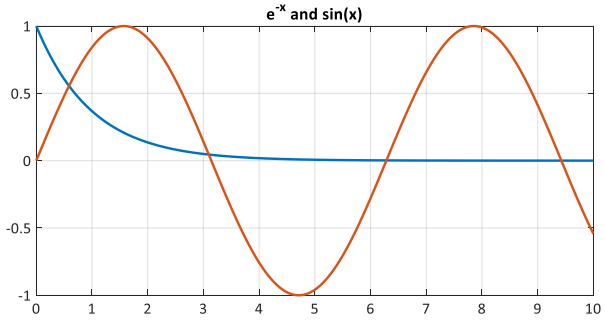
بعضی از معادله‌ها را نمی‌توان به صورت سمبلیک حل نمود و در این حالت دستور **solve** سعی می‌کند یک پاسخ عددی را پیدا کند. به عنوان مثال داریم.

```
>> syms x; solve(sin(x) == 2 - x)
ans =
1.106060157706271910616737297030 1
```

می‌توان با استفاده از دستور **fzero** جواب‌های تقریبی یک معادله را به دست آورد. اگر معادله چند جواب داشته باشد با مشخص کردن یک مقدار مشخص جواب نزدیک به این مقدار به دست می‌آید. جواب معادله‌ی $e^{-x} = \sin(x)$ صفر تابع $e^{-x} - \sin(x)$ می‌باشد. بنابراین برای یافتن جواب تقریبی آن در نزدیک $x = 0.5$ به صورت زیر عمل می‌شود.

```
>> h = @(x) exp(-x) - sin(x);
>> fzero(h, 0.5)
ans =
    0.5885
```

اگر به جای 0.5 عدد 3 قرار گیرد جواب بعدی به دست می‌آید.



شکل ۵ رسم همزمان دو تابع e^{-x} و $\sin(x)$ جهت تخمین نقاط تقاطع

حل ماتریسی

فرض کنید ماتریس **A** یک ماتریس غیرتکین $n \times n$ و b یک بردار ستونی با طول n باشد. عبارت **x = A\b** به صورت عددی جواب یکتای معادله‌ی **A*x = b** را محاسبه می‌نماید. برای اطلاعات بیشتر می‌توان عبارت **help mldivide** را وارد نمود.

اگر حداقل یکی از **A** یا **b** سمبلیک باشد، آن‌گاه **x = A\b** جواب معادله‌ی **A*x = b** را به صورت سمبلیک محاسبه می‌نماید. اگر بناست با وجود این‌که **A** و **b** هر دو عددی است، جواب سمبلیک معادله به دست آید، می‌توان عبارت **x = sym(A)\b** را وارد نمود.

گرافیک

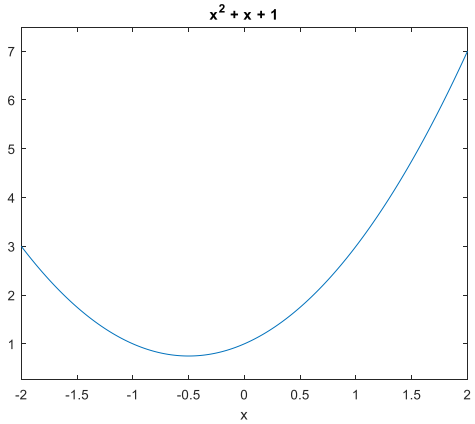
در این‌جا دو دستور پایه‌ای نرم‌افزار MATLAB و نحوه‌ی استفاده از آن‌ها معرفی می‌شود.

رسم نمودار با استفاده از ezplot

ساده‌ترین راه برای رسم یک تابع یک متغیره استفاده از دستور **ezplot** است که یک رشته، عبارت سمبولیک و یا تابع ناشناس توصیف‌کننده‌ی تابع مورد نظر را رسم می‌نماید. به عنوان مثال برای رسم $x^2 + x + 1$ در بازه‌ی $[-2,2]$ با استفاده از **ezplot** با ورودی رشته، عبارت زیر نوشته می‌شود.

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

نمودار مربوط به عبارت بالا در شکل ۶ رسم شده است.



شکل ۶ رسم نمودار $x^2 + x + 1$ با استفاده از دستور ezplot

با استفاده از یک عبارت سمبلیک نیز می‌توان نمودار قبل را به صورت زیر بازتولید نمود.

```
>> syms x; ezplot(x^2 + x + 1, [-2 2])
```

این کار را می‌توان با استفاده از یک تابع ناشناس به عنوان آرگومان **ezplot** نیز انجام داد.

```
>> ezplot(@(x) x.^2 + x + 1, [-2 2])
```

تغییر نمودارها

می‌توان یک نمودار را با روش‌های مختلفی تغییر داد. می‌توان عنوان نمودار را با نوشتن عبارت زیر در پنجره‌ی دستور تغییر داد.

```
>> title('A Parabola')
```

می‌توان با استفاده از دستور **ylabel** بر روی محور عمودی برچسب اضافه نمود. هم‌چنین با استفاده از دستور **xlabel** برچسب محور افقی را تغییر داد. با استفاده از دستور **axis** می‌توان محدوده‌ی افقی و عمودی نمایش نمودار را تغییر داد. به عنوان مثال برای تنظیم محدوده‌ی عمودی در بازه‌ی ۰ تا ۳ به نحو زیر عمل می‌شود.

```
>> axis([-1 2 0 3])
```

دو عدد اول محدوده‌ی محور افقی را نشان می‌دهد. حتی اگر بنا باشد یکی از محدوده‌های افقی و عمودی تغییر کند، می‌بایست هر دو محدوده وارد شود.

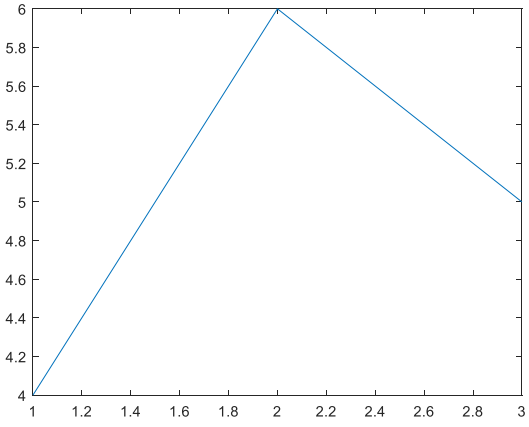
به منظور اینکه شکل نمودار به صورت مربعی درآید، از عبارت **axis square** استفاده می‌شود. این دستور در صورتی که محدوده‌ی x و y طول یکسانی داشته باشد، باعث می‌شود مقیاس هر دو محور یکی شود. برای محدوده‌های با طول دلخواه می‌توان مقیاس هر دو محور را بدون تغییر شکل نمودار،با دستور **axis equal** یکی کرد. به طور کلی این دستور یکی از محورها را متناسب با نیاز گسترش می‌دهد. اگر یک بخشی از محدوده‌ی نمایش نمودار انتخاب شده باشد، MATLAB بخشی که نمایش داده نمی‌شود را حذف نمی‌کند. می‌توان محدوده‌ی محورها را با دستور **axis** تنظیم نمود. با نوشتن **axis tight** به صورت خودکار محدوده‌ها طوری تغییر می‌کند که کل نمودار را در بر بگیرد. با نوشتن **help axis** حالت‌های دیگر را نیز می‌توان مشاهده کرد.

بسیاری از این تغییرات ذکر شده با استفاده از پنجره نمودار و به صورت گرافیکی نیز انجام می‌شود. ولی این دستورها برای نوشتن M-file کارایی فراوان دارد. برای بستن نمودار از دستور **close** یا **close all** استفاده می‌شود.

رسم نمودار با plot

دستور **plot** بر روی بردارهایی از داده‌های عددی عمل می‌کند. این دستور به صورت **plot(X, Y)** استفاده می‌شود که **X** و **Y** بردارهایی با طول یکسان می‌باشد. به عنوان مثال داریم.

```
>> X = [1 2 3]; Y = [4 6 5]; plot(X, Y)
```

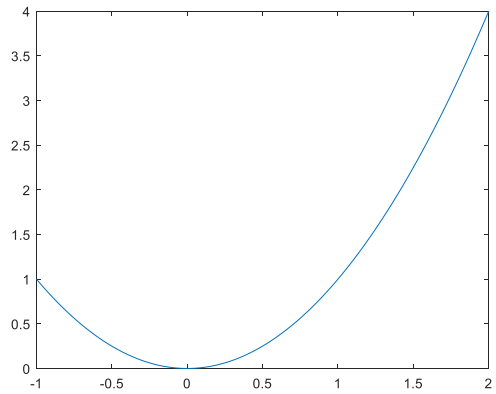


شکل ۷ رسم نمودار با استفاده از دستور plot

دستور **plot** بردارهای **X** و **Y** را به عنوان لیستی از مختصات x و y نقاط متوالی یک نمودار در نظر می‌گیرد و نقاط را با تکه‌های خط به یکدیگر وصل می‌نماید. بنابراین در شکل ۷ نرم‌افزار MATLAB نقطه‌ی (1,4) را به نقطه‌ی (2,6) و سپس به نقطه‌ی (3,5) وصل نموده است.

برای رسم نمودار x^2 در بازه‌ی -1 تا 2 ابتدا یک لیست **x** از مقادیر x تولید می‌شود و سپس عبارت **plot(X, X.^2)** وارد می‌شود. می‌بایست از تعداد کافی مقدار برای x استفاده شود تا نمودار حاصل ناشی از اتصال نقاط، نرم به نظر برسد. در این‌جا از گام‌های برابر با 0.01 استفاده می‌شود. با دستور زیر یک سهمی رسم می‌شود. حاصل آن در شکل ۸ رسم شده است.

```
>> X = -1:0.01:2; plot(X, X.^2)
```



شکل ۸ رسم نمودار با دستور plot

رسم هم‌زمان چند منحنی

هر زمان که دستور رسم یک نمودار اعمال می‌شود، MATLAB نمودار قبل را پاک کرده و یک نمودار جدید رسم می‌نماید. اگر بناست دو یا چند نمودار بر روی یکدیگر رسم شود، عبارت **hold on** را باید نوشت. این دستور باعث می‌شود نرم‌افزار MATLAB نمودارهای قبل را نگه داشته و نمودارهای جدید را بالای نمودار قبلی رسم نماید. این امر تا زمانی که از عبارت **hold off** استفاده نشود پابرجاست. در این‌جا با استفاده از دستور **ezplot** چند نمودار رسم می‌شود.

```
>> ezplot('exp(-x)', [0 10])
>> hold on
>> ezplot('sin(x)', [0 10])
>> hold off
>> title('exp(-x) and sin(x)')
```

نتیجه‌ی حاصل قبلا در شکل ۵ رسم شده است. دستورهای **hold on** و **hold off** با همه‌ی دستورهای گرافیکی کار می‌نماید. با استفاده از **plot** می‌توان چند منحنی را به صورت مستقیم رسم نمود. در زیر مثالی آمده است.

```
>> X = 0:0.01:10; plot(X, exp(-X), X, sin(X))
```

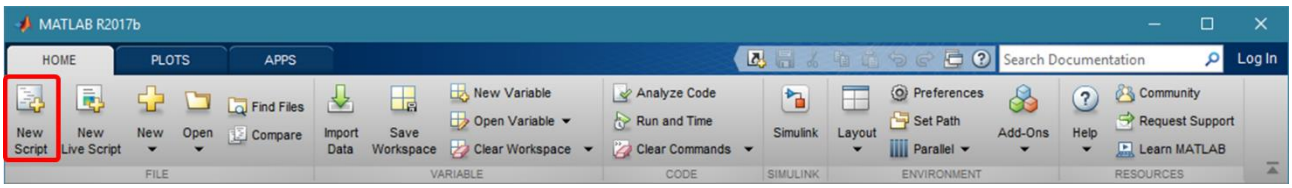
توجه داشته باشید که بردار مربوط به محور x برای هر تابعی که بناست رسم شود، باید از قبل تعریف شده باشد.

M-file نویسی

M-file این امکان را فراهم می‌آورد که چندین دستور MATLAB را در یک فایل ذخیره کرد و سپس آن‌ها را با یک دستور یا کلیک موشواره اجرا نمود. M-fileها در برنامه‌های طولانی که نیاز به سعی و خطاست کاربرد فراوان دارد و امکان در اختیار قرار دادن کدها را برای سایر کاربران فراهم می‌آورد. دو نوع مختلف M-file وجود دارد: M-file نگاره‌ای[۷] و M-file تابعی[۸]. در ادامه نحوه‌ی استفاده از هر دو نوع M-file بیان می‌شود.

M-fileها، فایل‌های متنی معمولی هستند که شامل دستورهای MATLAB است. این فایل‌ها را می‌توان با هر نوع ویرایشگر متن که قابلیت ذخیره‌ی فایل‌ها به صورت یک فایل متنی ASCII را داشته باشد، ایجاد و ویرایش کرد. برای راحتی بیشتر می‌توان از ویرایشگر/اشکال‌یاب خود MATLAB استفاده نمود. با نوشتن دستور **edit** به تنهایی می‌توان یک فایل جدید را ویرایش نمود و اگر به دنبال آن نام یک M-file موجود در پوشه‌ی فعلی و یا در مسیرهای تعریف شده برای نرم‌افزار بیاید، می‌توان آن را ویرایش نمود. یکی دیگر از راه‌ها برای ایجاد یک نگاره‌ی جدید MATLAB کلیک بر روی دکمه‌ی New Script در سربرگ Home نرم‌افزار است.

7 Script M-file
8 Function M-file



شکل 9 ایجاد یک نگاره‌ی جدید MATLAB

با دوبار کلیک بر روی M-fileهای موجود در پوشه‌ی فعلی نیز می‌توان آن‌ها را در ویرایشگر/اشکال‌یاب MATLAB باز نمود.

M-fileهای نگاره‌ای

M-fileهای نگاره‌ای شامل یک دنباله از دستورهای MATLAB است که دارای ترتیب مشخصی برای اجراست. یک فایل شامل خطوط زیر ایجاد می‌شود.

```
format long
x = [0.1, 0.01, 0.001];
y = sin(x)./x
```

این فایل با نام `task1.m` در پوشه‌ی فعلی ذخیره شده است. نام فایل را می‌توان به هر طریقی با قید محدودیت‌های نام‌گذاری سیستم عامل انتخاب کرد. باید توجه داشت که پسوند «.m» اجباری است.

برای اجرای این نگاره می‌توان در پنجره‌ی دستور عبارت **task1** را وارد نمود. در این‌جا نباید پسوند `.m` وارد شود و MATLAB به صورت خودکار این پسوند را برای جستجو در M-fileها اضافه می‌نماید. خروجی‌ها - و نه دستورها - در پنجره‌ی دستورها نمایش داده می‌شود. حال دنباله‌ی دستورها را می‌توان به راحتی و با تغییر M-file با نام `task1.m` ویرایش نمود. اگر بناست که sin(0.0001) /0.0001 نیز محاسبه شود، می‌توان M-file را به صورت زیر ویرایش نمود و نگاره‌ی ویرایش شده را با نوشتن **task1** دوباره اجرا نمود.

```
format long
x = [0.1, 0.01, 0.001, 0.0001];
y = sin(x)./x
```

نرم‌افزار MATLAB با هر بار اجرای M-file آخرین ویرایش‌ها را ذخیره می‌نماید.

اضافه کردن توضیحات

افزودن توضیحات در M-fileها بسیار مفید می‌باشد. این توضیحات بیان‌گر محاسبات صورت گرفته و یا تفسیر نتایج محاسبات می‌باشد. در نرم‌افزار MATLAB علامت درصد (%) یک توضیح را آغاز می‌نماید. MATLAB ادامه‌ی این خط را اجرا نمی‌نماید. ویرایش‌گر MATLAB رنگ نوشته‌ی توضیح را سبز رنگ نشان می‌دهد تا از دستورهای MATLAB که با مشکی نشان داده می‌شود، متمایز شود. در زیر نسخه‌ی جدید `task1.m` به همراه پاره‌ای توضیح آمده است.

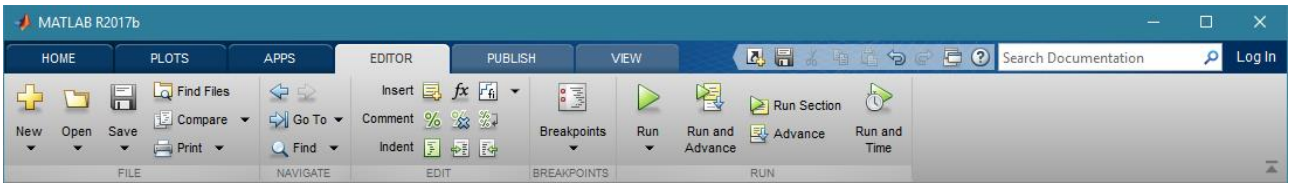
```
format long % turn on 15 digit display
x = [0.1, 0.01, 0.001];
y = sin(x)./x
% These values illustrate the fact that the limit of
% sin(x)/x as x approaches 0 is 1.
```

توجه شود که توضیحات چند خطی نیازمند قرار دادن یک علامت درصد در شروع هر خط می‌باشد. علاوه بر این می‌توان برای توضیحات چند خطی از عملگر **%{** و **%}** به صورت زیر استفاده نمود.

```
format long % turn on 15 digit display
x = [0.1, 0.01, 0.001];
y = sin(x)./x
%{
These values illustrate the fact that the limit of
sin(x)/x as x approaches 0 is 1.
%}
```

حالت سلول‌بندی

می‌توان یک M-file را به زیرواحدهایی با نام سلول تقسیم نمود. این قابلیت برای حالتی که با M-fileهای طولانی کار می‌شود مفید است. برای شروع یک سلول، یک خط جدید با دو علامت درصد %% و یک فاصله ایجاد می‌شود. ادامه‌ی این خط عنوان یک سلول تلقی می‌شود. می‌توان هر سلول با فشردن دکمه‌ی Ctrl+Enter و یا راست کلیک بر روی سلول و انتخاب گزینه‌ی Evaluate Current Section اجرا نمود.این امر کمک زیادی برای بررسی نتیجه‌ی تغییرات در یک سلول بدون اجرای کل فایل می‌نماید. این کارها از طریق منوی نواری و سربرگ Editor قابل انجام است.



شکل ۱۰ سربرگ Editor منوی نواری

راه‌اندازی اولیه‌ی M-file نگاره‌ای

به منظور این‌که نتایج M-file نگاره‌ای قابل بازتولید باشد، این نگاره می‌بایست همه‌ی متغیرهای مورد نیاز خود را داشته باشد و تحت تأثیر متغیرهای دیگر که ممکن است در جای دیگری از MATLAB تعریف شده و نمودارهایی که قبلاً رسم شده، قرار نگیرد. به عنوان مثال اگر در پنجره‌ی دستور متغیری با نام **sin** تعریف شود و سپس نگاره‌ی `task1.m` اجرا شود، پیام خطایی ظاهر می‌شود. علت این امر آن است که اکنون **sin** به جای این‌که بیان‌گر تابع داخلی MATLAB باشد، بیان‌گر نام یک متغیر است. با در نظر داشتن این موضوع، می‌توان با اضافه کردن خط **clear all** در شروع نگاره از تعریف نشدن متغیرهایی که نتایج را تحت تأثیر قرار می‌دهد، اطمینان حاصل نمود. هم‌چنین با نوشتن **close all** در ابتدای M-file نگاره‌ای که یک نمودار تولید می‌نماید، همه‌ی پنجره‌های نمودارها را می‌بندد تا خللی در کار نمودارهایی که بناست رسم شود، ایجاد نمی‌نماید.

M-fileهای تابعی

بر خلاف M-fileهای نگاره‌ای، با استفاده از M-fileهای تابعی می‌توان زمانی که درون یک خط دستور MATLAB و یا از یک M-file دیگر این تابع‌ها فرا خوانده می‌شود، مقادیری را به عنوان ورودی مشخص کرد. مشابه M-file نگاره‌ای یک M-file تابعی یک فایل متنی ساده است که می‌بایست درون پوشه‌ی فعلی و یا در مسیرهای تعریف شده‌ی MATLAB قرار داشته باشد.

بناست مسأله‌ی قبل دوباره پیاده‌سازی شود که در آن مقدار $\sin x / x$ برای $x = 10^{-b}$ برای چندین مقدار b محاسبه می‌شد. برای این کار یک M-file تابعی با نام `sinelimit.m` طراحی شده است.

```
function y = sinelimit(c)
% SINELIMIT computes sin(x)/x for x = 10^(-b),
% where b = 1, ..., c.
format long
b = 1:c;
x = 10.^(-b);
y = (sin(x)./x)';
```

خط اول این تابع با **function** آغاز می‌شود و مشخص می‌کند که این فایل یک M-file تابعی است. ویرایشگر MATLAB این کلمه‌ی خاص را آبی می‌نماید. خط اول این M-file نام تابع را مشخص می‌کند و در آن آرگومان‌ها یا پارامترهای ورودی و خروجی مشخص می‌شود. در این مثال نام تابع **sinelimit** است. نام فایل (بدون پسوند .m) می‌بایست با نام تابع یکی باشد. زمانی که یک M-file تابعی جدید در یک پنجره‌ی ویرایش بدون عنوان ایجاد می‌شود، با انتخاب save، ویرایشگر می‌داند که می‌بایست نام فایل را `sinelimit.m` بگذارد. تابع در این مثال یک ورودی با نام **c** و یک خروجی با نام **y** دارد. وقتی تابع فراخوانی می‌شود، این خروجی ایجاد می‌شود.

می‌بایست سعی شود که خطوط اول یک M-file تابعی چند خط توضیح باشد که نشان‌دهنده‌ی عملکرد M-file است. با انجام این کار، دستور help به صورت خودکار این اطلاعات را برمی‌گرداند.

```
>> help signlimit
--- help for sinelimit ---

  sinelimit computes sin(x)/x for x = 10^(-b),
```

```
  where b = 1, ..., c.
```

بقیه‌ی خطوط M-file بیان‌گر تابع است. در این مثال b یک بردار سطری است که اعداد صحیح از 1 تا c را در بردارد. سپس بردار x از روی بردار b به دست آمده و در نهایت y از روی x محاسبه می‌شود. توجه شود که متغیرهای تعریف شده درون یک M-file تابعی متغیرهای محلی هستند و هیچ ارتباطی با متغیرهای با نام مشابه در پنجره‌ی دستور MATLAB ندارد و MATLAB مقادیر آن‌ها را بعد از اجرای M-file تابعی در حافظه ذخیره نمی‌کند.

در ادامه مثالی از استفاده از تابع **sinelimit** آمده است.

```
>> sinelimit(5)
ans =
   0.998334166468282
   0.999983333416666
   0.999999833333342
   0.999999998333333
   0.999999999983333
```

برنامه‌نویسی MATLAB

هر زمان که یک M-file ایجاد می‌شود یک برنامه‌ی رایانه‌ای با زبان برنامه‌نویسی MATLAB نوشته می‌شود. در این‌جا به معرفی دستورهای برنامه‌نویسی و روش‌هایی مفید برای حل مسایل پیچیده در MATLAB پرداخته می‌شود. بسیاری از دستورهای MATLAB خودشان یک M-file هستند که می‌توان آن‌ها را با نوشتن دستور **type** یا **edit** مورد بررسی قرار داد. به عنوان مثال با وارد کردن **type isprime** می‌توان M-file دستور **isprime** را مشاهده نمود.

حلقه‌ها

با استفاده از حلقه می‌توان یک دستور و یا یک گروه از دستورها را چندین بار تکرار نمود. راحت‌ترین راه برای ایجاد یک حلقه استفاده از عبارت **for** است. در زیر یک مثال ساده از محاسبه و نمایش حاصل $10! = 10 \times 9 \times \ldots \times 2 \times 1$ نشان داده شده است.

```
f = 1;
for n = 2:10
    f = f*n;
end
f
```

یک حلقه با عبارت **for** آغاز و با عبارت **end** پایان می‌یابد. دستور بین این دو عبارت در مجموع برای هر مقدار **n** از 2 تا 10، نه بار تکرار می‌شود. در این‌جا از نقطه‌درنگ‌نما برای عدم نمایش نتایج میانی خروجی حلقه استفاده شده است. برای دیدن حاصل نهایی می‌بایست **f** را در انتهای حلقه نوشت. بدون علامت نقطه‌درنگ‌نما نرم‌افزار MATLAB همه‌ی مقادیر میانی !2، !3 و غیره را نمایش می‌دهد.

ویرایشگر/اشکال‌یاب به صورت خودکار دستورهای **for** و **end** را به رنگ آبی در می‌آورد. به منظور قابلیت خواندن بیشتر، سعی می‌شود نوشته درون حلقه دارای فرورفتگی باشد. البته این فرورفتگی لازم نیست.

شاخه‌بندی

در بسیاری از تابع‌های شخصی می‌توان از یک M-file تابعی برای اجرای مجموعه‌ای از دستورها برای هر ورودی اجرا نمود. با این وجود ممکن است بنا باشد مجموعه دستورهای متفاوتی برای حالت‌های مختلفی وابسته به ورودی اجرا شود. این کار با استفاده از دستورهای شاخه‌بندی انجام می‌شود. مانند بسیاری از زبان‌های برنامه‌نویسی دیگر، این شاخه‌بندی در MATLAB با استفاده از دستور **if** انجام می‌شود. در ادامه دستور شاخه‌بندی اصلی دیگر با نام **switch** نیز معرفی می‌شود.

شاخه‌بندی با if

به منظور نمایش شاخه‌بندی با **if** می‌توان M-file تابعی زیر را در نظر گرفت که قدر مطلق یک عدد حقیقی را محاسبه می‌نماید.

```
function y = absval(x)
if x >= 0
    y = x;
```

```
else
    y = -x;
end
```

خط اول این M-file بیان می‌کند که تابع دارای یک ورودی **x** و یک خروجی **y** است. اگر ورودی **x** یک عدد غیرمنفی باشد، MATLAB عبارت روبروی **if** را درست تلقی می‌کند و سپس دستور بین **if** و **else** اجرا می‌شود و مقدار **y** برابر مقدار **x** قرار می‌گیرد. اگر **x** منفی باشد، MATLAB تا عبارت **else** را نادیده می‌گیرد و دستور بعد از آن را اجرا می‌کند و مقدار **y** را برابر با **-x** قرار می‌دهد. فرورفتگی دستورها اختیاری است و برای خواننده‌ی دستورها مفید است و MATLAB این کار را به صورت خودکار انجام می‌دهد.

به طور کلی در ادامه‌ی **if** می‌بایست عبارتی باشد که MATLAB آن را صحیح یا غلط ارزیابی نماید. بعد از تعدادی دستور می‌بایست عبارت **end** بیاید. در این بین می‌تواند یک یا چند عبارت **elseif** و یک عبارت **else** قرار بگیرد. اگر نتیجه ارزیابی MATLAB صحیح بود، عبارت‌های بین **if** و هر کدام از **elseif**، **else** یا **if** که زودتر دیده شود، اجرا می‌شود. سپس همه‌ی دستورهای تا **end** نادیده گرفته می‌شود. اگر نتیجه‌ی ارزیابی غلط باشد MATLAB به سراغ اولین **elseif**، **else** یا **end** می‌رود و اجرای برنامه از آن‌جا ادامه می‌یابد. در صورت مشاهده‌ی یک **elseif** یک ارزیابی مجدد صورت می‌پذیرد. در زیر تابع **absval** طوری تغییر می‌کند که نیاز به عبارت **else** نباشد و به عبارتی هیچ دستوری در صورتی که نتیجه‌ی ارزیابی غلط باشد، اجرا نشود.

```
function y = absval(x)
y = x;
if y < 0
    y = -y;
end
```

عبارت **elseif** برای مواقعی که بیش از دو حالت مختلف برای ارزیابی صحیح و خطا وجود دارد، مفید می‌باشد. این دستور معادل استفاده از **else** و در ادامه‌ی آن عبارت **if** می‌باشد. در مثال زیر برای تابع `signum` **elseif** استفاده می‌شود که تابع علامت را محاسبه می‌نماید. مانند **absval** نرم‌افزار MATLAB برای تابع علامت نیز یک تابع داخلی دارد که کلی‌تر از حالتی است که در زیر آمده است.

```
function y = signum(x)
if x > 0
    y = 1;
elseif x == 0
    y = 0;
else
    y = -1;
end
```

در این‌جا اگر **x** مثبت باشد، خروجی **y** برابر 1 می‌شود و همه‌ی دستورهای بعد از عبارت **elseif** تا عبارت **end** نادیده گرفته می‌شود. به طور مشخص ارزیابی عبارت **elseif** انجام نمی‌شود. اگر **x** مثبت نباشد، نرم‌افزار MATLAB به عبارت **elseif** می‌رود و بررسی می‌کند که آیا **x** برابر **0** است یا نه. اگر این طور بود **y** برابر **0** می‌شود و در غیر این‌صورت **y** برابر **-1** می‌شود. توجه داشته باشید که در MATLAB لازم است از دو علامت مساوی برای ارزیابی مساوی استفاده شود. یک علامت مساوی برای تخصیص یک مقدار به یک متغیر رزور شده است.

عبارت‌های منطقی

در مثال‌های قبل از عملگرهای رابطه‌ای نظیر **<=**، **>** و **==** برای ایجاد یک عبارت منطقی مورد استفاده قرار گرفت و به MATLAB دستور می‌دهد تا بین دستورهای مختلف متناسب با اینکه عبارت درست یا غلط است. با نوشتن **help relop** می‌توان همه‌ی عملگرهای رابطه‌ای نرم‌افزار را مشاهده نمود. بعضی از این عملگرها مانند **&** (وَ منطقی) و **|** (یا منطقی) را می‌توان برای شکل دادن به عبارت‌های منطقی پیچیده‌تر استفاده می‌شود. به عنوان مثال عبارت **(x > 0) | (y > 0)** هرگاه **x** یا **y** (یا هردو) مثبت باشد، صحیح خواهد بود و اگر هیچ‌کدام مثبت نباشد، غلط خواهد بود. در این مثال خاص، نیازی به پرانتز نیست ولی به طور کلی عبارت‌های منطقی پیچیده مانند همین مثال راحت‌تر خوانده می‌شود و اگر از پرانتز برای پرهیز از ابهام استفاده شود،احتمال رخ دادن اشتباه کم‌تر خواهد بود.

در بحث شاخه‌بندی عبارت‌هایی که به عنوان درست یا غلط تعبیر می‌شوند، مورد توجه قرار گرفت. در حالی که این عبارت‌های در بیشتر موارد کافی به نظر می‌رسد، می‌توان از هر عبارتی در MATLAB که می‌تواند به صورت عددی محاسبه شود، بعد از **if** یا

elseif استفاده نمود. در واقع نرم‌افزار MATLAB هیچ تمایزی بین عبارت‌های منطقی و عبارت‌های عددی معمولی نمی‌گذارد. به مثال زیر توجه کنید که در آن یک عبارت منطقی به تنهایی در پنجره‌ی دستور نوشته می‌شود.

```
>> 2 > 3
ans =
     0
```

وقتی یک عبارت منطقی محاسبه می‌شود، MATLAB مقدار **0** (برای غلط) و مقدار **1** (برای درست) را به آن اختصاص می‌دهد. بنابراین اگر عبارت **3 > 2** وارد شود پاسخ **1** خواهد بود. در MATLAB با عملگرهای رابطه‌ای تا جایی که خروجی عددی داشته باشند، مانند عملگرهای ریاضی برخورد می‌شود. در زیر مثالی آمده است.

```
>> 2 | 3
ans =
     1
```

در عملگر «یا» **|** اگر هر دو طرف برابر با صفر باشند، حاصل برابر با **0** می‌شود و در غیر این صورت خروجی آن **1** می‌شود. در حالی که خروجی عملگرهای رابطه‌ای همواره **0** یا **1** هستند، اما ورودی‌های غیر صفر برای عملگرهایی نظیر **&** (وَ منطقی[9])، **|** (یا منطقی[10]) و **~** (نقیض[11]) در MATLAB درست تلقی می‌شوند و تنها **0** به عنوان غلط در نظر گرفته می‌شود.

اگر ورودی‌های یک عملگر رابطه‌ای به جای اسکالر، بردار یا ماتریس باشند، مانند عملگرهایی نظیر **+** و **.*** ، عملیات به صورت درایه‌به‌درایه صورت می‌پذیرد و خروجی به صورت یک آرایه از صفر و یک خواهد بود. در ادامه مثال‌هایی آمده است.

```
>> [2 3] < [3 2]
ans =
  1×2 logical array
   1   0
>> x = -2:2; x >= 0
ans =
  1×5 logical array
   0   0   1   1   1
```

در مورد دوم تک تک درایه‌های **x** با اسکالر **0** مقایسه می‌شود.

خروجی عملگر رابطه‌ای یک آرایه‌ی منطقی است و از آن می‌توان برای انتخاب درایه‌های یک آرایه که در یک شرط خاص صدق می‌کنند استفاده نمود. به عنوان مثال عبارت **x(x >= 0)** منجر به یک بردار می‌شود که تنها شامل المان‌های غیرصفر **x** (به طور دقیق‌تر المان‌هایی با بخش حقیقی غیر صفر) می‌باشند. بنابراین اگر **x = -2:2** باشد، خروجی به صورت زیر خواهد شد.

```
>> x(x >= 0)
ans =
     0   1
```

اگر از یک آرایه‌ی منطقی برای انتخاب المان‌های یک آرایه‌ی دیگر استفاده شود، دو آرایه می‌بایست اندازه‌ی یکسانی داشته باشند. المان‌های متناظر با هر **1** آرایه‌ی منطقی انتخاب شده در حالی که المان‌های متناظر با هر **0** انتخاب نمی‌شود. در مثال بالا، خروجی مشابه حالتی است که عبارت **x(3:5)** نوشته شود. اما در این‌جا **3:5** یک آرایه‌ی عددی معمولی است که اندیس عددی المان‌ها را انتخاب می‌نماید.

در ادامه توضیحاتی در مورد نحوه‌ی تصمیم‌گیری **if** و **elseif** برای درست یا غلط بودن یک عبارت آمده است. برای عبارتی که یک مقدار اسکالر حقیقی است، معیار به این صورت است که اعداد غیرصفر معادل درست و عدد **0** معادل غلط است. با این وجود برای اعداد مختلط، تنها بخش حقیقی در نظر گرفته می‌شود. بنابراین در عبارت **if** یا **elseif** هر عددی با بخش حقیقی غیرصفر درست تلقی شده و هر عدد با بخش حقیقی صفر، غلط تلقی می‌شود. علاوه بر این اگر عبارتی منجر به یک ماتریس یا بردار شود،شرط **if** یا **elseif** حتما باید هم‌چنان یک تصمیم درست یا غلط باشد. قرارداد MATLAB بدین صورت است که همه‌ی المان‌ها باید درست باشد. به عبارتی همه‌ی المان‌ها باید دارای بخش حقیقی غیرصفر باشند تا آن عبارت به عنوان درست تلقی شود.

---

9 AND
10 OR
11 NOT

می‌توان شیوه‌ی شاخه‌بندی با ورودی بردار را با نقیض کردن ارزیابی‌ها با ~ و استفاده از دستورهای **any** و **all** دست‌کاری نمود. به عنوان مثال عبارت‌های **if x== 0; … ; end** یک بلوک کد (بیان شده با **. . .**) را وقتی همه‌ی المان‌های **x** برابر صفر باشد اجرا می‌نماید. اگر بنا باشد یک دسته از دستورها زمانی که تمام المان‌های **x** صفر نباشد، اجرا شود از صورت **if x ~= 0; else; … ;** **end** استفاده می‌شود. در این‌جا **=~** یک عملگر رابطه‌ای برای «برابر نیست» می‌باشد. بنابراین زمانی که یکی از المان‌های **x** برابر صفر باشد، ارزیابی معادل غلط است و دستورها تا عبارت **else** در اجرا نادیده گرفته می‌شود. می‌توان با استفاده از دستور **any** به صورت سرراست‌تری به نتیجه‌ی قبل رسید. این دستور زمانی خروجی صحیح دارد که المان‌های آرایه غیر صفر باشند. مثالی از این دستور به صورت **if any(x == 0); . . . ; end** می‌باشد. به صورت مشابه اگر همه‌ی درایه‌های آرایه غیرصفر باشند خروجی **all** صحیح است.

در ادامه تعدادی مثال برای نمایش بعضی از ویژگی‌های عبارت‌های منطقی و شاخه‌بندی که تاکنون توضیح داده شد، آمده است.

اگر بنا باشد M-file‌ی برای محاسبه‌ی تابع $f(x) = \begin{cases} \sin(x)/x & x \neq 0 \\ 1 & x = 0 \end{cases}$ نوشته شود به صورت زیر عمل می‌شود.

```
function y = f(x)
if x == 0
    y = 1;
else
    y = sin(x)/x;
end
```

تابع فوق تنها برای ورودی **x** اسکالر به خوبی عمل می‌کند و نمی‌تواند بر روی بردار یا ماتریس‌ها اعمال شود. می‌توان با تبدیل **/** به **./** در تعریف دوم **y** و تغییر تعریف اول برای تبدیل **y** به اندازه‌ی **x** می‌توان تغییری در تابع داد. اما اگر **x** هم دارای المان‌های صفر و غیرصفر باشد، آن‌گاه MATLAB حاصل شرط **if** را معادل غلط در نظر گرفته و از تعریف دوم استفاده می‌کند. آن‌گاه بعضی از درایه‌های آرایه‌ی خروجی **y** مقدار **NaN** (not a number) خواهد بود. علت این امر رخ دادن **0/0** است که مبهم است.

یکی از راه‌هایی که می‌توان این M-file را بر روی بردارها و ماتریس‌ها اعمال نمود، استفاده از یک حلقه برای محاسبه تابع به صورت درایه به درایه است که درون حلقه یک عبارت شرطی **if** است.

```
function y = f(x)
y = ones(size(x));
for n = 1:prod(size(x))
    if x(n) ~= 0
        y(n) = sin(x(n))/x(n);
    end
end
```

در M-file فوق ابتدا خروجی **y** را به صورت یک آرایه از یک‌ها با اندازه‌ای برابر با ورودی **x** ایجاد می‌شود. در این‌جا از دستور **size(x)** برای تعیین تعداد سطر و ستون‌های **x** استفاده می‌شود. عبارت **prod(size(x))** تعداد المان‌های **x** را مشخص می‌کند. در عبارت **for** مقدار **n** از **1** تا این تعداد المان‌های **x** تغییر می‌کند. غیرصفر بودن هر المان **x(n)** بررسی شده و سپس درایه‌ی متناظر **y(n)** محاسبه می‌شود. اگر **x(n)** برابر 0 باشد، از آن‌جا که مقدار اولیه‌ی **y(n)** برابر با 1 است، هیچ نیازی به تغییر آن نیست.

نکته: در این‌جا یک قابلیت مهم و ظریف MATLAB استفاده شد. هر المان یک ماتریس را می‌توان با یک اندیس مشخص نمود. به عنوان مثال اگر **x** یک ماتریس ۳ در ۲ باشد، المان‌های آن به صورت **x(1)**، **x(2)**، …، **x(6)** می‌باشد. با این روش از حلقه‌های تو در تو اجتناب شده است. به صورت مشابه می‌توان از عبارت **length(x(:))** به جای **prod(size(x))** برای شمردن کل تعداد المان‌های **x** استفاده نمود. با این وجود می‌بایست دقت نظر داشت که اگر از قبل **y** به اندازه‌ی **x** تعریف نشده باشد و درون حلقه از عبارت **else** برای قرار دادن مقدار **y(n)** برابر یا **1** در زمانی که **x(n)** برابر 0 است، استفاده شود، آن‌گاه **y** به جای یک ماتریس ۳ در ۲ یک آرایه‌ی ۱ در ۶ خواهد بود. می‌توان در انتهای M-file از عبارت **y = reshape(y, size(x))** برای تغییر اندازه‌ی **y** به اندازه‌ی **x** استفاده نمود. با این وجود حتی اگر اندازه‌ی آرایه‌ی خروجی مهم نباشد، به طور کلی بهترین رویکرد قبل از محاسبه‌ی درایه‌به‌درایه درون حلقه، تعریف اولیه یک آرایه با مقدار مناسب است. در این حالت حلقه سریع‌تر اجرا می‌شود.

در ادامه بر روی M-file قبل تغییری ایجاد شده است.

```
Function y = f(x)
if x ~= 0
    y = sin(x)./x;
    return
end
```

```
y = ones(size(x));
for n = 1:prod(size(x))
    if x(n) ~= 0
        y(n) = sin(x(n))/x(n);
    end
end
```

در بالای حلقه یک بلوک چهار خطی به منظور اجرای سریع‌تر کد در هنگامی که همه‌ی درایه‌های **x** غیرصفر باشند، اضافه شده است. از آن‌جا که MATLAB بردارها را بسیار بهینه‌تر از حلقه‌ها محاسبه می‌کند، M-file جدید اگر دارای تعداد زیادی المان و همگی غیرصفر باشند، به مراتب سریع‌تر عمل می‌نماید. در این بلوک جدید اضافه شده اولین عبارت **if** زمانی درست خواهد بود که همه‌ی المان‌های **x** غیرصفر باشند. در این حالت خروجی **y** با استفاده از عملگرهای برداری MATLAB محاسبه می‌شود که به طور کلی بهینه‌تر از اجرای حلقه می‌باشد. سپس از دستور **return** برای متوقف کردن M-file بدون اجرای دستوری اضافی استفاده می‌شود. اگر **x** دارای تعدادی المانی صفر داشته باشد، آن‌گاه شرط **if** غلط است و M-file دستورها را تا بعد از عبارت **end** نادیده می‌گیرد.

معمولا می‌توان با استفاده از آرایه‌های منطقی از حلقه‌ها و دستورهای شاخه‌بندی اجتناب کرد. در ادامه M-filedای وجود دارد که کار مشابه مثال‌های قبل را انجام می‌دهد. مزیت این شیوه‌ی آخر خلاصه بودن و بهینه‌تر بودن نسبت به M-fileهای قبل است. علت این امر آن است که در هیچ حالتی از حلقه استفاده نمی‌شود.

```
function y = f(x)
y = ones(size(x));
n = (x ~= 0);
y(n) = sin(x(n))./x(n);
```

در این‌جا **n** یک آرایه‌ی منطقی با اندازه‌ی برابر با **x** است که درجاهایی که **x** دارای المان غیرصفر است برابر با **1** و درجاهای دیگر صفر است. بنابراین خطی که **y(n)** را تعریف می‌کند، تنها المان‌های **y** متناظر با مقادیر غیرصفر **x** را تغییر می‌دهد و بقیه‌ی المان‌ها را برابر با **1** باقی می‌گذارد.

شاخه‌بندی با دستور switch

یکی دیگر از دستورهای اصلی شاخه‌بندی **switch** است. با این دستور زمانی که شرط‌ها به صورت مساوی است، شاخه‌بندی بین چندین حالت به راحتی دو حالت انجام می‌شود. در ادامه یک مثال ساده آمده که بین سه حالت ورودی تمایز قائل می‌شود.

```
function y = count(x)
switch x
case 1
    y = 'one';
case 2
    y = 'two';
otherwise
    y = 'many';
end
```

در این‌جا عبارت **switch** ورودی **x** را محاسبه کرده و سپس اجرای M-file به عبارت **case** که دارای مقدار یکسانی است، می‌رود. بنابراین اگر ورودی **x** برابر با **1** باشد، سپس خروجی **y** رشته‌ی **'one'** می‌شود. اگر **x** برابر **2** باشد، آن‌گاه **y** برابر **'two'** می‌شود. در هر حالت زمانی که MATLAB به یک عبارت **case** و یا عبارت **otherwise** می‌رسد، برنامه به عبارت **end** می‌رود و حداکثر یک حالت اجرا می‌شود. اگر هیچ انطباقی بین عبارت‌های **case** وجود نداشت، نرم‌افزار MATLAB به سراغ عبارت اختیاری **otherwise** و یا عبارت **end** می‌رود. در مثال فوق عبارت **otherwise** وجود دارد، بنابراین اگر ورودی برابر با **1** یا **2** نباشد، خروجی برابر **'many'** خواهد بود.

برخلاف **if**، دستور **switch** اجازه‌ی عبارت‌های برداری را نمی‌دهد، ولی اجازه‌ی استفاده از رشته را می‌دهد. از این قابلیت می‌توان برای طراحی یک M-file تابعی استفاده می‌شود که از یک رشته به عنوان آرگومان ورودی استفاده می‌کند و با استفاده از آن حالت‌های مختلف برنامه‌ی نوشته شده را انتخاب می‌کند.

بسمه تعالی

پیش‌نیاز اول

آشنایی با نرم‌افزار MATLAB

اهداف آزمایش

یکی از ابزارهای شبیه‌سازی و پیاده‌سازی سامانه‌ها و الگوریتم‌های مخابراتی نرم‌افزار MATLAB می‌باشد. در این آزمایشگاه به منظور انجام شبیه‌سازی‌ها، این نرم‌افزار مورد استفاده قرار می‌گیرد. در این آزمایش بناست تا آشنایی کافی با این نرم‌افزار به دست آید تا فرآیند آموزشی نرم‌افزار در طول آزمایش‌های آتی به حداقل رسیده و انجام آزمایش‌ها تسریع شود. عمده‌ی مطالبی که در اکثر کتاب‌های آموزشی MATLAB وجود دارد، چندان مورد استفاده قرار نمی‌گیرد و تنها فرآیند آموزش را ملال‌آور و کند می‌نماید. از این رو در این آزمایش تأکید بر آموزش مطالب و موارد پرکاربرد و مفید می‌باشد.

در انتهای این آزمایش دانشجو می‌بایست:

- با محیط نرم‌افزار MATLAB آشنایی پیدا نماید.
- انواع محاسبات و عملیات‌های ریاضی و محاسبات سمبلیک را به راحتی انجام دهد.
- از بردارها، ماتریس‌ها و عملگرهای مرتبط با آن‌ها استفاده نماید.
- متغیرها و توابع داخلی نرم‌افزار MATLAB را بشناسد و از آن‌ها استفاده نماید.
- تسلط کافی در نوشتن M-File و Function داشته باشد.
- اصول برنامه‌نویسی حاکم بر نرم‌افزار را بیاموزد.
- از ابزارهای گرافیکی موجود بهره برده و انواع نمودارها را رسم نماید.

ابزارهای مورد نیاز

- رایانه
- نرم‌افزار MATLAB R2020b

شرح آزمایش

آزمایش پیش‌نیاز ۱-۱: محاسبات جبری و گرافیک

۱. مقادیر زیر را محاسبه نمایید.

أ. کسرهای 2709/1024، 10583/4000 و 2024/765. کدام یک از این کسرها تقریب بهتری برای $\sqrt{7}$ است. (**راهنمایی:** کسرها را درون یک بردار قرار دهید. کسر دقیق‌تر را یک‌بار با مقایسه ارقام کسرها با ارقام $\sqrt{7}$ و یک‌بار با استفاده از دستور **min** جهت یافتن اندیس درایه‌ای از بردار، با کمینه‌ی اختلاف با $\sqrt{7}$ به دست آوردید.)

ب. عدد 3^{301} را یک بار به صورت یک عدد اعشاری تقریبی با ۱۵ رقم اعشار (نمایش به صورت نماد علمی) و بار دیگر به صورت یک عدد صحیح دقیق محاسبه نمایید. (**راهنمایی:** از دستور **format** و دستور **vpa** با ورودی سمبلیک استفاده نمایید.)

ت. $(1/3) \times 20 - 20/3$. جواب جبری برابر با 0 است. اگر جواب 0 نشد، علت را بیان کنید.

ث. $10^{16} + 1 - 10^{16}$. به صورت جبری حاصل برابر با 1 است. اگر جواب 1 نشد، علت را بیان کنید.

۲. مقادیر زیر را تا ۱۵ رقم اعشار محاسبه نمایید. (**راهنمایی:** از دستور **vpa** استفاده ننمایید.)

أ. $\cosh(0.1)$

ب. $\ln(2)$

ت. $\arctan(1/2)$

۳. با استفاده از دستور **plot**، **fplot** و یا **ezplot** متناسب با نیاز، تابع‌های زیر را رسم نمایید.

أ. $y = \sin(1/x^2)$ برای $2 \le x \le -2$ و $2 \le y \le -2$. سعی کنید از هر سه دستور **ezplot**، **fplot** و **plot** استفاده نمایید. آیا هر سه جواب درست است؟ اگر از دستور **plot** استفاده می‌نمایید، یک بار گام‌ها را برابر با 0.1 و یک‌بار گام‌ها را برابر 0.001 در نظر بگیرید. (**راهنمایی:** ابتدا نمودار را رسم کرده و سپس از دستور **axis** استفاده نمایید.)

ب. منحنی پروانه را با استفاده از معادله‌های پارامتری زیر رسم نمایید.

$$x = \sin(t)\left[e^{\cos(t)} - 2\cos(4t) + \sin^5\left(\frac{t}{12}\right)\right]$$
$$y = \cos(t)\left[e^{\cos(t)} - 2\cos(4t) + \sin^5\left(\frac{t}{12}\right)\right]$$

این نمودار را برای $0 \le t \le 2\pi$ و $0 \le t \le 10\pi$ و با گام‌های $\pi/100$ رسم نمایید.

آزمایش پیش‌نیاز ۲-۱: ریاضیات و برنامه‌نویسی

۱. حاصل عبارت‌های زیر را به دست آورید و منطق جواب‌ها را توضیح دهید.

أ. $2 \times (3 < 8/4 + 2)^2 \quad < (-2)^3$

ب. $(5 + \sim 0)/3 == 3 - \sim(10/5 - 2)$

ت. $\sim 4 < 5 | 0 >= 12/6$

ث. $-7 < -5 < -2 \& 2 + 3 <= 15/3$

۲. برای عدد صحیح مثبت n، ماتریس $K(n)$ یک ماتریس $n \times n$ پایین‌مثلثی است که n سطر مثلث خیام را نمایش می‌دهد. برنامه‌ای بنویسید که ۷ سطر مثلث خیام را ایجاد نمایید.

$$K(5) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

۳. برای یک عدد صحیح مثبت n، ماتریس $A(n)$ یک ماتریس $n \times n$ است که درایه‌های آن در مکان (i,j) به صورت $a_{ij} = 1/(i+j-1)$ است. برای مثال

$$A(3) = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

مقادیر ویژه‌ی $A(n)$ همه اعدادی حقیقی است. یک M-file بنویسد که بزرگترین مقدار ویژه‌ی $A(500)$ را بدون هیچ خروجی اضافی چاپ نماید. (**راهنمایی:** اگر از دو حلقه‌ی تو در تو استفاده نمایید، M-file مدت زمانی طول می‌کشد که اجرا شود. سعی کنید از این کار برحذر باشید! این کار را بدون استفاده از حلقه می‌توان انجام داد. برای به دست آوردن مقادیر ویژه از دستور **eig** استفاده نمایید.)

۴. برنامه‌ای بنویسید که

أ. یک بردار با ۲۰ عدد صحیح تصادفی بین ۱۰ و ۳۰ تولید نماید.

ب. همه‌ی درایه‌های فرد را با اعداد تصادفی دیگری بین ۱۰ و ۳۰ جایگزین نماید. (بدون استفاده از حلقه)

ت. بخش (ب) را تا جایی که همه‌ی درایه‌ها اعدادی زوج شود، ادامه دهید.

ث. برنامه می‌بایست تعداد دفعات تکرار بخش (ب) برای زوج شدن همه‌ی درایه‌های بردار را شمارش کند. وقتی این اتفاق رخ داد، برنامه می‌بایست بردار و متنی را نمایش دهد که بیان می‌کند که چندبار تکرار برای تولید این بردار نیاز بوده است. محتوای متن می‌بایست شبیه عبارت زیر باشد.

The Vector Generated After 10 Iteration(s).

(**کلید واژه‌های راهنما: sprintf، while، randi، mod**)

۵. MATLAB تابعی به نام **lcm** دارد که کوچک‌ترین مضرب مشترک دو عدد را محاسبه می‌نماید. M-file تابع **mylcm.m** را بنویسید که کوچک‌ترین مضرب مشترک هر تعداد عدد مثبت صحیح را پیدا نماید. این اعداد می‌بایست بتوانند به صورت آرگومان‌های جداگانه و یا در قالب یک بردار وارد شوند. به عنوان مثال **mylcm(4, 5, 6)** و **mylcm([4 5 6])** می‌بایست عدد 60 را به عنوان جواب برگردانند. برنامه می‌بایست اگر یکی از ورودی‌ها مثبت نبود، یک پیغام خطای مفید تولید نماید. (**راهنمایی:** برای سه عدد می‌توان دستور **lcm** را برای به دست آوردن کوچک‌ترین مضرب مشترک دو عدد اول استفاده کرد و سپس از دستور **lcm** برای به دست آوردن کوچک‌ترین مضرب مشترک حاصل به دست آمده و عدد سوم استفاده نمود. M-file شما می‌بایست این رویکرد را تعمیم دهد.)

مراجع

[1] B. R. Hunt, R. L. Lipsman, and J. M. Rosenberg, *A Guide to MATLAB For Beginners and Experienced Users*. Cambridge: Cambridge University Press, 2014.

[2] A. Gilat, *MATLAB: An Introduction with Applications*. Hoboken, NJ: John Wiley & Sons, Inc., 2017.