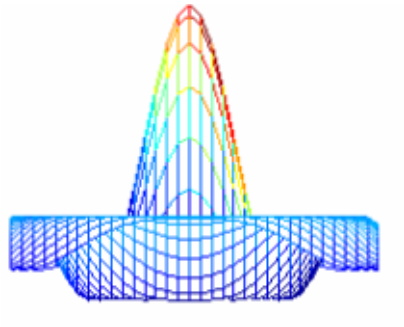


کاربرد کامپیوتر در مهندسی

مصطفی همت‌آبادی

دانشگاه صنعتی شیراز

ویرایش ٤ پائیز ۱۳۸۵

بخش یک MATLAB

فصل ۱ کلیات و اصول

۱-۱ مقدمه

واژه MATLAB

کلمه فوق، سر- واژه‌ی عبارت MATrix LABoratory است، که در این متن به‌شکل MATLAB یا متلب می‌آید. برای اجرای مثال‌ها و تمرین‌های این متن از MATLAB 6.5, Release 13 استفاده شده، و استفاده از آن (یا ویراست‌های بالاتر) به دانشجویان توصیه می‌شود.

سیمای MATLAB

MATLAB زبانی است که کاربرد کامپیوتر در مهندسی را با کارائی بالا تضمین کرده و امکانات محاسباتی، تصویری، و برنامه‌نویسی را در محیطی آسان و آشنا فراهم می‌کند. کارائی MATLAB در مقوله‌هائی نظیر: محاسبات ریاضی، دسترسی به/ و آنالیز داده‌ها، مُدل‌سازی و شبیه‌سازی، گرافیک، و تولید نرم‌افزار (حتی برای محیط ویندوز) به اثبات رسیده است. این زبان با توجه به نظرات کاربران دانشگاهی و صنعتی دست‌خوش بازنگری‌های زیادی شده و اکنون به زبان استاندارد جهت آموزش‌های مقدماتی و عالی و ابزار پژوهش و توسعه در صنایع تبدیل شده است.

MATLAB جعبه‌ابزارهائی برای کاربردهای خاص در اختیار قرار می‌دهد، که از جمله آن‌ها جعبه‌ابزار ریاضیات، کنترل، شبکه‌های عصبی، بازرگانی، . . . می‌باشند. جعبه ابزارها با زبان متلب و به صورت مجموعه‌ای از ام- فایل‌ها گسترش یافته‌اند و برای هر کاربر در زمینه تخصصی‌اش کاربرد و اهمیت زیاد دارند. امکان ساخت جعبه‌ابزارهای جدید و شخصی نیز برای کاربران پیشرفته فراهم است.

در این نرم‌افزار هر متغیر به عنوان یک ماتریس یا یک بردار (بردار ماتریس تک سطری یا تک ستونی است) شناخته می‌شود. لذا تعدادی مقدار را یک جا می‌توان به یک متغیرِ تک- نام (بدون نیاز به اعلام قبلی تعداد اعضاء) نسبت داد. این ابتکار ما را از مقدار دهی به و نمایش تک تک عناصر آرایه که در زبان‌های برنامه نویسی انجام می‌شود بی‌نیاز می‌کند.

متلب دارای پنج ویژگی شایان ذکر است:

۱- پنجره‌ی واسط کاربر Integrated Development Environment (IDE) بسیار دل‌پذیر و دست‌یافتنی که از امتیازات برنامه‌نویسی متنی و گرافیکی هر دو استفاده می‌کند. این واسط کاربر شامل پنجره‌های: فرمان، دیرکتوری جاری، تاریخچه فرمان، فضای کار، . . . است. پنجره‌ی فرمان دستورات را به صورت کنسول یا خط فرمان، مشابه برنامه‌نویسی DOS، دریافت و اجرا می‌کند. پنجره فضای کار کلیه متغیرهای فضای کار را با مشخصات آن‌ها نمایش می‌دهد. پنجره‌های واسط کاربر دینامیک بوده و ممکن است خود شامل پنجره‌های فرعی باشند. همیشه با اجرای زیرمنیوی Default از منیوی اصلی می‌توان چهار پنجره‌ی: Command, Command History, Workspace, Current Directory را که کاربردی‌تر هستند، ظاهر کرد و در دسترس داشت.

۲- کتاب‌خانه‌ی عظیمی از توابع مقدماتی و پیشرفته.

۳- زبان قوی هم برای فرامین کوتاه و یک‌بار مصرف و هم برای برنامه‌نویسی بزرگ و کاربردی.

٤- روش‌های متعدد ترسیمات دوبعدی و سه‌بعدی.

۵- واسط میان‌برنامه‌ای Application Program Interface (API) که امکان فراخوانی برنامه‌های متلب از زبان‌های C و Fortran ، تبدیل فایل‌های متلب (ام- فایل‌ها) به زبان C ، و استفاده از موتور محاسباتی متلب در این زبان‌ها را فراهم می‌کند. با اسفاده از تبدیلات فوق می‌توان برنامه‌های نوشته شده

در محیط متلب را به صورت اجرائیِ کنسولی (قابل اجرا از خط فرمان سیستم عامل) درآورد. هم‌چنین امکان تهیه فایل‌های اجرائی با واسط گرافیکی برای ویندوز وجود دارد.

٦- دارا بودن راهنمای جامع و کاملِ کنارِ دست و وجود مدارک راهنمای قابل چاپ به صورت PDF که توسط شرکت Mathworks (سازنده متلب) ارائه شده‌اند.

روش این کتاب

روش کار ما در این متن یادگیری- با- مثال Learn by Examples است. ازاین‌رو اجرای متلب، هم‌زمان با مطالعه متن اکیداً توصیه می‌شود. در مثال‌های کاربردی نوع کاربرد مشخص است.

- در کلیه مثال‌ها، دستورات فوری که روی پنجره فرمان نوشته می‌شوند، بعداز علامت << (نشانه سطر فرمان Command Prompt) آمده‌اند. در انتهای هر دستور فوری بایستی کلید <Enter> زده شود.

- سطرهائی که در ادیتور برنامه‌نویسی به صورت ام- فایل نوشته می‌شوند بدون علامت << آمده‌اند. برای اجرای آن‌ها باید از داخل ادیتور دکمه Run یا کلید F5 را زد.

- نتایجی که روی پنجره فرمان یا پنجره‌های دیگر ظاهر می‌شوند (تحت نام خروجی) داخل کادرِ خط‌چین آمده‌اند.

- اگر در پایان بعضی از دستورها علامت سمی‌کالن (نقطه- ویرگول) قرار دهیم نتیجه اجرا پس از زدن کلید <Enter> در پنجره فرمان ظاهر نمی‌شود. این موضوع در مورد دستوراتی نظیر ... , input() , disp() و دستورات نمایش گراف صدق نمی‌کند.

- علامت % برای توضیحات Comments می‌آید و آن‌چه بعداز آن نوشته شود جزو دستورات منظور نمی‌شود.

- خروجی‌های گرافیکی در پنجره گرافیک ظاهر می‌شوند.

- تمرین‌های پایان درس از نکات برجسته متن گل‌چین شده، نقش خودآزما را دارند. انجام آن‌ها ضروری است.

علائم و دورچین های استفاده شده در متن

در مثال‌های این کتاب برای تفکیک قسمت‌ها و ایجاد خوانائی بیشتر:

الف- در مقابل دستورات پنجره فرمان علامت << آمده است.

ب- اگر علامت فوق در مقابل دستوراتی نباشد به این معنی است که دستوران به صورت برنامه در ام- فایل نوشته و اجرا شده‌اند.

ج- هم‌چنین از دورچین‌های زیر استفاده شده است.

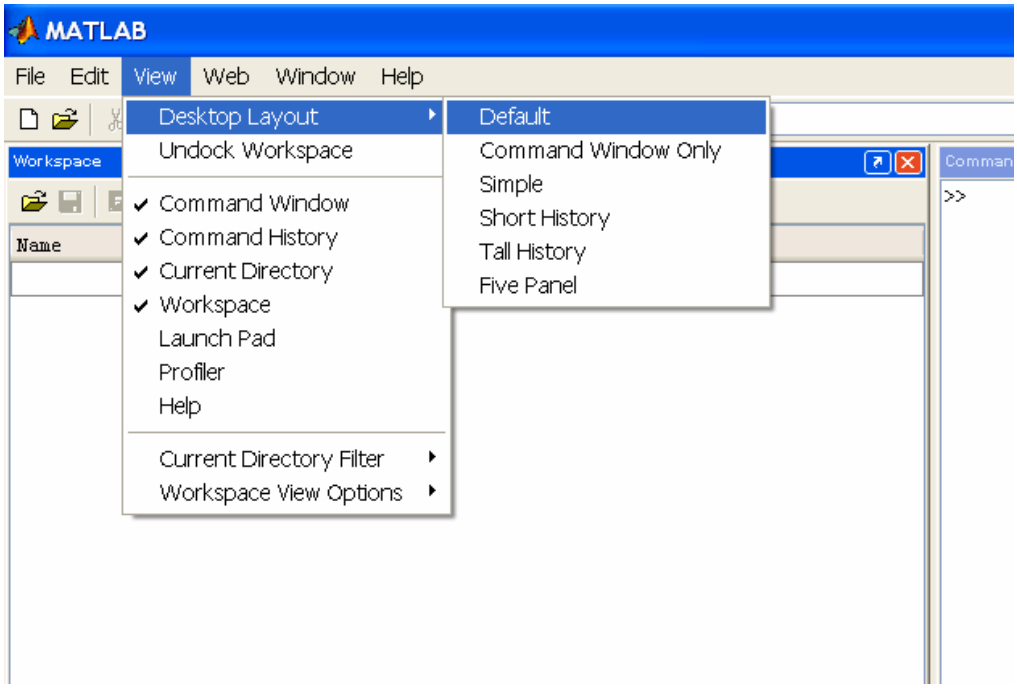
خروجی: نتیجه‌ی عملیات در پنجره فرمان یا **خروجیِ** تصویری یک برنامه

راهنما: نکات آموزشی داخل مثال‌ها یا جزئیاتی که در متن ذکر نشده‌اند

دقت کنید: مواردی که نیاز به تأکید بیشتر دارند و جلب توجه به موارد مهم

۱-۲ محیط کار متلب

پس از ورود به محیط کار متلب، هر زمان که گزینه Default را مطابق شکل زیر کلیک کنیم:



شکل ۱-۱ انتخاب گزینه Default در متلب ویراست 6.5

سه پنجره باز می‌شوند که بیشترین مورد استفاده را دارند:

۱- Command Window پنجره فرمان که محل اجرای کلیه فرمان‌ها و مقدار دهی‌های فوری به متغیرها، و نمایش نتایج اجرا است.

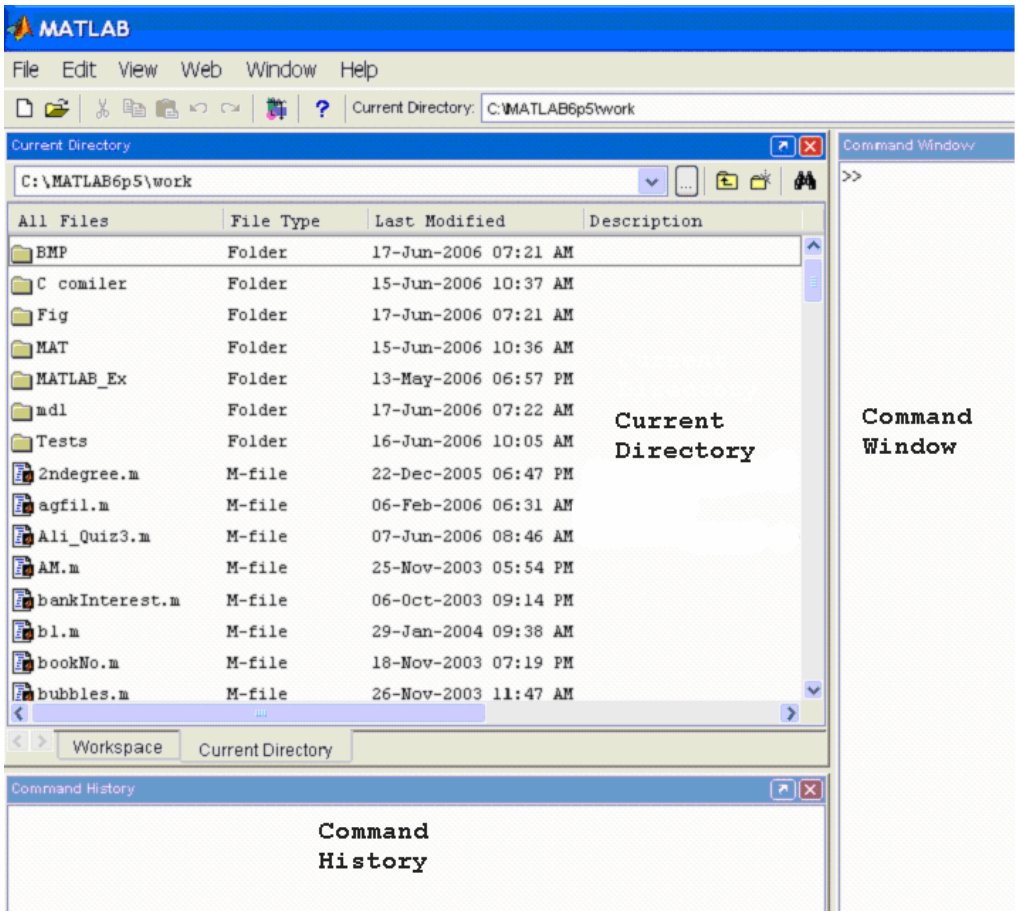
۲- Workspace پنجره فضای کار که نمایش دهنده کلیه متغیرهای تعریف شده از داخل پنجره فرمان (یا به طرق دیگر) و مشخصه‌های آن‌ها است.

می‌توان به جای پنجره فضای کار، پنجره دیرکتوری جاری Current Durectory را فعال کرد که کلیه فایل‌های موجود در دیرکتوری مورد رجوع متلب را نشان می‌دهد.

۳- Command History پنجره یادداشت فرمان که محل نگه‌داری آنچه تاکنون در پنجره فرمان آمده می‌باشد.

پنجره‌های متلب را می‌توان به انواع شکل‌های دلخواه درآورد، مثلاً با کلیک پیکان مورب در گوشه راست هر پنجره می‌توان آن را به تنها پنجره مورد مشاهده تبدیل کرد.لیکن ما معمولاً در حالت Default کار می‌کنیم.

شکل ۱-۲ پنجره‌های حالت Default در متلب



شکل ۱-۳ پنجره دیرکتوری جاری، دیرکتوری مورد رجوع متلب

۱-۳ آغاز کار با متلب

ماتریس Matrix

متلب با ماتریس یا بردار کار می‌کند.اغلب متغیرها و مقادیر آن‌ها در متلب فرم ماتریس دو بعدی با m ردیف (سطر) و n ستون دارند، که به آن ماتریس m×n گفته می‌شود (که در نام‌گذاری این نرم‌افزار MATrix LABoratory مستتر است). هر عضو ماتریس (Matrix Member) را عنصر ماتریس یا درایه می‌گوییم. یک ماتریس با تعدادی سطر (ردیف)، تعدادی ستون، و تعدادی عضو (ما بیشتر با اعضاء عددی سروکار داریم)، تحت یک نام واحد شناخته می‌شود. نام ماتریس و مقدار آن هم‌زمان در خط فرمان نوشته می‌شوند و برخلاف زبان‌های دیگر نیاز به اعلام قبلی نام و تعداد اعضاء وجود ندارد. مفهوم ماتریس و بردار که معادل مفهوم آرایه هستند، ما را از مقدار دهی به تک تک عناصر آرایه و نمایش تک به تک آن‌ها که در زبان‌های برنامه نویسی معمولاً از طرقی مانند استفاده از حلقه‌های تکرار انجام می‌شود بی-نیاز می‌کند. هر ماتریس کاملاً دینامیک است و پس‌از تعریف می‌توان مقدار و تعداد سطر و ستون آن‌را تغییر داد.
ماتریس سه بعدی با m ردیف (سطر) و n ستون و p صفحه نیز وجود دارد که بعداً به آن اشاره خواهد شد.
در متلب انواع ماتریس نظیر بردار، ماتریس عددی با اعداد مختلط، ماتریس کاراکتری، ماتریس نمادین، ماتریس ساختاری، ماتریس سلولی، . . . نیز وجود دارند.

بردار

بردار، یک ماتریس تک سطری (ردیفی) یا تک ستونی است که دارای تعدادی عضو (معمولاً مقادیر عددی) بوده و تحت یک نام واحد شناخته می‌شود. معمولاً کلمه بردار به تنهائی برای بردار ردیفی به کار می‌رود. در متلب بردارها کاربرد زیادی دارند. نام بردار و مقدار آن هم‌زمان در خط فرمان نوشته می‌شوند و برخلاف زبان‌های دیگر نیاز به اعلام قبلی نام و تعداد اعضاء وجود ندارد. مفهوم بردار در متلب ما را از مقدار دهی به تک تک عناصر آرایه که در زبان‌های برنامه نویسی معمول است بی‌نیاز می‌کند. از طرف دیگر برای رسم ترسیمات، یک متغیر برداری می‌تواند به تنهائی مقادیر روی یک محور را پوشش دهد.
هر بردار ردیفی یک ماتریس دو بعدی 1×n (تک‌ردیفی) و هر بردار ستونی یک ماتریس دو بعدی m×1 (تک‌ستونی) است. پس یکی از ابعاد هر نوع بردار یک است. عدد تکی یا ***اسکالر*** یک ماتریس 1×1 است.

آرایه، متغیر، عملیات آرایه ای

همان‌گونه که در سایر زبان‌های برنامه‌نویسی تعریف می‌شود، آرایه یک نام کلی برای انواع ماتریس، بردار و کلاً مجموعه‌های یک تا چندین عنصری است. این عناصر در تعدادی ردیف، ستون، و صفحه قرار گرفته‌اند. در این متن کلمه آرایه اعم از کلمات ماتریس، بردار، و متغیر به کار می‌رود، و کلمات متغیر و آرایه نیز به جای هم به کار می‌روند.
در متلب متغیرِ تک‌مقداری (اسکالر) یک آرایه‌ی تک عنصری است.
هم‌چنین عملیاتی که بر روی تک‌تک عناصر آرایه انجام شود، عملیات آرایه‌ای نام دارد که در مقابل عملیات ماتریسی قرار می‌گیرد (شرح بیشتر بعداً می‌آید).

تعریف و مقدار دهی به یک متغیر

A را که یک ماتریس ۲×۳ (m = 2, n = 3) عددی است، در جبر این‌گونه می‌نویسیم:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

معرفی ماتریس فوق declaration ، و مقدار دهی هم‌زمان آن‌را در متلب تعریف متغیر definition، می-گوئیم. ماتریس فوق دو بُعدی است.

مثال:

ماتریس فوق را در متلب تعریف، و بُعد آن را با تابع داخلی ndims () تعیین کنید (برای اطلاع بیشتر از ndims () به مبحث توابع اندازه‌گیر در همین فصل مراجعه کنید).

```
>> A = [1 2 3; 4 5 6]
A = 1   2   3
    4   5   6

>> n = ndims(A)
   n = 2
```

مشاهده یک متغیر و نوشتن توضیح

برای مشاهده مقدار هر متغیر در پنجره فرمان کافی است نام آن‌را وارد کنیم.

اگر در پایان یک سطر علامت سمی‌کالن (نقطه- ویرگول) قرار دهیم نتیجه اجرا در پنجره فرمان ظاهر نمی‌شود. اما مقدار در حافظه می‌ماند.

مشخصه‌های نام، نوع (عددی، منطقی، نمادین، رشته‌ای، ...)، اندازه (تعداد ردیف و ستون)، و تعداد بایتِ متغیر(آرایه)های تعریف شده، در ***پنجره فضای کار (فضای حافظه) Workspace*** نمایش داده می‌شوند

علامت درصد % برای نوشتن توضیحات Comments به کار می‌رود.

مثال ها:

وارد کردن نام ماتریسی که قبلاً تعریف شده

```
>> A
A =   1     2     3
      4     5     6
```

تعریف بردار با نقطه‌ویرگول و نوشتن توضیح

```
>> M = [10.2 3.4 5.6]; %  این متغیر در حافظه نگه‌داری می‌شود، اما نمایش داده نمی‌شود
>> M %  برای مشاهده مقدار متغیر
M = 10.2   3.4   5.6
```

بردار- هندسی

بردارهای هندسی یا فیزیکی، درون مختصات قطبی (مدور) دارای دو مختصه بزرگیMagnitude ، و زاویه هستند، و درون مختصات قائم دارای دو مختصه طول X و عرض Y می‌باشند. این بردارها را دراین متن ***بردار- هندسی*** نامیده‌ایم.

این نام‌گذاری برای ایجاد تمایز مابین مفهوم بردار در متلب با بردارِ فیزیکی یا هندسی است. گاهی از اوقات نیز با وام از معادل الکتریکی آن را فازور نام داده‌ایم.

آرایه سلولی Cell Array

آرایه‌ای که هر درایه آن یک آرایه دیگر باشد، آرایه سلولی Cell Array نام دارد. برای اطلاع بیشتر به مدارک متلب مراجعه کنید.

مثال:

یک آرایه سلولی که دارای سه نوع آرایه در هر سلول باشد ایجاد کنید.

```
C = {logical([1 0 1 1 0]) 'Array' [1 2;3 4]}
C = [1x5 logical]    'Array'    [2x2 double]
```

توابع اندازه گیر

توابع `length()`, `size()`, `ndims()` و نظائرشان را که در سنجش کمیت ماتریس‌ها به کار می‌روند، در این متن توابع اندازه‌گیر نامیده‌ایم.

تابع `size()` تعداد سطر و ستون (و تعداد صفحه در مورد ماتریسهای سه‌بُعدی) را می‌دهد. تابع `length()` معادل `max(size())` است که در مورد بردارها تعداد عناصر را که همان تعداد ستون‌ها است می‌دهد. تابع `ndims()` معادل `length(size())` است که برای اسکالر و بردار و ماتریس دو بعدی دارای مقدار 2 و برای ماتریس‌های سه بعدی دارای مقدار 3 است.

باید توجه داشت که اندازه `size` و نوع `type` هر آرایه در پنجره `Workspace` نشان داده می‌شود.

مثال:

یک بردار ردیفی ٤ عضوی تولید کنید. اندازه، طول و بُعد آن‌را چاپ کنید.

```
>> rv = [2 -5 16 7.6];
>> size(rv)
   ans = 1     4
>> L = length(rv)
   L = 4
>> m = ndims(rv)
   m = 2
```

متغیر داخلی ans

اگر به آخرین مقدار حساب شده نامی نداده باشیم، در متغیر داخلی `ans` قرار می‌گیرد..

مثال ها:

بردار بدون نام

```
>> [-2 3.4 56]
ans = -2.0000    3.4000   56.0000
```

درایه واقع در ردیف۲ ستون۳ ماتریس **A**

```
>> A = [1 2 3; 4 5 6]
>> A(2,3)
ans = 6
```

اعلام نوع متغیر

همان‌طور که گفته شد در متلب نیازی به اعلام یا معرفیِ `declaration` متغیرها قبل از کاربرد آن‌ها نیست (بر خلاف C++)، بلکه به هنگام مقدار دهی به یک متغیر، نوع آن خودبه‌خود تعیین می‌شود.

۴-۱ نوشتن دستورات و مشاهده نتایج

پنجره فوری و پنجره برنامه

دستورات متلب به صورت اجرای فوری `Immediate` در پنجره فرمان، یا به صورت برنامه در پنجره برنامه‌نویسی می‌آیند. فایل‌های برنامه ام- فایل نام دارند (شرح ام- فایل بعداً می‌آید). علامت << نشانه سطر فرمان `Command Prompt` است. هر دستور در پنجره فرمان با فشردن کلید <Enter> انجام می‌شود.

شکستن یک دستور در چند سطر

متغیر A مثال قبل را می‌توان با شکستن سطر به این صورت:

```
>> A = [1 2 3;
```

```
4 5 6]
A =  1     2     3
     4     5     6
```

یا به این صورت زیر مقدار دهی کرد:

```
>> A = [1 2 3
4 5 6]
A =  1     2     3
     4     5     6
```

نوشتن کلیه فرمان‌ها و متغیرها را می توان با گذاشتن . . .(سه نقطه) در پایان سطر، در سطر بعد ادامه داد. اما دقت کنید که گذاشتن سه نقطه دنبال‌هم قرار می‌دهد، لذا B مثال زیر یک ماتریس تک‌سطری (بردار ردیفی) خواهد بود و نه ماتریس ۳×۲ .

مثال:

```
>> B = [10 20 30 ...
40 50 60]
B = 10    20    30    40    50    60
```

نوشتن چند دستور در یک سطر

اگر بعداز هر دستور کاما بگذاریم می‌توانیم دستور بعدی را در ادامه بنویسیم، اما نتیجه اجرا در پنجره فرمان نمایش داده خواهد شد. آوردن سمی‌کالن (نقطه- ویرگول) باعث می‌شود نتیجه اجرا نشان داده نشود.

مثال:

```
>> x = 3, y = 34; z = y ^ x
x = 3
z = 39304
```

تفاوتِ حروف بزرگ و کوچک

متلب در نام‌گذاری متغیرها مابین حروف بزرگ و کوچک انگلیسی فرق می‌گذارد (Case Sensitive است).

مثال:

```
>> New = 3;
>> new
??? Undefined function or variable 'new'.
```

قطع اجرا

هرگاه اجرای دستورات در پنجره فرمان طولانی شد، در صورت تمایل برای قطع اجرا <Ctrl+C> را بزنید.

دستور more

این دستور خروجی‌های طولانی را صفحه به صفحه نمایش میدهد. اگر زمان نمایش خروجی طولانی شد، برای قطع نمایش و برگشت به خط فرمان Ctrl+C را بزنید.

مثال:

```
>> more on, rand(10000,1)
>> more off
```

اجرا و ضبط دستورات قبلی

تکرار دستورات قبلی که در پنجره فرمان نوشته شده‌اند، با کلیدهای ↑ و ↓ انجام میشود. برای اجرای یک سطر کلید <Enter> و برای حذف یک سطر تکراری نادل‌خواه کلید <Esc> یا در ابتدای سطر کلید <Ctrl+K> را بزنید. دستورات وارد شده در پنجره فرمان خود به خود نگه‌داری می‌شوند و برای اجراهای بعدی متلب هم قابل استفاده هستند. پنجره Command History هم فرمان‌های قبلی را نگه می‌دارد.

اگر سطر فرمانی در هریک از پنجره‌های فرمان یا تاریخچه فرمان یا ادیتور ام- فایل مارک شود، با راست کلیک و انتخاب Evaluate Selection آن دستور اجرا خواهد شد.

می‌توان با اجرای notebook از درون متلب و تمهیدات مربوط به آن دستورات متلب را به داخل Microsoft Word برد.

دستور زیر سطور command window را در فایل filename نگاهداری می‌کند:

```
>> diary filename
```

فرمان زیر

```
>> diary off
```

ضبطِ پنجره فرمان را متوقف می‌کند. اجرای مجدد diary سطور را به انتهای فایل filename اضافه می‌کند.

اجرای دستورات سیستم عامل

دستورات سیستمی نظیر dir, time, date را با گذاشتن علامت ! در مقابل آن‌ها از داخل پنجره فرمان اجرا کنید.

تابع eval()

اگر یک عبارت قانونیِ متلب را به صورت رشته در داخل دو آپوستروف بنویسیم text macro نام می‌گیرد.

تابع eval () صحت و سُقم چنین عبارتی را تعیین می‌کند. در صورت صحت، تابع eval () مقدارِ عبارت مورد تست را برمی‌گرداند، وگرنه پیغام خطای مربوطه را چاپ می‌کند. می‌توان آرگومان دومی برای eval () قرار داد که به جای پیغام خطا چاپ شود. هر دو آرگومان eval () باید به صورت رشته باشند.

مثال ها:

عبارت صحیح

```
>> x = pi/6;
>> sx = 'sin(x)^3 + cos(x)^3';
>> eval(sx)
ans = 0.7745
```

عبارت ناصحیح با پیغام متلب

```
sx1 = 'sin^2(x) + cos^2(x)' ;
>> eval(sx1)
Error: Missing operator, comma, or semicolon.
```

عبارت ناصحیح با پیغام کاربر

```
>> eval(sx1,'disp(''wrong'')')
wrong
```

سؤال: در مثال فوق اگر مقدار x را از workspace پاک و eval را اجرا کنیم نتیجه چیست؟ امتحان کنید.

دستور lasterr

دستور lasterr آخرین پیغام خطای داخلی را نشان می‌دهد.

مثال:

```
>> lasterr
ans = Error: Missing operator, comma, or semicolon.
```

بلوک try...catch

بلوک try...catch...end برای به دام انداختن خطا error trapping به کار می‌رود.

در زیرِ try دستوراتی می‌آیند که در صورت عدم وقوع خطا باید اجرا شوند، و در زیرِ catch دستوراتی می‌آیند که در صورت وقوع خطا باید اجرا شوند. برای اطلاع بیشتر به مثال حلقه‌های تکرار مرجعه کنید.

تفاوت دستورات پنجره فرمان و برنامه

در پنجره فرمان دستور فوراً اجرا می‌شود، اما نسبت به پنجره های immediate بعضی از نرم‌افزارها امتیازاتی دارد، مثلاً دستورات حلقه، یا فرمانهای دنبال هم از درون پنجره فرمان قابل اجرا هستند.

برنامه‌ها در فایل‌هائی به نام M-File ضبط می‌شوند. برنامه‌هائی که در آن‌ها تعدادی فرمان یک‌جا اجرا می‌شوند Script M-File و برنامه‌هائی که یک تابع را تعریف می‌کنند Function M-File نام دارند.

اغلب قواعد نوشتاری و اجرائی که در بالا آمد در مورد برنامه نویسی نیز صدق می‌کنند.

دستورهائی در مورد متغیرها

نام دستور	نتیجه اجرا
who	نام متغیرها را می‌دهد
whos	نام و اطلاعات بیشتری از مشخصه‌های متغیرها می‌دهد
clear var	یک متغیر را پاک می‌کند
clear	کلیه متغیرهای تعریف شده را پاک می‌کند
clear all	حافظه را کلاً پاک می‌کند
pack	حافظه را منظم و جمع و جور می‌کند

۱-۵ دریافت اطلاع و راهنمائی از مدارک راهنمای متلب

دستورات زیر شامل برخی از روش‌های استفاده از مدارک غنی متلب است. استفاده مرتب از مدارک راهنمای متلب به کاربران و دانشجویان توصیه می‌شود. اغلب مدارک راهنما با فرمت PDF نیز قابل دسترسی هستند.

مثالِ دستور	نتیجه اجرا
lookfor word	جستجو برای یک کلمه داخل متون متلب
help rand	راهنمائی یک عبارت یا دستور مانند rand
type realmin	نمایش محتوای یک تابع کتاب‌خانه‌ای یا تابع کاربر- تعریف در دیرکتوری جاری
info	اطلاعات راجع به متلب
[st,n]=computer	اطلاع راجع به کامپیوتر و تعداد عناصر مجاز یک ماتریس
version, ver	ویرایشِ متلب
doc sin	دریافت راهنما از پنجره MATLAB Help
help	نمایش راهنمای متلب
helpwin	نمایش راهنمای متلب در پنجره MATLAB Help

۱-۶ تمرین

۱- یک ماتریس ۳×٤ عددی تعریف و بُعد آن را تعیین کنید.

۲- یک بردار ردیفی عددی ٦ عضوی تولید کنید.

۳- یک آرایه سلولی که دارای انواع منطقی، رشته‌ای، عددی، و سمبلیک (نمادین) باشد ایجاد کنید. اندازه، طول و بُعد آن‌را چاپ کنید.

٤- دو بردار عددی و حاصل‌جمع آن دو را در یک سطر ایجاد کنید.

٥- دستورات `dir, time, date` را با گذاشتن علامت ! در مقابل آن‌ها از داخل پنجره فرمان اجرا کنید.

٦- دستورات `who` ، `whos` ، `clear var` را در مورد متغیرهائی که به وجود آورده‌اید، اجرا کنید.

۷- دستور `lookfor` را برای `exp` اجرا کنید (اگر طولانی شد از `Ctrl-C` کمک بگیرید).

۸- دستور `help` را برای چند تابع مثلثاتی و برای `magic` اجرا کنید.

۹- دستور `type` را برای `realmax, realmin` اجرا کنید.

۱۰- دستور `doc tan` و `doc randn` و `helpwin` را اجرا کنید.

۱۱- دستورات `info, [st,n]=computer, version, ver` را اجرا کنید.

۱۲- یک دستور غلط را با `eval` و `lasterr` امتحان کنید.

فصل ۲ آشنائی با متغیرها و توابع

۲-۱ آشنائی با انواع متغیر و داده

انواع داده

در متلب داده‌های مختلفی وجود دارند از جمله: داده‌های عددی که بیشتر به آن‌ها خواهیم پرداخت، داده‌های کاراکتری و منطقی که در واقع نوعی داده عددی هستند. ماتریس سلولی `cell matrix` که داده‌هائی از انواع مختلف را نگه-داری می‌کند، ماتریس ساختاری که ماتریس‌هائی از انواع مختلف را نگه‌داری می‌کند، `inline` ، `function_handle` ، `...`

داده‌های عددی، و دستور ()isa

به علت این‌که متلب یک نرم‌افزار مهندسی است، داده‌های عددی در آن اهمیت خاصی دارند. پیش‌فرض متلب این گونه است که هر متغیر عددی را به هنگام ورود (معرفی، یا اعلام نام و مقدار هم‌زمان)، با دقت افزوده (مضاعف) `double` در حافظه نگه‌داری می‌کند. دستور `isa()` نوع متغیر یا داده را امتحان می‌کند برای اطلاع بیشتر از انواع داده در متلب `help datatypes` را اجرا کنید. مشخصه‌های بعضی از انواع عددی در جدول زیر آمده است.

نام	دستور تبدیل نوع	جا در حافظه (بایت)	تعداد ارقام با معنی بعداز ممیز
دقت مضاعف `double`	`double()`	8	15
دقت ساده `single` *	`single()`	4	7
صحیح بزرگ `long integer` **	`int16()`	2	N/A
صحیح کوچک `short integer` ***	`int8()`	1	N/A
منطقی `logical`	`logical()`	1	N/A
کاراکتری `character` ****	`char()`	1	N/A

نکاتی پیرامون single

دقت ساده از لحاظ بازه محدودتر از دقت مضاعف است.

اغلب عملیات ریاضی برای نوع `single` تعریف نشده اند. اما اگر مایل باشیم مثلاً عمل جمع را برای نوع `single` تعریف کنیم، باید یک متد (تابع) برای این کار تعریف و در دیرکتوری `@single` (منشعب از دیرکتوری `work`) قرار دهیم. برای توضیح بیشتر به مثال فصل توابع کاربر- تعریف مراجعه کنید.

نکاتی پیرامون int

تبدیل به هر نوع عدد صحیح ارقام بعداز ممیز را حذف می‌کند.

بازه عدد صحیحِ هشت بیتی ۱۲۸- تا ۱۲۷+ است.

اغلب عملیات ریاضی برای نوع `int` تعریف نشده اند. اگر مایل باشیم مثلاً عمل ضرب را برای نوع `int` تعریف کنیم، باید با روشی مشابه آن‌چه در مورد `singel` آمد عمل کنیم.

نکاتی پیرامون char و ترکیب انواع

داده کاراکتری در محدوده اسکی معادل عدد صحیح هشت بیتی است.

در ترکیب کاراکتر، منطق، و عدد، برتری با عدد بوده و حاصل از نوع عددی خواهد بود.

مثال ها:

داده اعشاری **double**

```
>> format long e % برای نمایش ارقام بیشتری از یک عدد
>> e = exp(1)
```

برای به دست آوردن عدد نپری از تابع کتابخانه‌ای exp() استفاده کرده‌ایم %

```
e = 2.718281828459046e+000
>> isa(e,'double')
ans = 1
>> pi, 4*atan(1), imag(log(-1))
ans = 3.141592653589793e+000 ans = 3.141592653589793e+000
ans = 3.141592653589793e+000
```

داده اعشاری **single**

```
>> y = single(e)
y = 2.718281745910645e+000
>> isa(y,'single')
ans = 1
```

نوع single عدد اعشاری است که تا هفت رقم بعداز ممیز دقیق است، لذا نمایش آن فقط تا رقم هفتم اعتبار دارد، و ارقام بعداز آن فاقد اعتبار هستند (با نمایش e مقایسه کنید).

عملیات با نوع **single**

اغلب عملیات ریاضی برای نوع single تعریف نشده اند.

```
>> si1 = single(2.2), si2 = single(3.5)
si1 = 2.2000      si2 = 3.5000
>> si1 * si2
??? Error using ==> *
Function '*' is not defined for values of class 'single'.
```

انواع اعداد صحیح

```
>> format
>> x1 = 125.6;   % طبق پیش‌فرض متلب به صورت عدد اعشاری با دقت افزوده نگه‌داری می‌شود
>> ix1 = int8(x1)
ix1 = 125
>> isa(ix1,'int8')
ans = 1
>> ix2 = int16(x1)
ix2 = 125
>> isa(ix2,'int16')
ans = 1
```

عملیات با نوع **int**

اغلب عملیات ریاضی برای نوع int تعریف نشده اند.

```
>> it1 = int16(14.4), it2=int16(12.3)
it1 = 14     it2 = 12
>> it1 + it2
??? Error using ==> +
Function '+' is not defined for values of class 'int16'.
```

داده کاراکتری و ترکیب انواع

```
>> x = 66; % double
>> cx = char(x) % character
cx = B
```

در ترکیب کاراکتر، منطق، و عدد، حاصل عددی است

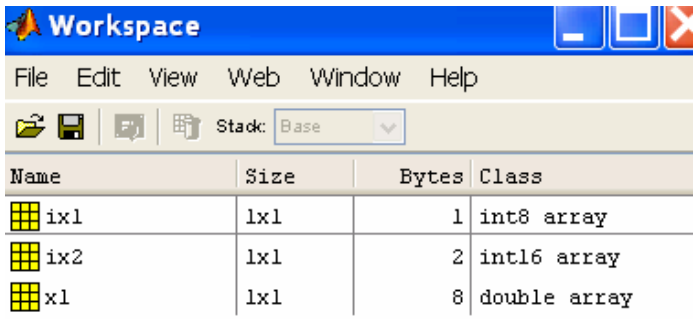
```
>> ar = 'A' + logical(1) + 1.3 %
var = 67.3000
```

داده منطقی

```
>> t = [0 1]; % double
>> tL = logical(t)
tL = 0     1
```

مشاهده در پنجره فضای کار

داده‌های تعریف شده در محیط متلب در فضای کار (مشابه شکل ۲-۱) نمایش داده می‌شوند. لذا توصیه می‌شود همواره از انتخاب View از منوی اصلی، پنجره Workspace را تیک بزنید تا بتوانید مشخصه‌های متغیرهای خود را در فضای حافظه مشاهده نمائید. اگر به روی یک متغیر در پنجره Workspace کلیک-کلیک کنید، دسترسی بیش‌تری به‌آن پیدا کرده، حتی می‌توانید آن را ادیت کنید (امتحان کنید).



شکل ۲-۱

متغیرهای داخلی یا توکار متلب

بعضی از متغیرها که در داخل متلب پیش- تعریف شده‌اند، در جدول زیر آمده‌اند:

نام	تعریف	مقدار (بر روی کامپیوتر من)
ans	متغیر داخلی که آخرین مقدار حساب شده را نگه می‌دارد	آخرین مقدار حساب شده
eps	فاصله بین 1 و بزرگ‌ترین عدد اعشاری بعد از آن	pow2(1,-52) 2^(-52) 2.220446049250313e-016
realmax	بزرگ‌ترین عدد ممکن بر روی کامپیوتر شما	1.797693134862316e+308
realmin	کوچک‌ترین عدد ممکن بر روی کامپیوتر شما	2.225073858507201e-308
pi	عدد پی	4*atan(1) imag(log(-1)) 3.141592653589793e+000
Inf, inf	Infinity	نتیجه تقسیم عدد بر صفر
NaN,nan	Not a Number	تقسیم inf بر inf و صفر بر صفر

* تابع pow2(m,n) معادل است با $m \times 2^{n}$.

دستور فرمت format

دستور format فقط روش نمایش اعداد در صفحه نمایش را تعیین می‌کند، و تأثیری در دقت نگه‌داری اعداد در حافظه ندارد. فقط اعداد صحیح قابل تبدیل به و نمایش در مبنای شانزده هستند، لذا هر عدد را باید ابتدا به عدد صحیح تبدیل و سپس آن‌را با فرمت هگز نمایش داد.

در جدول زیر چند نوع روش نمایش اعداد آمده است، برای اطلاع بیشتر help format را اجرا کنید.

نام	دستور	ارقام بعد از نقطه- اعشار
فرمت بانکی (فقط برای اعداد حقیقی)	format bank	۲
فرمت کوتاه	format format short	٤
فرمت بلند	format long	١٤
فرمت بلندِ نمائی	format long e	١٥
فرمت مبنای شانزده (فقط برای اعداد صحیح)	format hex	N/A
نمایش به صورت کسر متعارفی	format rational	N/A

مثال ها:

ارزش محاسباتی **eps**

با توجه به این‌که دستور format long e خروجی را با تعداد ارقام زیاد و به صورت نمائی چاپ می‌کند.

eps - 1 را با این فرمت نمایش دهید.

```
>> format long e
>> a = 1 - eps
a = 9.999999999999998e-001
```

ارزش محاسباتی **realmin**

با فرمت فوق مقدار eps و realmin + eps را نمایش دهید.

```
>> eps
ans = 2.220446049250313e-016

>> realmin + eps
ans = 2.220446049250313e-016
```

هر دو مقدار مساوی هستند، لذا نتیجه می‌گیریم که کوچک‌ترین عددی که ارزش محاسباتی دارد eps است.

مبنای شانزده

```
>> ix = int8(125);
>> format hex
>> ix
ix1 = 7d
```

فرمت کسر متعارفی

```
>> format rational
>> pi
ans = 355/113
>> format
```

ضریب در نمایش اعداد

در صورت دور بودن مقدار عناصر یک بردار از هم، اعداد با ضریبی به نام scale factor در مقابلشان نمایش داده می‌شوند، که بایستی این ضریب را در عدد ضرب کرد.

مثال:

```
>> format long
>> x = [1e6 1e7 1e-1]
x = 1.0e+007 *    0.100000000    1.00000000    0.00000001
      ↑
scale factor
```

۲-۲ عملگرها

عملگرهای آرایه ای

این عملگرها بر روی آرایه‌های مورد عمل به صورت عنصر به عنصر عمل می‌کنند و در مقابل عملگرهای ماتریسی قرار دارند که تعاریف خاص خود را دارند. جمع و تفریق آرایه‌ای و ماتریسی یکسان هستند. ماتریس‌ها یا بردارهائی که در عمل آرایه‌ای مشارکت دارند باید همسان باشند.

عملگر ترانهاد Transpose با علامت آپوستروف، جای ستون و ردیف ماتریس را عوض می‌کند، و در مورد بردارها بردار ردیفی را به بردار ستونی (و بالعکس) تبدیل می‌کند.این عملگرها در جدول زیر آمده‌اند.

جمع	تفریق	توان	ضرب	تقسیم (معمولی)	تقسیم راست به چپ	ترانهاد Transpose
+	-	.^	.*	./	.\	'

مثال ها:

ضرب و تقسیم آرایه‌ای

```
>> a = [2 4 8];
>> b = [3 2 2];
>> a .* b
ans = 6     8    16
>> a ./ b
ans = 0.6667    2.0000    4.0000
>> ar = a .\ b
ar = 1.5000    0.5000    0.2500
```

ترانهاد (ترانسپوز)

```
>> a = [2 3 -4.5] '
>> at = a'
at = 2.00
     3.00
    -4.50
>> b = 0:30:180;
>> table = [b' sin(b*pi/180)']
table = 0         0
        30.000    0.500
        60.000    0.866
        90.000    1.000
       120.000    0.866
       150.000    0.500
       180.000    0.000
```

در مثال فوق **table** یک ماتریس عددی است و زوایا و سینوس زوایای بین صفر و ۱۸۰ درجه را نگه‌می‌دارد.

اولویت عملگرها

اولویت	ترانهاد	پرانتز	توان	ضرب و تقسیم	جمع و تفریق	کالن
اولویت در صورت هم‌راهی				چپ به راست	چپ به راست	

۲-۳ توابع کتاب خانه ای

یکی از قابلیت‌های مهم متلب داشتن تعداد معتنابهی توابع توساخت `built in functions` یا توابع داخلی یا توابع کتاب‌خانه‌ای است، که نیازهای گوناگون انواع کاربران را عمدتاً برآورده می‌سازند. معرفی چند تابع در زیر آمده:

تولید عدد تصادفی

تابع `rand(m,n,p)` یک ماتریس تصادفی p صفحه‌ای با صفحاتی دارای m ردیف و n ستون تولید می‌کند. برای ایجاد N عدد تصادفی بین MinN و MaxN از فرمول زیر استفاده می‌کنیم:

```
rd = round((MaxN - MinN) * rand(1, N))+ MinN
```

مثال ها:

ماتریس تصادفی دو صفحه‌ای

```
>> B = rand(2,3,2)
B(:,:,1) = 0.9218    0.1763    0.9355
           0.7382    0.4057    0.9169
B(:,:,2) = 0.4103    0.0579    0.8132
           0.8936    0.3529    0.0099
```

بردار عددی با **n** عنصر تصادفی

```
>> d1 = rand(1,6)
d1 = 0.9901    0.7889    0.4387    0.4983    0.2140    0.6435
```

در مثال فوق ماتریسی با p = 1 (ماتریس دو بعدی، تک صفحه) m = 1 (تک ردیف)

n = 6 (6 ستون) تولید می‌شود.

بردار عددی با **n** عنصر تصادفی، و تعیین بزرگای هر عضو

```
>> rd = round(9 * rand(1, 6))+ 1
rd = 4    10     8     5     8     3
```

تولید آرایه با تکرار آرایه دیگر

تابع `B = repmat(A,M,N)` آرایه A را M بار افقی و N بار عمودی تکرار کرده و در آرایه B قرار می‌دهد.

مثال:

یک بردار با ده عضو متناوب صفر ویک تولید کنید.

```
>> A = [1 0];
>> B = repmat(A, 1, 5)
B = 1     0     1     0     1     0     1     0     1     0
```

توابع تحلیل داده ها

برخی از توابع که جهت تحلیل آماری مجموعه‌ای از داده‌ها به کار میروند در جدول زیر آمده‌اند.

برای اطلاع بیشتر به فصل ***برنامه‌نویسی*** مثال ***تحلیل آماری نمرات دانشجویان*** مراجعه کنید.

نام تابع	**max()**	**min()**	**mean()**	**hist()**
خروجی تابع	عنصر ماکزیمم	عنصر می‌نیمم	مقدار میانگین عناصر	پیشینه نگار

نام تابع	std()	sort()	sum()
خروجی تابع	مقدار انحراف میانه عناصر	مرتب‌سازی عناصر	مجموع عناصر

توابع زمانی

جدول زیر بعضی از توابع زمانی را نشان می‌دهد:

دستور	شرح
`tic`	شروع تایمر
`toc`	نشان دهنده زمان تایمر
`clock`	روز و ساعت
`etime(t1,t2)`	فاصله بین t1 و t2
`cputime`	زمان CPU بعد از شروع متلب
`date`	تاریخ روز
`calendar`	تقویم ماه جاری
`calendar(yyyy,mm)`	تقویم سال و ماه معین

مثال ها:

فاصله زمانی بین نوشتن **tic** و **toc**

```
>> tic
>> toc
elapsed_time = 2.0330
```

نمایش تاریخ، روز و ساعت

```
>> fix(clock)
```

برای نمایش واضح‌تر از تابع `fix()` استفاده کرده‌ایم. %

```
ans = 2002          5          26          11          33           8
```

نمایش تاریخ کامپیوتر

```
>> date
ans = 26-May-2002
```

توابع خاص

نمونه‌هائی از توابع پیشرفته ریاضی در زیر آمده‌اند:

تابع لژاندر	تابع بسل	تابع گاما $\Gamma(x)=\int_0^\infty t^{x-1}e^{-t}dt$	تابع بتا $\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$
`legendre(n,x)`	`bessel(n,x)`	`gamma(x)`	`beta(x)`

برای کسب اطلاع بیشتر در مورد توابع خاص از `help` مربوط به آن تابع استفاده کنید.

توابع کاربر- تعریف

توابع کتاب‌خانه‌ای متلب با تمام گستردگی تمام نیازها را برآورده نمی‌کنند، لذا کاربران می‌توانند توابع مورد نیاز و خاص خود را در فایل‌هائی به نام `Function M-File` تعریف و مشابه با توابع کتاب‌خانه‌ای از آن‌ها استفاده کنند. شرح بیشتر در فصول بعد می‌آید.

بعضی از توابع ریاضی کتاب خانه ای

نام	تابع
قدر مطلق	`abs(x)`
ریشه‌ی دوم	`sqrt(x)`
e^x	`exp(x)`
لگاریم طبیعی	`log(x)`
$\log_{10}x$	`log10(x)`
توابع مثلثاتی	`sin(x), cos(x), tan(x),cot(x)`
توابع مثلثاتی معکوس	`asin(y), acos(y)` `atan(y)` $[-\pi/2, \pi/2]$ * `atan2(y,x)` $[-\pi, \pi]$ *
گرد کننده به نزدیک‌ترین عدد صحیح به x	`round(x)` **
گرد کننده به نزدیک‌ترین عدد صحیح به x در جهت صفر	`fix(x)` **
گرد کننده به نزدیک‌ترین عدد صحیح به x در جهت $+\infty$	`ceil(x)` **
گرد کننده به نزدیک‌ترین عدد صحیح به x در جهت $-\infty$	`floor(x)` **
باقی‌مانده تقسیم، با علامت موافق x	`rem(x,y)` ***
باقی‌مانده تقسیم، با علامت موافق y	`mod(x,y)` ***
بزرگ‌ترین مقسوم علیه مشترک	`gcd(x,y)`
کوچک‌ترین مضرب مشترک	`lcm(x,y)`
نزدیک‌ترین کسر متعارفی به x به صورت t/n	`[t,n] = rat(x)`
نمایش x به صورت مجموعی از عددصحیح و کسر متعارفی	`rat(x)`

*، **، *** این توابع عمل‌کردهای متفاوت اما نزدیک به‌هم دارند، برای اطلاع بیش‌تر به help هریک مراجعه کنید.

۲-۴ اعداد و متغیرهای مختلط

در متلب اعداد مختلط وجود دارند و ریشه دوم 1- با حرف i نشان داده می‌شود. حرف j هم به همان معنا است.

مثال:

```
>> sqrt(-1), i, j
ans = 0 + 1.0000i
ans = 0 + 1.0000i
ans = 0 + 1.0000i
```

توابع مربوط به متغیرهای مختلط

توابع مختلط را نسبت به متغیر z = a + jb اعمال می‌کنیم. نتایج حاصله در جدول زیر آمده است.

فرم نوشتاری عدد مختلط

عدد مختلط z را می‌توان به‌یکی از اشکال مشروح در جدول زیر نوشت. وقتی عدد مختلط به عنوان یک عضو در ماتریس قرار می‌گیرد، باید بین عددها، علائم، و حرفِ i فاصله نباشد.

تابع	شرح	مقدار
real(z)	قسمت حقیقی را برمی‌گرداند	a
imag(z)	قسمت موهومی را برمی‌گرداند	b
abs(z)	قدر مطلق |z| را برمی‌گرداند	$\sqrt{a^2+b^2}$
angle(z)	زاویه را برمی‌گرداند	atan2(b,a)
conj(z)	مزدوج z را برمی‌گرداند	a - jb

توابع مربوط به متغیرهای مختلط

فرم نوشتاری	توضیح
z = a + jb	باید j به b چسبیده باشد.
z = a + j*b	
z = a + ib	باید i به b چسبیده باشد.
z = a + i*b	
z = mg * exp(j*ang)	mg = abs(z) ang = angle(z)

فرم‌های نوشتاری عدد مختلط

مثال:

ماتریس مختلط px را با قسمت‌های حقیقی و مجازی im,re به دو فرم قائم و قطبی بنویسید. مختصات قطبی آن‌را mag,tet بگیرید.

```
>> re = [1 2 3]; im = [-4 -5 -6];
>> px = re + j*im
px = 1.0000 - 4.0000i   2.0000 - 5.0000i   3.0000 - 6.0000i
>> tet = angle(px), mag = abs(px)
tet = -1.3258   -1.1903   -1.1071
mag =  4.1231    5.3852    6.7082
>> px = mag .* exp(j*tet)
px = 1.0000 - 4.0000i   2.0000 - 5.0000i   3.0000 - 6.0000i
```

۲-۵ تمرین

۱- با فرمت‌های مختلف مقدار `rmn ± eps` را نمایش دهید. `rmn = 2.22044604925031e-6`.

۲- با `format long` مقدار `m = [1e6 -1e7 1]` را با نمایش دهید. `scale factor` را ملاحظه کنید. سپس با `format long e` تکرار کنید.

۳- عدد پی را با انواع مضاعف، محدود، و صحیح در حافظه نگه‌داری کنید، سپس با فرمت‌های مختلف نشان دهید. نتایج را با `isa()` امتحان و در پنجره فضای کار مشاهده کنید.

٤- با نحوه نمایش `format long`، و با تکرار تعیین کنید n چه عدد صحیحی باشد که رقم یک در اعداد بعداز ممیز حاصلِ عبارت `1 + n*eps` ظاهر شود. به این شکل: `1.00000000000001`

۵- متغیر `x = 42235.62567889404296875` را با `single(x)` به تابع یکا تبدیل و با فرمت بلند، کوتاه و فرمت بانک `format bank` نمایش دهید. متغیر x را تقسیم بر صفر کرده برابر y قرار دهید.

٦- عملیات آرایه‌ای را در مورد بردارهای `a = [-2.1 4 8]` و `b = [3 0 2]` امتحان کنید. بردارها را ترانهاد (ترانسپوز) و عملیات را تکرار کنید.

۷- جدول کسینوس زوایای بین `360,+360-` درجه را تولید کنید و نمایش دهید.

۸- یک ماتریس تصادفی 5×4 تولید کنید.

۹- یک بردار عددی با 14 عنصر تصادفی تولید کنید، که بزرگای هر عضو آن بین 10 تا 100 باشد.

۱۰- با `repmat()` یک بردار با ده‌هزار صفر تا نُه تولید کنید.

۱۱- فاصله زمانی بین نوشتن `tic` و `toc` را در کامپیوتر خودتان تعیین کنید.

۱۲- با استفاده از `clock` فقط تاریخ را به فرم مناسب نمایش دهید.

۱۳- تابع `calendar(2002,01)` را امتحان کنید.

۱٤- متغیر t را مساوی صفر تعریف کرده `f = sin(t)/t` را به دست آورید.

۱۵- توابع `realmax` و `realmin` را امتحان کنید. عبارات `type realmax` و `type realmin` را مجدداً اجرا و بررسی کنید.

۱٦- عدد گنگِ e را با فرمول `e = exp(1)` (طرف راست، e^1 است) به دست آورده با هر دو فرمت کوتاه و بلند و فرمت کسری `format rational` نمایش دهید. **راهنما:** e یک متغیر داخلی نیست.

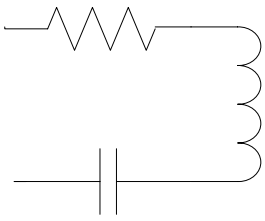
۱۷- توابع مثلثاتی زاویه ۳۰ درجه را پیدا کرده در یک ماتریس نمایش دهید.

۱۸- توابع `round()` و `fix()` و `ceil()` و `floor()` را برای 13/7 و 13/7- به دست آورید.

۱۹- متغیر `x = 0.34` را به تنهائی به صورت یک کسر متعارفی و مخلوطی از عدد و کسر متعارفی نشان دهید.

۲۰- با استفاده از توابع مختلط مقدار امپدانس و ضریب توان مدار زیر را با مقادیر داده شده پیدا کنید.

`R = 1Ω, L = 1mH, C = 1μF, ω = 1000rad/s`



فصل ۳ ماتریس ها

۳-۱ تشابه مابین ماتریس ها

رابطه تشابه به یکی از اشکال زیر مابین ماتریس‌ها برقرار است. در این متن از اصطلاح **هم‌سان** استفاده زیادی می‌شود.

نام	تعریف	کاربرد
هم‌ردیف- هم‌ستون (هم‌سان)	چند ماتریس با ***تعداد*** ردیف‌ها و ستون‌های مساوی	ترسیمات، ...
هم ردیف- با ستون (ضرب‌پذیر)	دو ماتریس با ***تعداد*** ردیف‌های یکی مساوی ستون‌های دیگری	ضرب ماتریسی
مساوی (یکسان)	چند ماتریس با ***تعداد*** ردیف‌ها و ستون‌های مساوی و ***مقدار*** درایه‌های مساوی	عملیات ریاضی
متساوی‌العنصر	چند ماتریس با ***تعداد*** درایه‌های مساوی اما ***تعداد*** ردیف‌ها و ستون‌های نامساوی	عملیات ماتریسی

۳-۲ ماتریس های پایه Elementary Matrices

ماتریس‌هائی نظیر `eye()`, `rand()`, `ones()`, `zeros()` در متلب ماتریس پایه نام دارند. راهنمای آن‌ها و بعضی اطلاعات دیگر با اجرای `help elmat` روی پنجره فرمان نمایش داده می‌شوند.

ماتریس یگانی identity matrix

دستور `eye(n,m)` ماتریسی با قطری متشکل از یک‌ها و سایر عناصر مساوی با صفر ایجاد می‌کند. دستور `eye(n)` یک ماتریس یگانی مربعی $n \times n$ ایجاد می‌کند.

مثال:

```
>> x = eye(3)
x = 1     0     0
    0     1     0
    0     0     1
```

ماتریس های ones() و zeros()

دستور `ones(n,m)` ماتریس $n \times m$ با عناصر یک و دستور `zeros(n,m)` ماتریس $n \times m$ با عناصر صفر می‌سازند. دستورهای `ones(n)` و `zeroz(n)` ماتریس‌های مربعی $n \times n$ ایجاد می‌کنند.

مثال:

```
>> zeros(3,2)
ans = 0     0
      0     0
      0     0
>> zeros(2)
ans = 0     0
      0     0
>> ones(2)
ans = 1     1
      1     1
```

۳-۳ ایجاد تغییرات بر روی ماتریس

استخراج قسمتی از یک ماتریس

برای استخراج قسمتی از یک ماتریس ابتدا ردیف‌ها و سپس ستون‌های مستخرجه از آن‌ردیف‌ها را می‌نویسیم. مثلاً عبارت A(1:2,2:3) نشان دهنده ردیف 1 تا ردیف 2 محدود بین ستون 2 تا ستون3 است.

مثال ها:

استخراج یک ردیف از ماتریس

```
>> A = [1 2 3; 4 5 6];
>> A1 = A(1,:) % ردیف ۱ را استخراج می‌کند
```

```
A1 = 1     2     3
```

استخراج یک ستون از ماتریس

```
>> A = [1 2 3; 4 5 6];
>> A2 = A(:,2) % ستون ۲ را استخراج می‌کند
```

```
A2 = 2
     5
    -2
```

استخراج قسمتی از ماتریس

```
>> A = [1 2 3; 4 5 6];
>> A23 = A(1:2,2:3) % ستون‌های ۲ و ۳ از ردیف‌های ۱ و ۲ را استخراج می‌کند
```

```
A23 = 2     3
      5     6
```

تغییر مقدار عناصر ماتریس

مثال:

ماتریس $b=\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}$ را تولید و با آن ماتریس‌های $b1=\begin{bmatrix}1&0&0\\2&0&0\\3&6&9\end{bmatrix}$ و $b2=\begin{bmatrix}1&1&1\\2&1&1\\3&6&9\end{bmatrix}$ را بسازید.

ابتدا ماتریس b را تعریف می‌کنیم. با استفاده از ترانهاد b را راحت‌تر تعریف کرده‌ایم.

```
>> b = [1:3;4:6;7:9]'
```

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

ماتریس b1 را مساوی b تولید می‌کنیم:

```
>> b1 = b;
>> b1(1:2,2:3) = zeros(2)
```

```
b1 =  1.00        0        0
      2.00        0        0
      3.00     6.00     9.00
```

ماتریس b2 را مساوی b تولید می‌کنیم:

```
>> b2 = b;
>> b2(1:2,2:3) = ones(2)
```

```
b2 =  1.00     1.00     1.00
      2.00     1.00     1.00
      3.00     6.00     9.00
```

قرار دادن یک ماتریس در ماتریس دیگر

عبارت `d(:) = b` عناصر b را به ترتیب ستون در داخل d جاسازی می‌کند.

مثال:

دو ماتریس d و b متساوی‌العنصر اما ناهم‌سان هستند. عناصر b را داخل d قرار دهید.

```
>> d = zeros(3,2)
d =  0     0
     0     0
     0     0
>> b = [1:3; 4:6]
b =  1     2     3
     4     5     6
>> d(:) = b
d =  1     5
     4     3
     2     6
```

استخراج یا حذف ردیف و ستون

عبارت `a(:,n)` ستون n، و عبارت `a(m,:)` ردیف m را برمی‌گردانند.

مثال:

ستون دوم و ردیف اول ماتریس `b = [1:3;4:6;7:9]'` را حذف کنید.

```
>> b = [1:3;4:6;7:9]'
b =  1     4     7
     2     5     8
     3     6     9
>> b(:,2)
ans = 4
      5
      6
>> b(1,:)
ans = 1     4     7
>> b(:,2)=[]
b =  1     7
     2     8
     3     9
>> b(1,:)=[]
b =  2     8
     3     9
```

۳-۴ عمل گرهای ماتریسی

علائم ضرب، تقسیم، و توان، بدون نقطه در سمت چپ، عملگرهای ماتریسی هستند و به صورت عنصر به عنصر عمل نمی‌کنند. جمع و تفریق آرایه‌ای و ماتریسی یک‌سان هستند. این عملگرها در جدول زیر آمده‌اند.

توان	ضرب	تقسیم راست به چپ	جمع	تقسیم (معمولی)	تفریق
^	*	\	+	/	-

ضرب ماتریسی

ضرب ماتریسی پرکاربردترین عمل ماتریسی است. اگر c حاصل‌ضرب a در b باشد، عناصر ردیف یک a در عناصر ستون یک b به ترتیب ضرب و با هم جمع می‌شوند، نتیجه عنصر `c(1,1)` است. عناصر ردیف یک a در عناصر

ستون دو b به ترتیب ضرب و با هم جمع می‌شوند، نتیجه عنصر c(1,2) است. و به همین ترتیب ماتریس c به دست می‌آید. حاصل‌ضرب دو ماتریس در صورتی بدون خطا به دست می‌آید که ستون‌های a با سطرهای b برابر باشند. اگر بخواهیم a * b و b * a هر دو معنی‌دار باشند، علاوه بر این‌که باید ستون‌های a با سطرهای b برابر باشند، دو ماتریس باید متساوی‌العنصر هم باشند، وگرنه با پیغام خطا مواجه خواهیم شد. اگر a * b و b * a هر دو معنی‌دار باشند از تعریف ضرب پیداست که a * b مساوی b * a نخواهد بود.

مثال ها:

حاصل اسکالر

```
>> am = [2 4 8];
>> bm = [3; 2; 2];
>> cm = am * bm
cm = 30
```

حاصل ماتریسی

```
>> a = [1 2 -6; 3 0 -3]
a =  1     2    -6
     3     0    -3
>> b = [2 5 6; 0 1 4; 2 6 -8]
b = 2     5     6
    2     6    -8
    0     1     4
>> c = a * b
c = -10   -29    62
      0    -3    42
```

خطا در نتیجه جابجائی عوامل ضرب

```
>> ci = b * a
??? Error using ==> *
Inner matrix dimensions must agree.
```

دو جواب مختلف در نتیجه جابجائی عوامل ضرب

```
>> a = [1 2 -6; 3 0 -3]
a =  1     2    -6
     3     0    -3
>> b = [2 5 ; 0 1 ; 2 6]
b =  2     5
     0     1
     2     6
>> a * b
ans = -10   -29
       0    -3
>> b * a
ans = 17     4   -27
       3     0    -3
      20     4   -30:
```

توان ماتریسی

a ^ 2 که معادل a * a است فقط برای ماتریس مربعی معنی دارد. a ^ 2 با a .^ 2 تفاوت اساسی دارد.

۳-۵ بعضی از توابع ماتریسی

چند تابع معمول

نام	تابع	نتیجه
دترمینان determinant	det(a)	دترمینان ماتریس a
توان نپری ماتریسی matrix exponent	expm(a)	e^a
ماتریس معکوس inverse matrix	inv(a)	معکوس ماتریس a

استخراج ماتریس از ماتریس دیگر

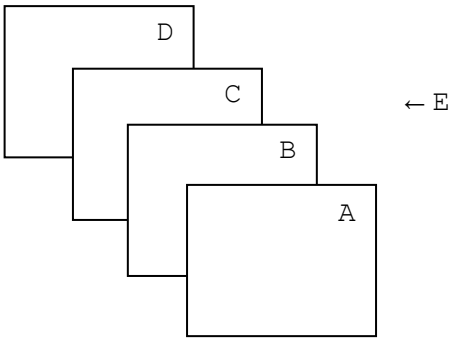
نام	تابع	نتیجه
diagonal	diag()	قطر ماتریس را استخراج می‌کند
lower triangle	tril()	مثلث پائین ماتریس را استخراج می‌کند
upper triangle	triu()	مثلث بالای ماتریس را استخراج می‌کند

توابعی که به روی مجموعه عناصر ماتریس عمل می‌کنند

بعضی از توابع مانند diff(),prod(),sum(),cumpro() به روی مجموعه عناصر ماتریس عمل می‌کنند. برای اطلاع بیشتر به help هر تابع مراجعه شود.

ماتریس سه بعدی (فضائی)

یک ماتریس سه بعدیِ $m \times n \times p$ دارای p صفحه است که هر صفحه یک ماتریس دو بعدی $m \times n$ است. انتخاب مقادیر m, n, p دلخواه است، اما باید دقت کرد که n و m برای تمامی صفحات یکسان باشند. در شکل زیر شمای فضائی ماتریس E نشان داده شده که در آن p = 4 است:



صفحات ۱، ۲، ۳ و ٤ ماتریس فضائی E به این صورت تعریف می‌شوند:

```
E(:,:,1) = A
E(:,:,2) = B
E(:,:,3) = C
E(:,:,4) = D
```

A و B و C و D ماتریس‌های دو بعدی هستند که قبلاً تعریف شده‌اند.

در زبان متلب تعریف صفحه‌ی nام از یک ماتریس سه بعدی به نام E به‌صورت E(:,:,n) = M است. صفحه-های تشکیل دهنده یک ماتریس فضائی باید همسان (هم‌ردیف و هم‌ستون) باشند. size(E) تعداد ردیف، ستون، و صفحه‌های ماتریس فضائی را نشان می‌دهد. عبارت E(:) یک بردار تک‌ستونی متشکل از تمام عناصر ماتریس فضائی ایجاد می‌کند. عبارت E(:,:) یک ماتریس دو بعدی متشکل از ماتریس‌های سازنده‌ی E ایجاد می‌کند.

مثال:

یک ماتریس فضائی با چهار صفحه تولید کنید. تعداد ردیف و ستونِ هر صفحه، و تعداد صفحه‌های آن‌را با تابع داخلی size() نمایش دهید. بُعد آن را با استفاده از تابع داخلی ndims() تعیین کنید.

```
>> A = [2 -5; 16 7.6; 13.3 -9];
>> B = [-2 -5.2; 6 7.3; 5 1.4 ];
>> C = [-2 2 ; 7 3; 1.55 1.4 ];
>> D = [-2.2 12 ; 7.3 30; 5.5 1.4 ];

>> E(:,:,1) = A;
>> E(:,:,2) = B;
>> E(:,:,3) = C;
>> E(:,:,4) = D
```

```
E(:,:,1) =  2.0000   -5.0000
           16.0000    7.6000
           13.3000   -9.0000
E(:,:,2) = -2.0000   -5.2000
            6.0000    7.3000
            5.0000    1.4000
E(:,:,3) = -2.0000    2.0000
            7.0000    3.0000
            1.5500    1.4000
E(:,:,4) = -2.2000   12.0000
            7.3000   30.0000
            5.5000    1.4000
```

```
>> size(E), ndims(E), length(size(E))
ans =  3     2     4    ans =  3      ans = 3
```

۳-۶ ماتریس های نمونه

ماتریس جادوئی

دستور magic(n) یک مربع جادوئی n*n می‌سازد. خاصیت مربع جادوئی این است که حاصل‌جمع عناصر آن در طولِ ردیف، قطر، و ستون برابرند.

مثال:

یک مربع جادوئی ۳×۳ به نام Mg بسازید. با تکرار Mg یک ماتریس۳×۶ به نام Mgr بسازید.

```
>> Mg = magic(3)
Mg = 8     1     6
     3     5     7
     4     9     2
>> Mgr = repmat(Mg,2,1)
Mgr = 8     1     6
      3     5     7
      4     9     2
      8     1     6
      3     5     7
      4     9     2
```

ماتریس Mgr یک مربع جادوئی نیست

ماتریس پاسکال

ماتریسی است که از مثلث پاسکال تشکیل می‌شود. یک نمونه مثلث پاسکال در زیر آمده است:

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5    1
1   6   15  20  15   6   1
```

در مثلث پاسکال ستون‌ها ضرائب بینم binomial coefficient هستند، که از رابطه زیر به دست می‌آیند:

$$\binom{n}{r} = \frac{n!}{r!(n-r)!} = \binom{n-1}{r} + \binom{n-1}{r-1}$$ که خلاصه‌تر آن این است $$\binom{n}{r} = \frac{n(n-1)(n-2)\ldots(n-r+1)}{r!}$$

نتیجه این رابطه تعداد چینش r شیئ از میان n شیئ متمایز است.

با برنامه زیر می‌توان ضرائب بینم را محاسبه کرد. برای اطلاع در موردحلقه for به فصل ساختارهای تصمیم و تکرار مراجعه فرمائید.

```
ncr = 1;
r = ...;
n = ...;
for k = 1:r
 ncr = ncr*(n-k+1)/k;
end
disp(ncr)
```

مثلاً اگر یک بار r را مساوی ۲ و یک‌بار آن را مساوی ۳ بگیریم، و هر بار n را بین ۲ تا ٦ تغییر دهیم ستون‌های سوم و چهارم مثلث به دست می‌آیند. مثلث پاسکال اول بار توسط حکیم عمر خیام کشف شد و دارای خواصی است که در ریاضیات هنوز هم مورد تحقیق هستند، از جمله این که هر عنصر آن از حاصل جمع قطر بالای سرش به دست می‌آید.

مثال:

یک ماتریس پاسکال ٤×٤ تولید کنید.

```
>> ps = pascal(4)
ps = 1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```

۳-۷ بردار Vector

عملگر کالن

عملگر کالن در متلب برای تعیین دامنه range و گام step بردار به کار می‌رود.

مثال:

```
>> x = 1:5
x = 1   2   3   4   5
>> x = -1: 0.5: 1
x = -1.0000   -0.5000    0    0.5000    1.0000
```

تابع linspace()

تابع linspace(m,n,p) تعداد p عنصر بین اعداد m و n تولید می‌کند. پیش‌فرض p یک‌صد عنصر است.

مثال:

ده عدد بین صفر و pi را داخل بردار x قرار دهید. سپس x را به درجه تبدیل کنید.

```
>> format bank
>> x = linspace(0,pi,10)
x = 0  0.35  0.70  1.05  1.40   1.75   2.09  2.44  2.79  3.14
>> dgx = x * 180/pi
dgx = 0 20.00 40.00 60.00 80.00 100.00 120.00 140.00 160.00 180.00
```

کاربرد یک بردار در تعریف بردار دیگر

مثال:

```
>> a = [2 3 -4.5];
>> b = [-2.2 3 0.5];
>> c = [a b]
c = 2.00    3.00   -4.50   -2.20    3.00    0.50
```

بردار تهی

عبارت x = [] بردار تهی را به x نسبت می‌دهد. این تعریف با x = 0 یا پاک کردن x (clear x) فرق دارد.

مثال:

برای حذف عنصر 2 آن را برابر بردار تهی قرار می‌دهیم (توضیح بیشتر راجع به شماره عناصر بعداً می‌آید).

```
>> c = [2    3   -4.5   -2.2    3    0.5];
>> c(2) = []
c = -2.0000    4.5000    2.2000   -3.0000   -0.5000
```

اندیس اعضاء بردار

هر عنصر از بردار جای‌گاه یا اندیسی دارد که با آن شناخته می‌شود. در متلب شماره جای‌گاه با 1 شروع می‌شود، لذا مراجعه به یک عضو بردار بسیار آسان‌تر از زبانی مانند C++ است که در آن اندیس از صفر شروع می‌شود مثال زیر روش‌های مراجعه به عناصر بردار را توضیح می‌دهد.

مثال ها:

مراجعه به عناصر بردار با چند روش‌ها

```
>> x = 0:3:23
x = 0     3     6     9    12    15    18    21
>> x(1), x(3)
ans = 0
ans = 6
>> x(2:4)
ans = 3     6     9
>> x(1:2:8)
ans = 0     6    12    18
>> x([1 5 8])
ans = 0    12    21
```

حذف عناصر اول، دوم، و هفتم بردار

```
>> x = 1:7
x = 1     2     3     4     5     6     7
>> x([1 2 7]) = []
x = 3     4     5     6
```

۳-۸ بردارهای منطقی

عناصر بردار منطقی، مجموعه‌ای از صفر و یک‌های منطقی هستند. صفر و یک منطقی از لحاظ نوع با صفر و یک عددی فرق دارند. مثال‌های زیر این مورد را توضیح می‌دهند. برای اطلاع بیشتر `help logical` را اجرا کنید.

تبدیل بردار عددی به منطقی با تابع logical()

تابع `logical()`، یک بردار با اعضاء عددی را به یک بردار با اعضاء صفر و یکِ منطقی تبدیل می‌کند. متغیر منطقی فقط دو مقدار درستی (منطق یک) و نادرستی (منطق صفر) را می‌گیرد، که از لحاظ نوع `type` با 0 و 1 عددی متفاوت هستند. مقادیر منطقی یک بایت از حافظه را اشغال می‌کنند، در حالی‌که مقادیر عددی (از نوع `double` که پیش‌فرض متلب است) هشت بایت جا می‌گیرند. اگرچه آرایه‌های عددی با اعضاء غیر از صفر و یک را می‌توان به آرایه منطقی تبدیل کرد، اما توصیه می‌شود فقط آرایه‌هائی با اعضاء صفر و یکِ عددی به آرایه منطقی تبدیل شوند. منطقی بودن یک آرایه با تابع `islogical()` امتحان می‌شود.

توابعی که صحّت یا سُقمِ امری را امتحان می‌کنند، در صورت صحت، منطق یک و در صورت کذب، منطق صفر برمی‌گردانند. معمولاً این گونه توابع با `is...` شروع می‌شوند.

مثال ها:

ساختن بردار منطقی

```
>> oz = [1    0    1     1     0    0    0    1];
>> islogical(oz)
ans = 0
```

مقادیر بردار oz یک وصفر عددی هستند.

```
>> ozL = logical(oz)
ozL = 1   0   1   1   0   8   0   0   1
>> islogical(ozL)
ans =  1
```

اگرچه مقادیر دو بردار oz و ozL شبیه هستند اما نوع آن‌ها متفاوت است. Workspace را ببینید.

مشاهده در پنجره فضای کار **Workspace**

```
>> ad = 1
ad = 1
>> ag = logical(1)
ag = 1
```

Workspace

Name	Size	Bytes	Class
ad	1x1	8	double array
ag	1x1	1	logical array

شکل ۳-۱

حذف بعضی عناصر آرایه

اگر بردار منطقی را اندیس بردار عددی قرار دهیم (اندیس‌گذاری منطقی `Logical Indexing`) فقط عناصر متناظر با یک‌های منطقی باقی می‌مانند. عناصر بردار اندیس می‌تواند از عناصر بردار عددی کم‌تر باشد، اما بیش‌تر نمی‌تواند باشد.

مثال ها:

حذف چند عنصر

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 0 1 1 1 0 1]);
>> LiEv = Li(Lg) % Logical Indexing
LiEv = 2     4     6     8    10    12
```

نگه داری چند عنصر

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 1]);
>> LiEv = Li(Lg)
LiEv = 2     4     5
```

ضرب بردار عددی در بردار منطقی

برای ضرب عنصر به عنصر، علامت ضرب آرایه‌ای `.*` به کار می‌رود، و دو آرایه باید همسان باشند. نتیجه ضرب منطق در عدد از نوع عددی است..

مثال ها:

مشابه ضرب عدد در عدد

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 0 1 1 1 0 1]);
>> Lip = Li .* Lg
   Lip =  2     4     0     6     8    10     0    12
>> islogical(Lip)
   ans = 0
>> isnumeric(Lip)
   ans = 1
```

استخراج بردار از بردار دیگر

```
>> r = 1 : 5 ;
>> rL = (r <= 3)
rL = 1     1     1     0     0
>> s = r .* rL
s = 1     2     3     0     0
```

یافتن محل عناصری با مقدار معین

مثال:

محل عناصر مساوی با 9 بردار `ab = [2 0 9 5 0 1.5 -6 9 0 -4.35]` را تعیین کنید. توجه کنید که دو علامت مساوی پیوسته `(==)` تساوی دو بردار را تست می‌کند.

```
>> ab = [2 0 9 5 0 1.5 -6 9 0 -4.35];
>> ab == 9
ans = 0     0     1     0     0     0     0     1     0     0
```

عناصر مساوی با 9 بردار ab در مکان‌های سوم و هشتم قرار دارند.

یافتن اندیس عناصر مورد نظر با تابع find()

تابع `find()` اندیس عناصر مورد نظر را در یک بردار دیگر قرار داده و برمی‌گرداند، و آرگومان آن باید یک بردار منطقی باشد. عبارت `find(a)` با `find(a ~= 0)` معادل است.

مثال ها:

یافتن اندیس عناصر مساوی با **9**

```
>> ab = [2 0 9 5 0 1.5 -6 9 0 -4.35];
>> find(ab == 9)
ans = 3 8
>> ab(3),ab(8)
ans = 9
ans = 9
```

راه حل دیگر:

```
>> k = (ab == 9)
k = 0   0   1   0   0   0   0   1   0   0
find(k)
ans = 3 8
>> ab(3),ab(8)
ans = 9
ans = 9
```

k یک بردار منطقی است که در ازای عناصر مساوی 9 بردارِ ab دارای منطق یک (درستی) است.

یافتن اندیس عناصر: غیرصفر، منفی، و ناموجود

```
>> ab = [2 0 9 5 0 1.5 -6 9 0 -4.35];
>> af1 = find(ab)
af1 = 1     3     4     6     7     8    10
>> af2 = find(ab < 0)
af2 = 7          10
find(ab == 8) % Not found
ans = Empty matrix: 1-by-0
```

۳-۹ تمرین

۱- عبارات `x = []` و `x = 0` و `clear x` را اجرا کنید و نتایج آن‌ها را در پنجره فضای کار ببینید.

۲- عنصر شماره ۵ از بردار `c = 1:10` را حذف کنید.

۳- عناصر شمار ۱ و ۲ و ۸ را از بردار `x = 10:-1:1` حذف کنید.

٤- نمره‌های ترم خود را داخل یک بردار قرار دهید و با یک دستور معدل را حساب کنید.

۵- خریدهای ماه خود را داخل یک بردار قرار دهید و با یک دستور جمع آن‌ها را حساب کنید.

٦- یک بردار عددی و یک بردار منطقی بسازید. بردار منطقی را اندیس Rv قرار دهید. در Rv ضرب کنید.

۷- یک ماتریس عددی ۳×٦ با مربع جادوئی ۳×۳ بسازید، توابع اندازه‌گیر را در مورد آن اجرا کنید.

۸- یک بردار عددیِ ستونی به نام Rh با ۵ عنصر تعریف کنید. بردار `y = Rh.^2` را به دست آورید.

۹- ماتریس مختلط px را با قسمت‌های حقیقی و مجازی `im,re` به دو فرم قائم و قطبی بنویسید. مختصات قطبی آن‌را `mag,tet` بگیرید.

```
re = [-1 2 3.5]; im = [-4.5 5 6.2];
```

۱۰- یک ماتریس تصادفی سه صفحه‌ای با صفحات 4×3 تولید کنید. از صفحه‌های سازنده‌ی ماتریس فضائی فوق: یک ماتریس دوبعدی، و یک بردار تک ستونی تولید کنید. توابع اندازه‌گیر را در مورد این ماتریس‌ها اجرا و نتایج را بررسی کنید.

۱۱- ماتریس B را از ستون ۲ و ستون ٤ ماتریس A استخراج کنید. ماتریس C را از ردیف ۲ و ردیف ۳ ماتریس A استخراج کنید.

```
A = [1 2 3 -8; 4 -5 0 9; -1 -2 3 0;]
```

۱۳- بردار `a = 0:4` را دو بار تکرار کنید. با استفاده از یک بردار منطقی صفرهای a را حذف کنید.

۱٤- این بردار عددی را درنظر بگیرید: `rv = [2 1 0 7 4 9 4 4 8 5]` با استفاده از بردار منطقی عناصر کوچک‌تر از یا مساوی با ٤ بردار rv را در nv قرار دهید.

۱۵- با توجه به نتایج زیر مقدار بردار a را حدس بزنید:

```
>> a4 = a < 4
a4 = 1     1     1     1     0     1     1     1     1     0
>> a(a4)
ans = 0     1     2     3     0     1     2     3
```

۱٦- با توجه به نتیجه زیر بردار ab را با ۱۰ عنصر بنویسید، آیا af یک بردار منطقی است؟

```
>> af = find(ab)
af = 1     3     4     6     7     8    10
```

۱۷- با داشتن `Vk = magic(3)` نتایج حاصل از اجرای دستورات زیر را به دست آورید.

```
>> b6 = (Vk < 7), b = Vk(b6),  b'
>> bn = Vk(1:1,1:3)
>> b1 = repmat(bn,1,3), b2 = repmat(bn,3,1), b3 = repmat(bn,3,3)
>> diag(Vk), tril(Vk), triu(Vk) inv(Vk), expm(Vk), det(Vk)
```

۱۸- ماتریس پاسکال `ps = pascal(5)` را با مثلث پاسکال مقایسه کنید.

فصل ۴ دستورها و توابع ورودی خروجی

۴-۱ دریافت ورودی

معمولاً در یافت متغیرها از صفحه کلید از داخل برنامه انجام می‌شود. برای اطلاع بیشتر در مورد برنامه نویسی به فصل مربوطه مراجعه کنید.

دریافت با input()

این دستور مقدار یک متغیر (آرایه) را از صفحه کلید دریافت می‌کند. پیغام لازم را نیز می‌توان در آن گنجاند.

در جواب input () می‌توان یک ماتریس، یک تابع داخلیِ متلب (مثل: rand ())، یا یک محاسبه (مثل: a+b)، را وارد کرد به شرطی که مقدار متغیرهای موجود در آن‌ها از قبل معین باشد.

مثال ها:

دریافت بردار عددی

```
>> a = input('Enter a number vector: ')
```

```
Enter a number: [2 2.5 3 3.5]
a = 2 2.5 3 3.5
```

ورود ترکیبی از یک عدد و یک متغیر از پیش تعریف شده (a مثال فوق)

```
>> b = input('Enter another number using a: ')
```

```
   Enter another number using a: 3 + a(1)
   b = 5
```

ورود یکی از توابع توساخت متلب

```
>> d = input('Enter a function using previous variables: ')
```

```
   Enter a function using previous variables: rand(a,b)
   d = 0.9501    0.6068    0.8913    0.4565    0.8214
       0.2311    0.4860    0.7621    0.0185    0.4447
```

دریافت رشته

```
>> nam = input('Enter the student name: ','s')
```

قرار دادن 's' به عنوان آرگومان دوم input () باعث می‌شود که بتوانیم یک رشته را به wr نسبت دهیم %

```
Enter the student name: Mostafa
nam = Mostafa
```

دریافت با keyboard

دستور keyboard ما را آزاد می‌گذارد که هر تعداد متغیر از هر نوع را مقدار دهی کنیم، برای مقدار دهی باید نام متغیر و علامت تساوی را در مقابل نشانه‌ی K>> بنویسیم. برای خروج از محیط keyboard باید <Ctrl+C> اجرا شود.

اجرای این دستور از داخل ***برنامه*** برای دریافت مقادیر متغیرها مفید است.

مثال:

```
>> disp('Enter A, B, C'), keyboard
```

```
Enter A, B, C
K>> A=-1:3; B=[9 -4.5]; C = 'new';
```

۴-۲ ارسال خروجی

ارسال خروجی به صفحه نمایش، دستور disp(var)

آرگومان disp () یک آرایه است، و اعضاء آن باید از یک نوع باشند. اگر آرگومان یک بردار سلولی باشد. هر عضو جداگانه نشان داده خواهد شد.

مثال‌ها:

متغیرهای ناهم‌جنس

```
>> x = 6.5;  mv = 'MATLAB Version is ';
>> disp([mv num2str(x)]);
MATLAB Version is 6.5
```

تابع num2str(x) عدد را به رشته تبدیل می‌کند.

متغیرهای هم‌جنس

```
>> v = version;
>> disp([mv v]);
MATLAB Version is 6.5.0.180913a (R13)
```

version یک متغیرِ رشته‌ایِ داخلی متلب است، که ویراست آن را نگه‌داری می‌کند.

آرایه سلولی

```
>> C ={ 'MATLAB Version is ', 6.5};
>> disp(C);
    'MATLAB Version is '    [6.5000]
```

ارسال خروجی به صفحه نمایش، دستور fprintf()

این تابع کاملاً مشابه نظیرش در زبان C++ است و امکان ترکیب عدد و رشته و دادن فرم دل‌خواه به اعداد را در خروجی فراهم می‌کند. علائم: %s, %c, %G, %g, %E, %e, %f, %X, %x, %u, %o, %i, %d علائم تعیین فرمت خروجی Format Specifier هستند. علائم ..., \b, \t, \n دستورهای چاپی یا escape code هستند. فرمت %g (good) مشابه %f و با انتخاب فرم مناسب عمل می‌کند. علائم %s و %f به ترتیب فرمت چاپ رشته (string) و اعشاری (float) هستند. عبارت %7.2f هفت مکان برای چاپ معدل اختصاص داده و ارقام بعداز ممیز را به دو رقم گرد می‌کند. علامت \n (new line) ادامه چاپ را به سطر بعد منتقل می‌کند. چینش پیش‌فرض خروجی راست‌چین است، علامت منفی بعد از % خروجی را چپ‌چین می‌کند.

کاربرد بعضی از این علائم در مثال‌ها آمده‌اند. برای اطلاع بیشتر به help fprintf مراجعه نمائید.

مثال ها:

گرد کردن تا سه رقم بعد از نقطه ممیز، راست‌چین کردن داخل کرسی

```
>> av = [17.4537 4.57 15.3 17.869 3.7];
>> nam = 'Students Average:';
>> fprintf('%s\n', nam);fprintf('%7.3f\n', av);
Students Average:
 17.454
  4.570
 15.300
 17.869
  3.700
```

چپ‌چین کردن داخل کرسی

```
>> fprintf('%s\n', nam);fprintf('%-7.3f\n', av);

Students Average:
17.454
4.570
15.300
17.869
3.700
```

دستور echo off/on

`echo off` نمایش دستورات برنامه بر روی پنجره را متوقف و فقط نتایج را نمایش می‌دهد. این دستور فقط از داخل برنامه قابلیت اجرائی دارد و نباید مستقیماً از پنجره فرمان اجرا شود. برای اطلاع بیشتر به مثال‌های برنامه‌نویسی مراجعه کنید. `echo on` برعکس عمل می‌کند. برای طلاع بیشتر `help echo` را جرا کنید.

دستور pause

این دستور اجرا را درانتظار عمل کاربر متوقف می‌کند. `pause(n)` به اندازه n ثانیه می‌ایستد و سپس ادامه می‌دهد.

۴-۳ ضبط بر روی دیسک

ارسال خروجی به فایل متن TXT.

می‌توان آن‌چه را که برروی صفحه نمایش می‌آید با دستور `fprintf()` در یک فایل متن ضبط کرد.

نام این فایل بر روی دیسک اختیاری است و معمولاً با پسوند `.TXT` انتخاب می‌شود.

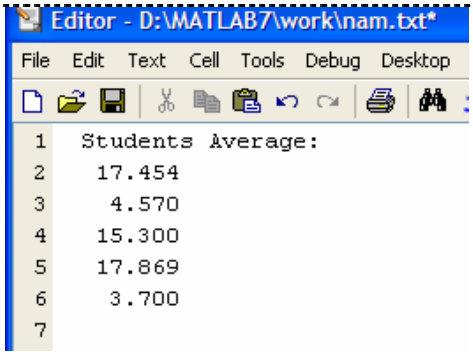
هنگام باز شدن فایل یک نام مستعار به آن اختصاص می‌یابد. نام مستعار یک عدد است که متلب در دستورات داخلی خود از آن استفاده می‌کند. در دستورات سیستمی نام دیسکی فایل می‌آید. یک مثالِ باز کردن فایل برای نوشتن چنین است:

```
myf = fopen('nam.txt','w');
  ↑      ↑         ↑      ↑
نام مستعار  دستور باز کردن  نام فایل  برای نوشتن
```

مثال:

یک فایل متن `.TXT` ایجاد می‌کنیم. آرایه نمرات و عنوان را در آن نوشته، فایل را مشاهده (شکل ٤-١) و سپس پاک می‌کنیم. نام مستعار فایل `myf`، و نام دیسکی فایل `nam.txt` است.

```
>> av = ...
[17.4537 4.57 15.3 17.869 3.7];
>> Tit = 'Students Average:';
>> myf = fopen('nam.txt','w');
>> fprintf(myf,'%s\n',Tit);
% استفاده از نام مستعار فایل
>> fprintf(myf,'%7.3f\n',av);
>> fclose(myf);
% این دستور را حتماً بنویسید
>> open nam.txt
% استفاده از نام دیسکی فایل در دستور سیستمی
>> delete nam.txt
```



شکل ٤-١

ضبط ماتریس در فایل متن TXT.

مثال:

برنامه‌ای می‌نویسیم که مقادیر `x = 0.1: 0.1: 1` را همراه `log10(x)` به صورت یک ماتریس در یک فایل متن نگه‌داری کند. فضای کار را پاک کرده اطلاعات فایل متن را وارد فضای کار کرده، و ماتریس را نمایش می‌دهیم. فایل متن را حذف می‌کنیم.

```
>> x = 0.1: 0.1: 1;
>> y = [x; log10(x)];
>> fl = fopen('lgt.txt','w');
% open to write
>> fprintf(fl,'%f  %f\n',y);
>> fclose(fl);
>> load lgt.txt
>> lgt
>> delete lgt.txt
>> tx
```

```
lgt = 0.1000   -1.0000
      0.2000   -0.6990
      0.3000   -0.5229
      0.4000   -0.3979
      0.5000   -0.3010
      0.6000   -0.2218
      0.7000   -0.1549
      0.8000   -0.0969
      0.9000   -0.0458
      1.0000         0
```

ضبط ماتریس در فایل باینری MAT.

دستور `save filename` کلیه متغیرهای فضای کار را در فایل باینری `filename.mat` ضبط می‌کند، دستور `save filename var` فقط متغیر `var` را در فایل ضبط می‌کند. متغیرهای ضبط شده در فایل، در اجراهای بعدی متلب با دستور `load filename` دوباره در فضای حافظه بار می‌شوند. اجرای دستور `fclose()` فراموش نشود.

مثال:

```
>> x = 0.1: 0.1: 1;
>> y = [x; log10(x)];
>> y = y';
>> save lgm y % saves only variable y
>> mtt
```

در اجرای بعدی متلب:

```
>> clear
>> load lgm
>> y
```

```
y = 0.1000   -1.0000
    0.2000   -0.6990
    0.3000   -0.5229
    0.4000   -0.3979
    0.5000   -0.3010
    0.6000   -0.2218
    0.7000   -0.1549
    0.8000   -0.0969
    0.9000   -0.0458
    1.0000         0
```

باز کردن درگاه port

مثال:

درگاه COM3 را باز کرده، خواص آن را مشاهده، و آن را می‌ببندیم.

```
>> sr = serial('COM3');
>> fopen(sr)
>> sr
>> fclose(sr)
```

```
Serial Port Object : Serial-COM3
   Communication Settings
      Port:               COM3
      BaudRate:           9600
      Terminator:         'LF'
   Communication State
      Status:             open
      RecordStatus:       off
   . . . .
```

۴-۴ تمرین

۱- عدد `in = 9322034.52347468` را با فرمت‌های `%e` و `%10.3f` و `%g` نمایش دهید، و از کدهای`\t`و `\n` نیز استفاده کنید.

۲- سه عدد `av1 = 17.45` و `av2 = 18.34` و `av3 = 15.8` را در فایلی به نام `num.txt` بنویسید. فایل را `fclose()` کنید. فایل را برای مشاهده باز کنید. فایل را با دستور `load num.txt` به صورت یک آرایه به نام `num` وارد حافظه کرده و مقدار آن را ملاحظه کنید. فایل `num.txt` را پاک کنید. **راهنما:** اعدادی که به صورت آرایه در یک فایل متن نگه‌داری شوند با دستور `load filename` در حافظه به صورت متغیر بار می‌شوند.

فصل ۵ آشنائی با ترسیمات

۵-۱ صفحه های مختصات

صفحه مختصات قائم

اعلب ترسیمات متلب در میان دو محور (سه محور برای رسم سه بعدی) رسم می‌شوند. محورهای افقی، عمودی، و فضائی در متلب به ترتیب X و Y (و Z برای رسم سه بعدی) نام دارند. مختصات نقاطِ سازنده منحنی ها (سطوح برای رسم سه بعدی) روی این محورها مشخص می‌شوند.

بعضی از دستورات ترسیمی نظیر خانواده‌ی plot () توسط متلب درون صفحه مختصات قائم(طول X ، عرض Y) رسم می‌شوند.

صفحه مختصات قطبی

دستوراتی نظیر polar () , compass () درون صفحه مختصات قطبی (زاویه θ ،بزرگی r) رسم می‌شوند.

تبدیل مختصات قائم و قطبی

نحوه تبدیل مختصات نقطه‌ی A با مختصات قائم (x,y) و مختصات قطبی (r و theta) در جدول زیر آمده است.

فرمول	نوع تبدیل مختصات
[theta,r] = cart2pol(x,y)	قائم به قطبی
[x,y] = pol2cart(theta,r)	قطبی به قائم

۵-۲ بردار و رسم منحنی، دستورهای plot(), comet()

کار ترسیم آرایه‌ها برحسب یک‌دیگر (یا ترسیم توابعی که آرایه‌ها را به هم ربط می‌دهند) بسیار آسان است.

دستور plot (t,x) عناصر بردار x را بر حسب بردار t نظیر به نظیر به صورت نقاطی روی صفحه مختصات قائم قرار می‌دهد، (هر دو بردار باید متساوی‌العنصر باشند) سپس آن‌ها را به هم وصل و منحنی مورد نظر را ایجاد می‌کند.

مثال توضیحی زیر اصول کار را بیان می‌کند.

مثال:

برای رسم معادله یک سهمی ابتدا یک آرایه شامل مقادیر x تعریف می‌کنیم:

```
>> x = [-4 -3 -2 -1 0 1 2 3 4];
```

سپس تابع مورد نظر را مینویسیم:

```
>> y = [-21 -13  -7  -3  -1  -1  -3  -7 -13];
```

و با یک دستور ساده y را بر حسب x نقطه به نقطه رسم می‌کنیم (شکل ۵-۱)

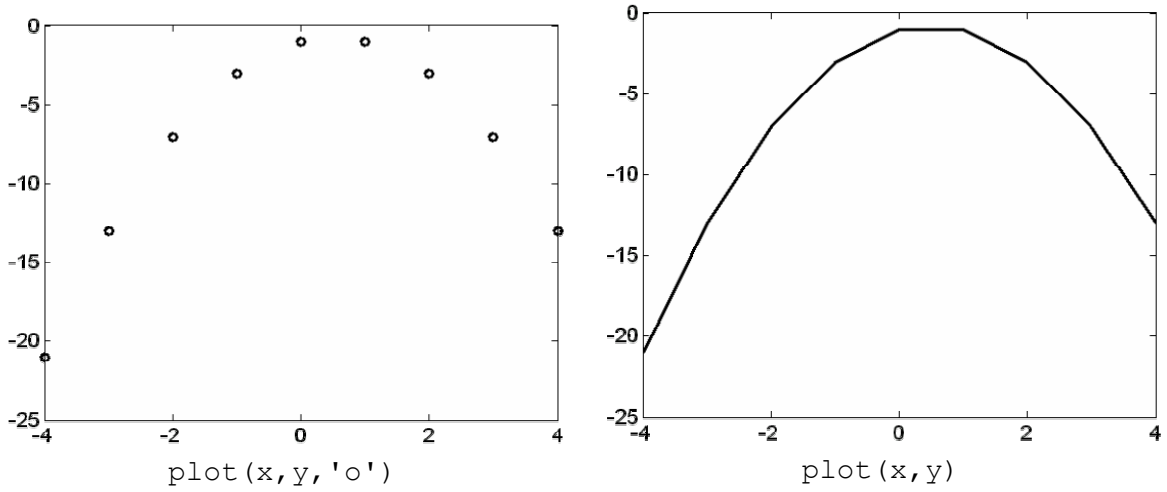
```
>> plot(x,y,'o')
```

اگر پارامتر 'o' را از دستور حذف کنیم، نقاط به صورت اتوماتیک به یک‌دیگر وصل و ترسیم شکل منحنی به خود می‌گیرد. برای داشتن منحنی بدون شکست باید نقاط بیشتری را به x و نتیجتاً به y نسبت داد. دقت کنید که برای رسم دو آرایه عددی برحسب یک‌دیگر آن دو باید هم‌سان باشند. صورت فرمولی معادله سهمی چنین نوشته می‌شود:

```
y = -x.^2 + x - 1
```

برای به توان رساندن تک‌تک عناصر آرایه (توان آرایه‌ای) از علامت ^ . استفاده شده است.

دستور comet () عیناً شبیه plot () عمل می‌کند با این تفاوت که رسم را با حرکت آهسته به صورت پویا نمائی انجام می‌دهد (امتحان کنید).



شکل ۵-۱

دستورهای figure(), subplot(), hold on/off, clf

دستورهای زیادی در ارتباط با ترسیمات وجود دارند که جهت آشنائی برخی از معمول‌ترین آن‌ها را ذکر می‌کنیم:

دستور `clf` پنجره جاری گراف را پاک کرده و برای گراف جدید باز نگه‌می‌دارد.

دستور `hold on` پنجره جاری را برای گراف جدید باز نگه‌می‌دارد.

دستور `subplot(2,2,n)` صفحه گراف را به چهار قسمت مساوی تقسیم می‌کند، عدد `1=< n >=4` یکی از این چهار قسمت را فعال می‌کند. دستور `subplot(1,2,m)` و دستور `subplot(2,1,m)` صفحه گراف را به ترتیب به دوقسمت عمودی و افقی تقسیم می‌کنند `1=< m >=2` . امتحان کنید.

دستور `figure(p)` یک پنجره جدید گراف با شماره `p` باز و آن را پنجره جاری می‌کند.

افزودن توضیحات روی منحنی

جدول زیر بعضی از توضیحاتی را که می‌توان برروی گراف آورد نشان می‌دهد. سه نقطه یعنی شرح مورد نظر خودتان:

شرح	عنوان منحنی	برچسب محور x	برچسب محور y	برچسب رنگ چند منحنی
دستور	`title('...')` `title '...'`	`xlabel('...')` `xlabel '...'`	`ylabel('...')` `ylabel '...'`	`legend()`

پنجره Data Statistics

اگر پس از رسم نمودار در پنجره `Figure` زیر- منیوی `Tools_Data Statistics` را انتخاب کنیم، ترتیب منحنی‌ها روی پنجره `Figure` نشان داده شده، و پنجره `Data Statistics` ظاهر می‌شود این پنجره مقادیر آماری منحنی را نشان می‌دهد. اگر در مقابل هریک ازاین مقادیر چک‌مارک بزنیم، مقدار آن روی منحنی هم نشان داده خواهد شد.

چند منحنی در یک صفحه

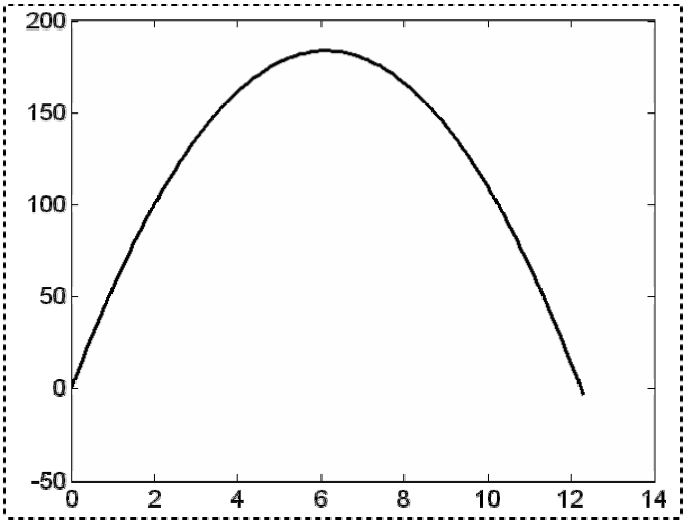
عبارت `plot(x,y1,x,y2)` منحنی‌های `y1` و `y2` را با دو رنگ مختلف بر حسب `x` رسم می‌کند.

مثال ها:

پرتابه عمودی با `plot()`

سنگی با سرعت اولیه 60 در زمان صفر، عمودی به بالا پرتاب می‌شود، نمودار مسافت- زمان را تا زمان `12.3` با گام `0.1` رسم کنید.

```
>> g = 9.8;
>> v0 = 60 ;
>> t = 0: 0.1: 12.3;
>> x = v0*t - g/2 * t.^2 ;
>> plot(t,x)
```

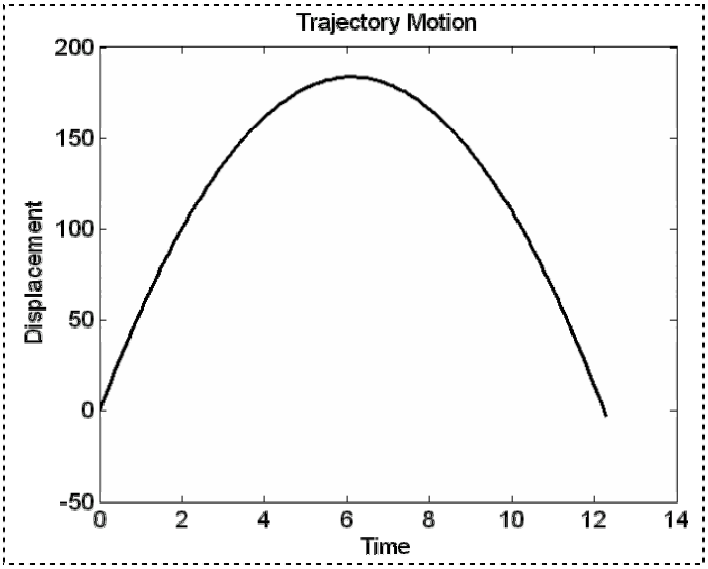


شکل ۵-۲

برچسب ها

برای مثال فوق برچسب های مناسب برای عنوان، و محورهای x و y را ایجاد کرده و اثر آن‌ها را بر روی گراف ببینید.

```
>> title ...
'Trajectory Motion'
>> xlabel('Time')
>> ylabel('Displacement')
```



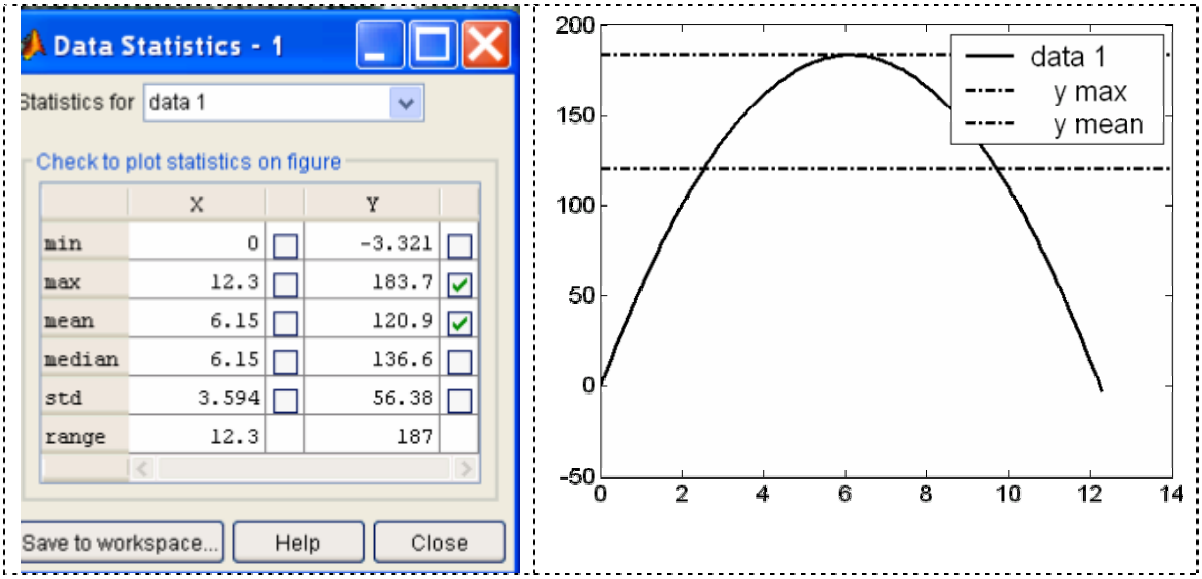
شکل ۵-۳

اطلاعات آماری Data Statistics

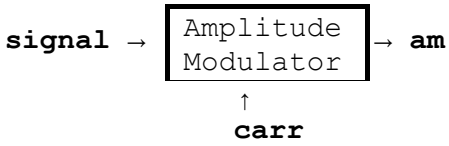
برای منحنی فوق بعضی از مقادیر آماری را روی پنجره Data Statistics علامت بزنید و نتایج آنرا روی پنجره Figure ببینید (شکل ۵-٤)..

مدولاتور دامنه AM Modulator

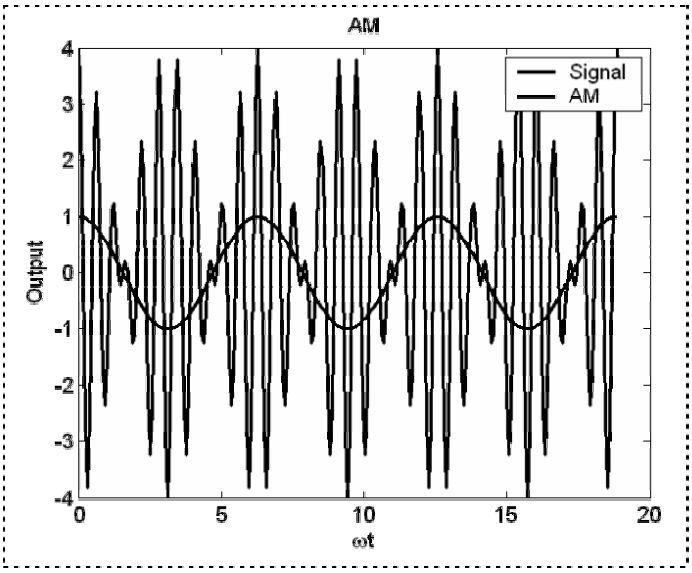
در مدولاتور دامنه زیر، سیگنالِ ۱۰۰۰ هرتزی sig با کریر ۱۰۰۰۰ هرتزی carr ودامنه ٤ برابر sig مدوله می‌شود. خروجیِ am و sig را برحسب زمان رسم کنید (شکل ۵-۵).



شکل ۵-٤ پنجره آمار داده ها




```
>> t = 0:12.5e-6:0.003;
>> f = 1000;
>> om = 2*pi*f;
>> omt = om*t;
>> sig = cos(omt);
>> carr = 4*cos(10*omt);
>> am = sig.*carr;
>> plot(omt,sig,omt,am)
>> legend('Signal','AM')
>> title('AM')
>> xlabel('\omegat')
>> ylabel('Output')
```



شکل ۵-۵

دستور **legend** رنگ یا استیل هر منحنی را برچسب می‌گذارد.

علامت \omegat کاراکتر TeX نام دارد که در گراف به صورت ωt نمایش داده می‌شود. برای اطلاع بیشتر در مورد کاراکترهای TeX ، از منوی Help زیر منوی Help MATLAB را اجرا و TeX characters را جستجو کنید.

۵-۳ دستورهای هم خانواده plot()

دستورهای ترسیمیِ `plotyy()`, `loglog()`, `semilogy()`, `semilogx()` با اندکی تغییر مشابه `plot()` هستند. برای اطلاع بیشتر به `help plot` و مثال‌های فصول بعد مراجعه کنید.

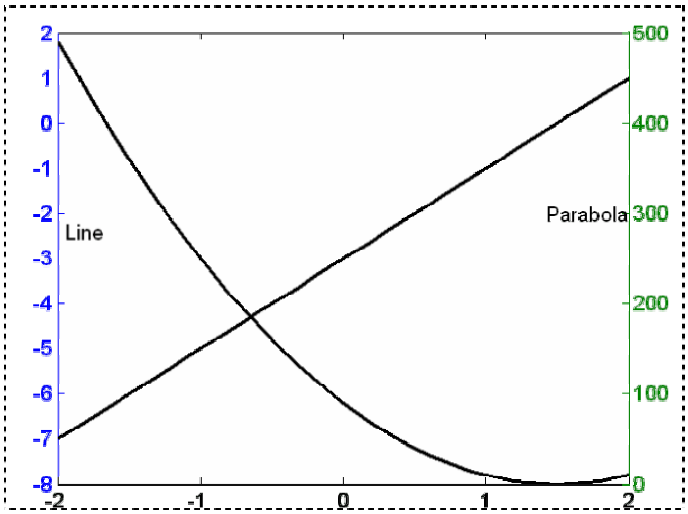
به عنوان نمونه دستور `plotyy()` را بررسی می‌کنیم. رسم دو تابع که مقادیر آن‌ها با یک‌دیگر فاصله زیاد دارند روی یک گراف ناممکن است، اما امکان استفاده از دو محور با استفاده از دستور `plotyy(x1,y1x2,y2)` این مشکل را حل کرده و اجازه رسم دو تابع با مقادیر دور از هم روی یک گراف را می‌دهد.

مثال:

خط و سهمی را با مقادیر دور ازهم با `plotyy()` رسم کنید. برای برچسب‌گذاری روی محورهای عمودی از گزینه

Tools_Edit Plot یا دکمه Text استفاده کنید. محور افقی برای هردو تابع یکسان است.

```
>> x = [-2: 0.1: 2];
>> y1 = 2*x - 3;% Line
>> y2 = 10 * y1.^2;
%Parabola
>> plotyy(x,y1,x,y2)
```



شکل ۵-۶

۵-۴ روش های دیگر نمودار سازی

نمودار ستونی bar()

دستور bar(x,y,w) بردار y را برحسب بردار x به صورت میله‌ای رسم می‌کند. معمولاً از این نوع نمودار برای نمایش آماری بردارهای کم‌تعداد استفاده می‌شود. پارامتر w عرض میله‌ها را تعیین می‌کند که پیش‌فرض آن 0.8 است. دستورهای stairs(), barh(), stem() نزدیک به bar() هستند (امتحان کنید).

مثال:

نمودار ستونی درجه حرارت یک روز را بین ساعت ۸ تا ۲۴ با فاصله دو ساعته رسم کنید (شکل ۵-۷).

پیشینه نگار hist()

این تابع که پیشینه‌نگار histogram رسم می‌کند، یکی از دستورات گرافیکی مهم است. طرز کار آن این‌گونه است که ابتدا مجموعه‌ای از داده‌ها (اشیاء) را مرتب می‌کند، و سپس در چند ظرف (به صورت پیش‌فرض ۱۰ ظرف) جا می‌دهد. هر ظرف، حجمی از داده‌های نزدیک به هم را درخود می‌گنجاند. سپس محتوای هر ظرف برحسب درشتی داده‌های داخل آن‌ها نمایش داده می‌شود.

مثال:

نمرات زیر را در تعداد ظرفِ معادل یک‌چهارم تعداد دانشجو ریخته و پیشینه نگار را با توضیح رسم کنید(شکل ۵-۸).

دستور strcat() رشته‌ها را سرهم می‌کند. برای اطلاع بیشتر از به فصل رشته‌ها مراجعه کنید.

```
>> scr = [12 14.56 18.44 16 8.3 19.1 18.2 16 5.3 7.8 15 12 14.6
8.8 17 11.2 13.25 12 13 9 14 11 12 11.5 15 15 7 4 6 11 12 8 9];
```

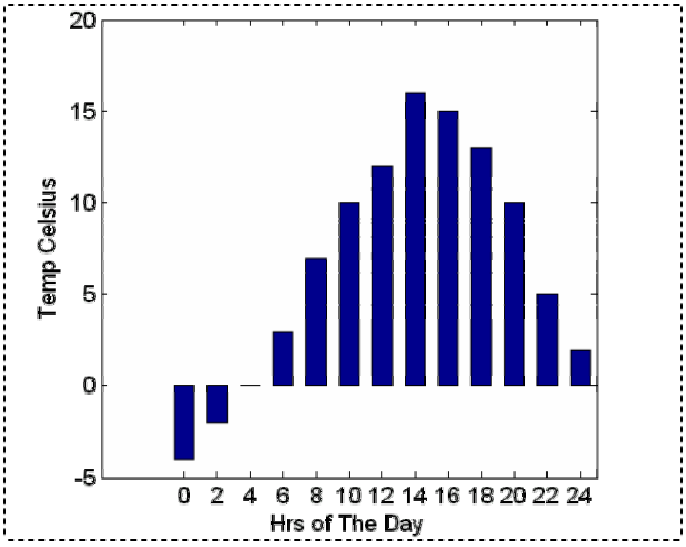
نمودار دایره pie()

دستور pie(p, w, m) یک دایره را به تعداد داده‌ها قطاع می‌زند، به نحوی‌که به هر داده سطحی متناسب با بزرگای آن اختصاص یابد. برای نمودارهای آماری نسبی استفاده می‌شود. p درصد یا مساحت هر قطاع، w فاصله بین قطاع‌ها، و m توضیح یا برچسب هر قطاع است. دستور pie3() هم مشابه pie() است (امتحان کنید).

مثال:

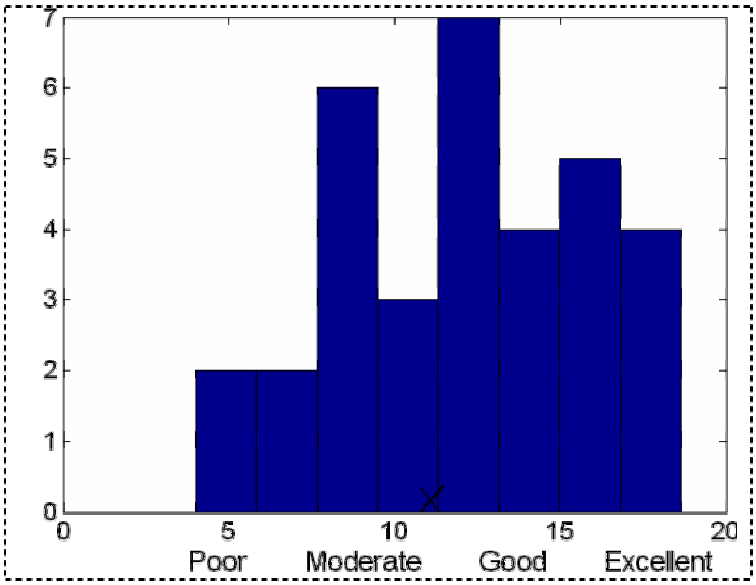
نسبت جمعیتی چند شهر را با نمودار دایره‌ای نمایش دهید(شکل ۵-۹).

```
>> hr = 0:2:24;
>> temp = [-4 -2 0 3 7...
   10 12 16 15 13 10 5 2];
>> bar(hr,temp,0.6)
>> xlabel('Hrs of The Day')
>> ylabel('Temp Celsius')
```



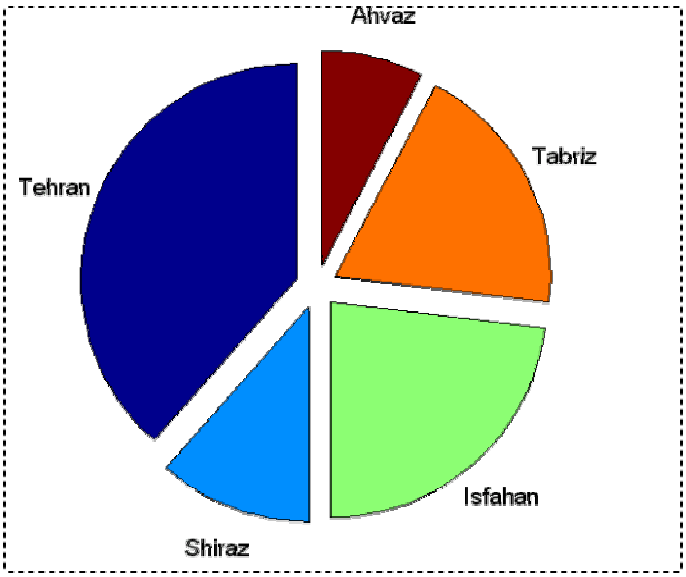
شکل ۵-۷

```
>> Ls = length(scr);
>> hist(scr,Ls/4)
>> sp=repmat(...
[' '],1,8);
>> xb =
strcat([sp,sp,...
          'Poor',sp,...
      'Moderate',sp,...
          'Good',sp, ...
      'Excellent']);
% string addition
>> xlabel(xb)
```



شکل ۵-۸

```
>> p = [10,3,6,5,2];
>> w = [0.5 1 1 0.5 1];
>> m =
{'Tehran','Shiraz',...
'Isfahan','Tabriz','Ahvaz'}
% m is a cell array
>> pie(p, w, m)
```



شکل ۵-۹

۵-۵ رسم نموداری ماتریس

نمودار ستونی ماتریس

وقتی نمودار ستونی یک ماتریس رسم شود، هر ردیف در یک مجموعه جداگانه از میله‌ها همراه با شماره ردیف نشان داده می‌شود.

نمودار منحنی ماتریس

وقتی نمودار منحنی یک ماتریس رسم شود، هر ستون در یک منحنی جداگانه نشان داده می‌شود. برای تفکیک منحنی‌ها از دستور `legend()` یا از پنجره `Data Statistics` استفاده می‌کنیم.

مثال ها:

میزان ریزش باران پنج سال متوالی

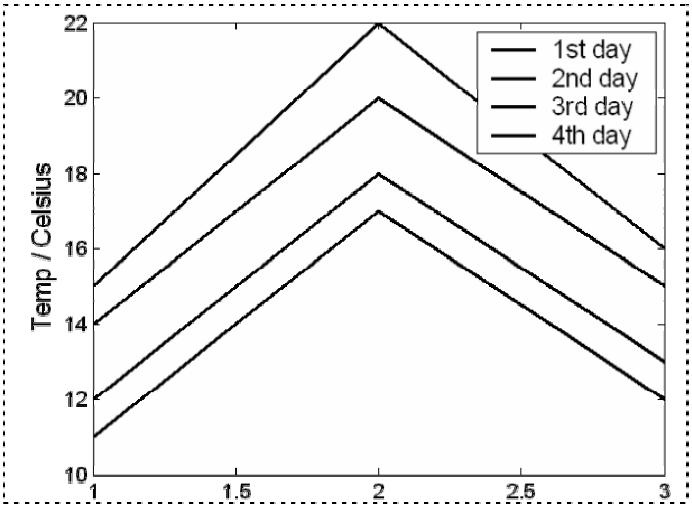
میزان ریزش باران هر فصل را در پنج سال پیاپی در ردیف‌های یک ماتریس قرار دهید. نمودار ستونی میزان باران هر سال را نمایش دهید. منحنی را خودتان رسم کنید.

```
>> rain = [110 70 125 152;     % 1st year
           210 55 104 223;     % 2nd year
           120 56 173 156;     % 3rd year
           195 72 211 178;     % 4th year
           118 58 123 149];    % 5th year
>> bar(rain)
>> xlabel('Year 1 to year 5'), ylabel('Amount of rain per season')
>> legend('Spring','Sumer','Fall','Winter')
```

درجه حرارت چهار روز متوالی

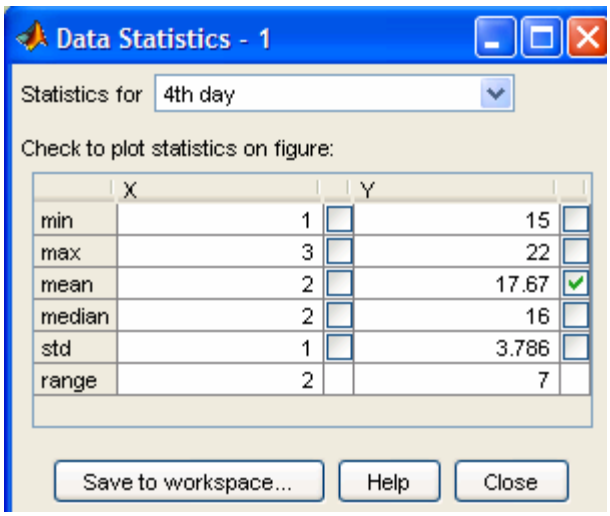
درجه حرارت صبح، ظهر، و غروب چهار روز پیاپی را در ستون‌های یک ماتریس قرار دهید. نمودار منحنی درجه حرارت هر روز را نمایش دهید. میانگین درجه حرارت روز چهارم را با استفاده از پنجره `Data Statistics` مشاهده کنید.

```
temp = ...
[11 12 14 15;
 17 18 20 22;
 12 13 15 16;];
plot(temp)
ylabel('Temp / Celsius')
legend( ...
'1st day','2nd day',...
'3rd day','4th day')
```



شکل ۵-۱۰

پس از رسم منحنی‌ها زیرمنیوی `Tools_Data Statistics` را از پنجره `Figure` انتخاب کرده، برای نمایش میانگین روز چهارم بر روی نمودار، مقابل مقدار مربوطه علامت می‌زنیم.



شکل ۵-۱۱

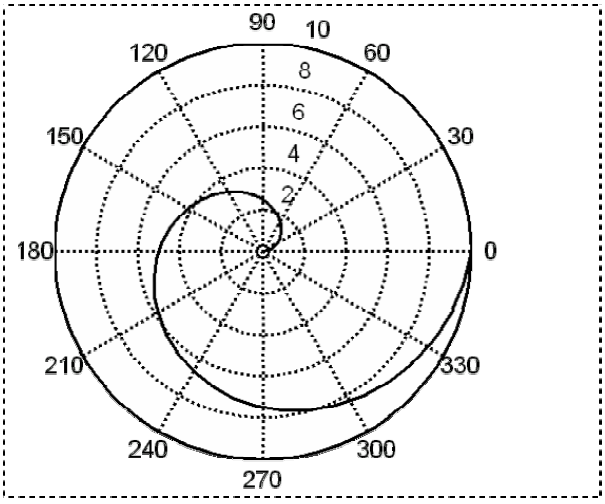
نمودار قطبی

عبارت `polar(theta,r)` مقادیر `r` را بر حسب زاویه `theta` روی مختصات قطبی نشان می‌دهد. نمودار قطبی `polar()` همان کار را که `plot()` با `x,y` انجام می‌دهد با `theta,r` انجام می‌دهد.

مثال:

مختصات قطبی آرایه‌ای از بردار- هندسی‌های افزاینده به صورت مارپیچ را تعریف و بزرگی هر بردار- هندسی را برحسب زاویه‌اش رسم می‌کنیم (شکل ۵-۱۲).

```
>> tet = linspace(0, 2*pi, 40);
>> r = linspace(0,10,40);
>> polar(tet,r)
```



شکل ۵-۱۲

نمودار عقربه ای، دستور compass()

دستور `compass(x,y)` یک بردار- هندسی با طول و عرض قائم `x,y` را درون مختصات قطبی نمایش می‌دهد.

از این نمودار برای نمایش هندسی اعداد مختلط نیز استفاده می‌شود.

مثال ها:

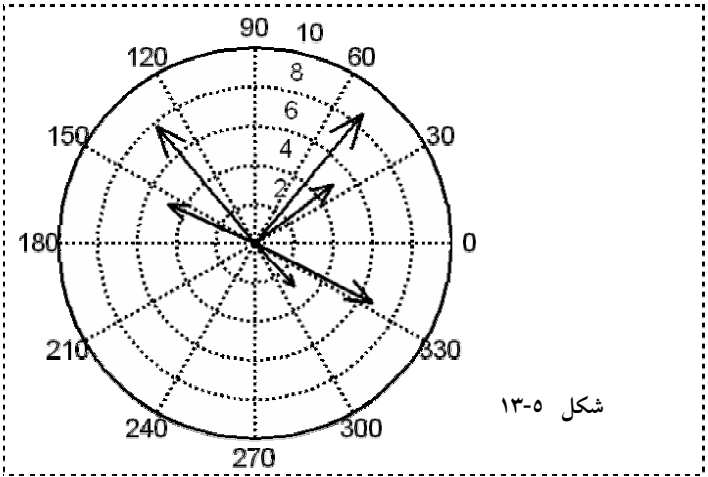
بردار- هندسی شش نقطه

برای دیدن دستورات و ترسیم به شکل ۵-۱۳ مراجعه کنید.

مختصات قائمِ بردار- هندسی‌هائی با ده عنصر زیاد شونده.

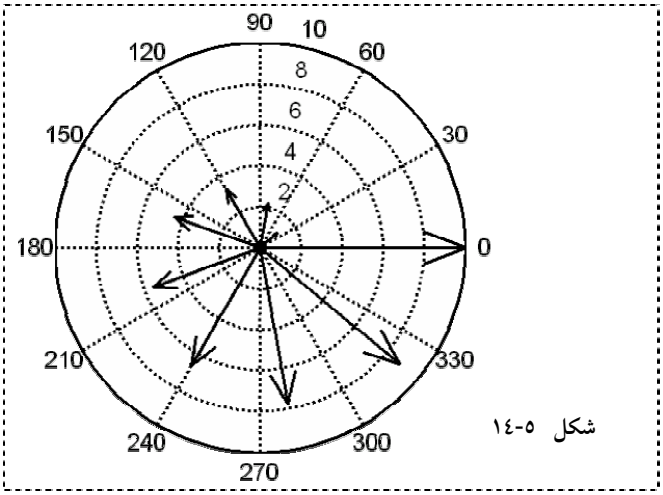
برای دیدن دستورات و ترسیم به شکل ۵-۱۴ مراجعه کنید.

```
>> x = [-5 4 -4.4 2 ...
        5.5 6];
>> y = [6 3 2 -2.2 ...
        6.6 -3];
>> compass(x,y)
```



شکل ۵-۱۳

```
>> tet = ...
linspace(0, 2*pi, 10);
>> r = linspace(0,10,10);
>> [x y] = pol2cart(tet,r);
>> compass(x,y)
```



شکل ۵-۱۴

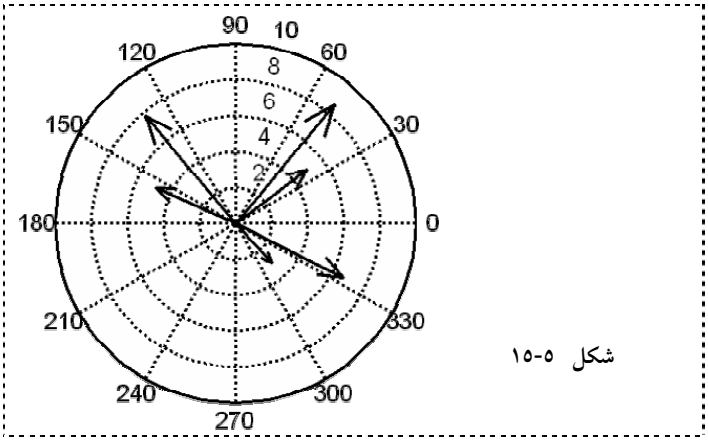
نمایش هندسی عدد مختلط با ()compass

برای نمایش بردار- هندسیِ اعداد مختلط از دستور compass() استفاده می‌کنیم. دستور compass(z) معادل compass(real(z),imag(z)) می‌باشد.

مثال ها:

نمایش بردار-هندسی فرم قائم

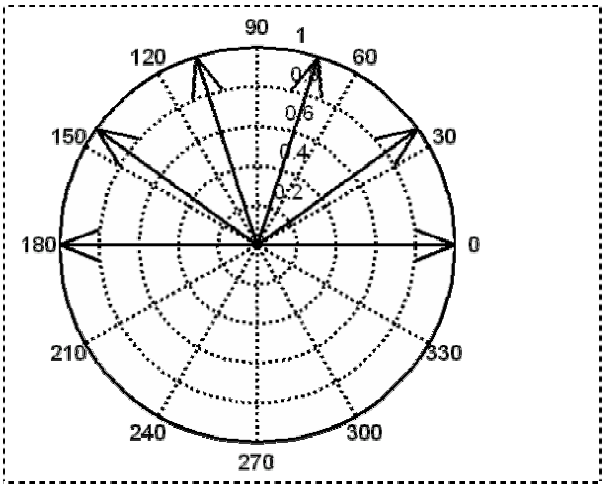
```
>> M = ...
 [-5+6*i,      3*j+4;
  -4.4+2*i,    2-2.2*i;
   5.5+6.6*j,  6-3*j ];
>> compass(M)
```



شکل ۵-۱۵

نمایش بردرا- هندسی فرم قطبی

```
>> tet = 0:pi/5:pi;
>> compass(exp(j*tet))
```



شکل ۵-۱۶

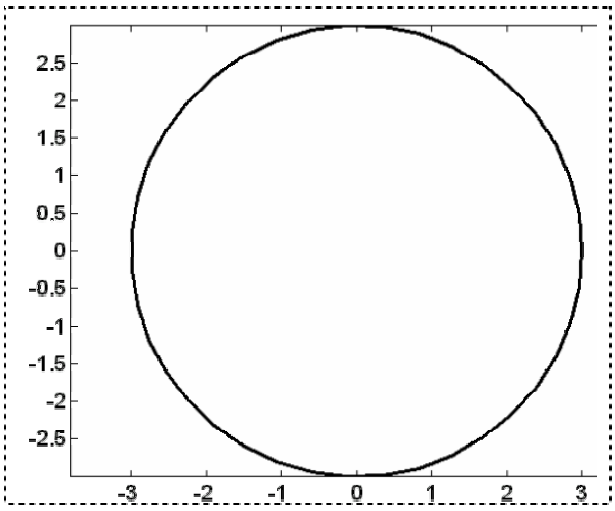
رسم عدد مختلط با plot()

اگر zp مختلط باشد، `plot(zp)` قسمت موهومی آن‌را برحسب قسمت حقیقی رسم می‌کند. اگر zp مختلط و t یک متغیر حقیقی باشد، عبارت `plot(t,zp)` قسمت حقیقی zp را برحسب t رسم کرده و از قسمت موهومی صرف‌نظر می‌شود.

مثال:

قسمت موهومی تابع مختلط $z_p = re^{j\theta} = r(\cos\theta+j\sin\theta)$ را برحسب قسمت حقیقی آن رسم کند. (توجه کنید که منحنی سینوس (قسمت موهومی) بر حسب کسینوس (قسمت حقیقی) یک دایره مثلثاتی با شعاع r است $(\cos^2\theta + \sin^2\theta = r^2)$.

```
>> tet = linspace(0,2*pi,40);
>> r = 3;
>> zp = r*exp(j*tet);
% or zp=r*(cos(tet)+j*sin(tet))
>> plot(zp)
% Sine versus Cisine
% is a circle
>> axis equal
```



شکل ۵-۱۷

دستور `axis equal` مقیاس عرضی و طولی مانیتور را (که معمولاً ۶ به ۵ است) یکسان می‌کند وگرنه دایره، بیضی دیده می‌شود.

مثال:

zp مثال فوق و قسمت حقیقی آن را برحسب tet رسم می‌کنیم. نتیجه هر دو ترسیم منحنی‌های کسینوسی مشابه است (امتحان کنید).

۵-۶ رسم آسان با ezplot()

در متلب بعضی از توابع بدون مقدار دهی به متغیر با توابعی مانند ezplot(f(x)) رسم می‌شوند. ezplot() دامنه متغیر را به صورت پیش‌فرض $-2\pi < x < +2\pi$ قرار می‌دهد. فرم ezplot(f(x),[min,max]) برای تعیین دستی دامنه متغیر به‌کار می‌رود. آرگومان ezplot() که یک تابع است، به صورت رشته، گیره تابع، یا تابع سطری inline نوشته می‌شود (این‌ها روش‌هائی هستند که برای ارسال یک تابع به تابع دیگر در متلب وجود دارند). برای اطلاع بیشتر در مورد آرگومان تابعی یا تابعِ تابع به مباحث بعدی مراجعه کنید.

رسم توابع آشکار explicit functions

این‌گونه توابع به‌صورت y = f(x) می‌آیند مثل: $y = -2x^2 + 3$. توابع آشکار را می‌توان با plot() رسم کرد، اما رسم بعضی از این توابع مثل y = tan(x) که در دامنه معمول x مقادیر بزرگ پیدا می‌کنند بوسیله plot() راحت نیست (به مثال مراجعه نمائید). اما ezplot() این‌گونه توابع را هوشمندانه و به‌راحتی رسم می‌کند. دامنه پیش‌فرض با گام مناسب نقطه‌گذاری‌شده، فرمول تابع، و برچسب محور x نیز خودبه‌خود نوشته می‌شوند.

رسم توابع ضمنی implicit functions

توابع ضمنی توابعی هستند که در آن‌ها x و y به صورت مخلوط می‌آیند، مثل: $x^2 + y^2 = 1$. کاربرد plot() به طور مستقیم در این‌گونه موارد نتیجه مطلوب نمی‌دهد، اما ezplot() به‌راحتی از عهده برمی‌آید. وقتی یک تابع ضمنی به ezplot() ارسال شود درطرف راست معادله خودبه‌خود صفر قرار می‌گیرد.

رسم توابع پارامتریک

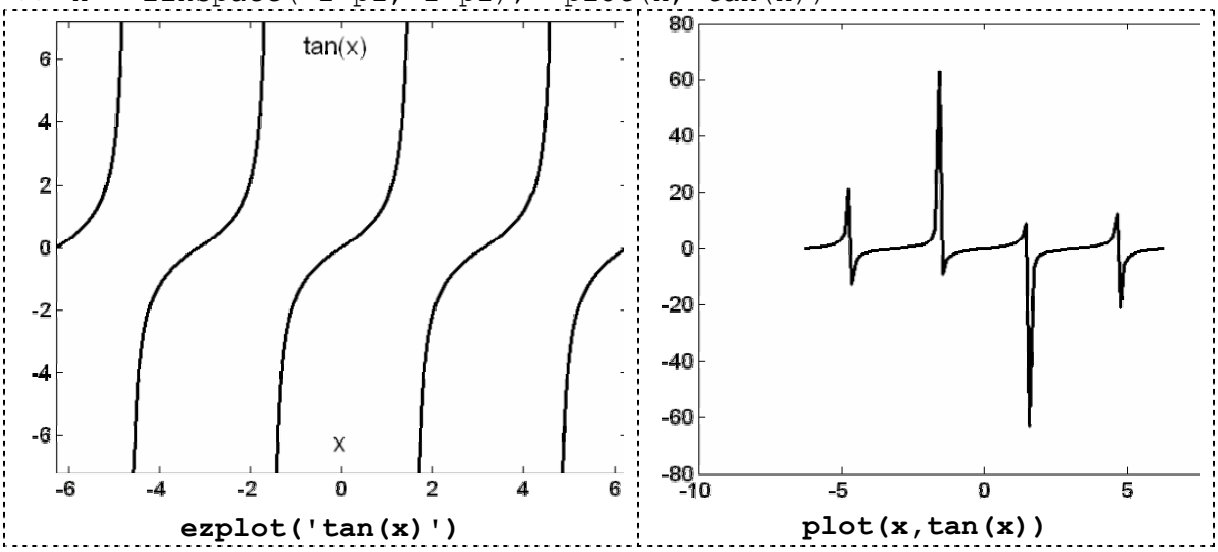
اگر f و g توابع t باشند، با ezplot(f,g) می‌توان آن دو را برحسب یک‌دیگر رسم و پارامتر t را حذف کرد. دامنه پیش‌فرض t فاصله $0, \pi$ است.

مثال ها:

تابع آشکار

تابع y = tan(x) را بر حسب x با plot() و ezplot() رسم کنید.

```
>> ezplot('tan(x)')
>> x = linspace(-2*pi, 2*pi);  plot(x, tan(x))
```

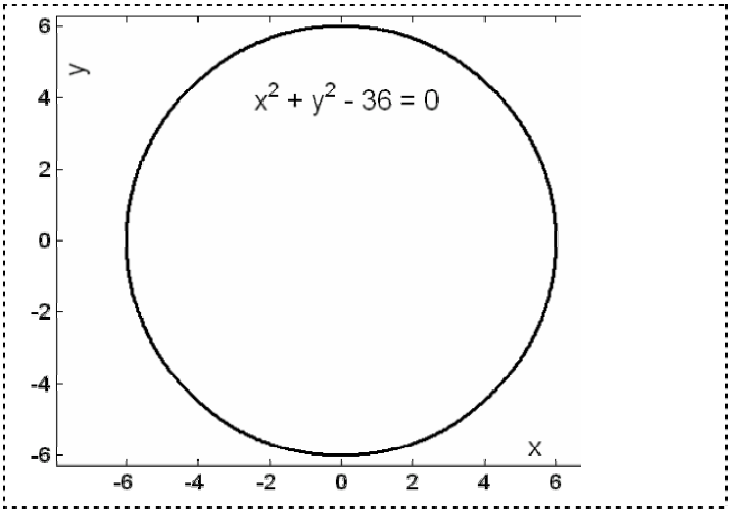


شکل ۵-۱۸

تابع ضمنی

یک دایره با شعاع 6 را با ezplot() رسم کنید.

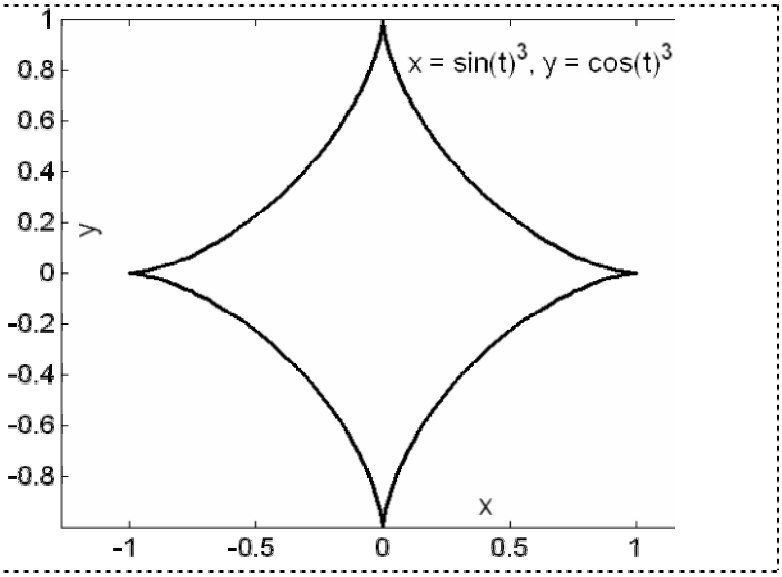
```
>> fc = 'x^2 + y^2 - 36';
>> ezplot(fc)
>> axis equal
```



شکل ۵-۱۹

تابع پارامتریک

```
>> f1 = 'cos(t)^3';
>> f2 = 'sin(t)^3';
>> ezplot(f2,f1)
```



شکل ۵-۲۰

۵-۷ تابع داخلی fplot()

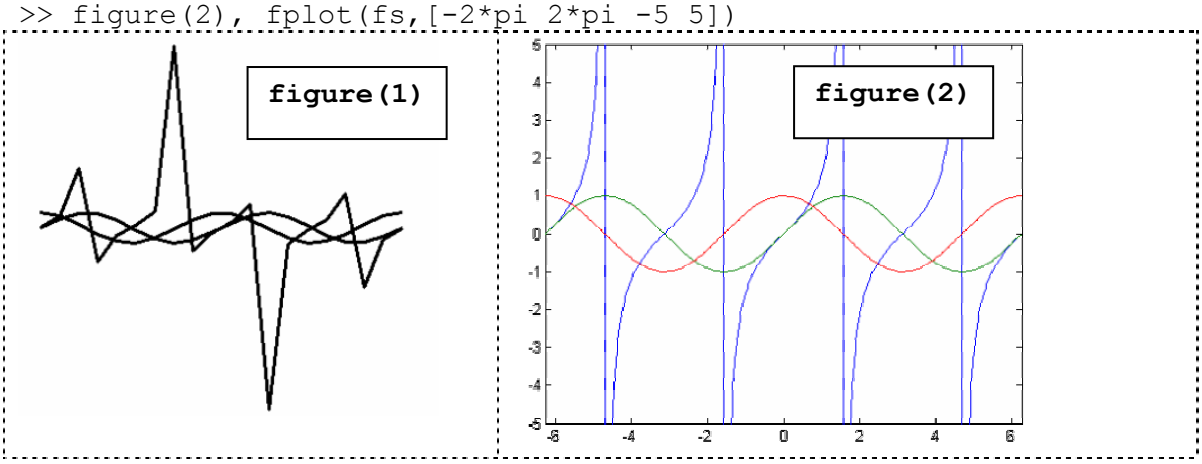
فرم کلی این تابع چنین است fplot(fun,lims) . fun یک تابع آشکار از x است که همانند آرگومان ezplot() باید به صورت یک عبارت رشته‌ای، گیره‌ی تابع، یا تابع inline باشد (برای اطلاع بیشتر در مورد آرگومان تابعی به مباحث بعدی مراجعه کنید). lims محدوده محورها را تعیین می‌کند. fun می‌تواند یک بردار از توابع گوناگون، یا یک ماتریس باشد که هر ستون آن یک تابع جداگانه است. lims یا باید به‌صورت [XMIN XMAX]باشد یا به‌صورت [XMIN XMAX YMAX YMIN]. در صورت اول محدوده محور عمودی خودبه‌خود تعیین خواهد شد، در صورت دوم محدوده محور عمودی دستی تعیین می‌شود.

fplot() مقادیر ناهم‌گونِ توابع را طوری تنظیم می‌کند که ترسیم شکیلی حاصل شود، به‌خصوص وقتی توابع دارای مقادیر دور ازهم باشد ۰شکل ۵-۲۱).

مثال:

منحنی‌های سینوس، کسینوس، و تانژانت را بر روی یک گراف با plot() و fplot() نمایش می‌دهیم. چون تغییرات تانژانت نسبت به سینوس و کسینوس سریع است، با plot() منحنی خوبی نخواهیم داشت.

```
>> x = linspace(-2*pi, 2*pi, 20);
>> figure(1), plot(x,tan(x),x,sin(x),x,cos(x))
>> fs = '[tan(t) sin(t) cos(t)]';
>> figure(2), fplot(fs,[-2*pi 2*pi -5 5])
```



شکل ۵-۲۱

۵-۸ ویرایش گراف

تعیین محدوده محورها

متلب خود به خود میزان محورها را تنظیم می‌کند، اما با دستور زیر می‌توان درجه محورها را به دلخواه عوض کرد:

```
axis( [xmin, xmax, ymin, ymax] )
```

دستور axis auto درجه‌گذاری را به حالت معمول بر می‌گرداند.

توری روی گراف از پنجره فرمان

تکرار grid در پنجره فرمان برای آوردوبُردِ توری روی گراف است. از grid on و grid off هم می‌توان استفاده کرد (امتحان کنید).

برچسب گذاری با ماوس

gtext() امکان برچسب گذاری در هر محل ترسیم را با کلیک ماوس فراهم می‌کند. برای امتحان روی یک گراف دستور gtext('label') را اجرا کرده و با خط موی ایجاد شده روی نقطه‌ای از گراف کلیک کنید.

مؤلفه های RGB

جدول شکل ۵-۲۲ بعضی از رنگهائی را که از ترکیب سه رنگ اصلی قرمز، سبز، آبی RGB به دست می‌آیند نشان می‌دهد. برای مشاهده این جدول منیوی Help_MATLAB Help را انتخاب و در لبه Index کلمه RGB را بنویسید.

رنگی کردن با fill()

تصویر را رسم و آنرا با رنگ مورد نظر پر می‌کند (امتحان کنید).

ادیت از روی پنجره گراف

با زدن دکمه edit plot از میله ابزار tool bar یا انتخاب منیوی Tools_Edit Plot پنجره

Property Editor ظاهر و امکان ویرایش دستی گراف فراهم می‌شود. با استفاده از این پنجره می‌توان شاخصه-های صورت گراف Surface ، که همان منحنی یا سطح رسم شده است، و شاخصه‌های محورها (Axes) را تغییر داد (امتحان کنید).

رنگ حاصل	قرمز	سبز	آبی
سیاه	0	0	0
سفید	1	1	1
قرمز	1	0	0
سبز	0	1	0
آبی	0	0	1
زرد	1	1	0
ارغوانی	1	0	1
فیروزه‌ای	0	1	1
خاکستری	0.5	0.5	0.5
قرمز تیره	0.5	0	0
مسی	1	0.62	0.40
کبود	0.6	0.4	0.8

شکل ۵-۲۲

۵-۹ گیره های گرافیک graphics handles

هر گراف دارای ۳ مؤلفه است:

الف- شیئ که به ترسیم یا دکمه یا هر نوع شیئ داخل گراف گفته می‌شود.

کلمه gco (get handle to current object) گیره شیئ را برمی‌گرداند.

ب- محورها، که محوطه درون محورها (صفحه مختصات) است.

کلمه gca (get handle to current axes) گیره محورها را برمی‌گرداند.

ج- پنجره تصویر که مجموعه گراف است. gcf (get handle to current figure) گیره پنجره تصویر را برمی‌گرداند.

شاخصه‌های هر گیره با دستور get () به دست می‌آید. با استفاده از این گیره‌ها می‌توان شاخصه‌های گرافیک را با دستور set () به دلخواه تنظیم کرد.

مثال:

روی منحنی یک گراف کلیک و شاخصه‌های properties گیره را با دستور get () مشاهده می‌کنیم.

```
>> get(gco)
```

```
Color = [0 0 1]
EraseMode = normal
LineStyle = -
LineWidth = [0.5]
Marker = none
...
```

سپس بعضی از شاخصه‌هارا با دستور set () تنظیم و نتایج را ملاحظه کنید. برای حصول نتیجه روی شیئ مورد نظر کلیک و آنرا به شیئ فعال تبدیل کنید.برای بهتر دیدن شیئ فعال گزینه edit plot را از میله ابزار انتخاب کنید.

```
>> set(gca, 'color', [1 1 0.5])
```

```
>> set(gcf, 'color', [1 0.8 0.4])
>> set(gco, 'marker', 'o')
>> set(gco, 'linewidth', 4)
```

۵-۱۰ تمرین

۱- یک منحنی مثلثاتی را با هر یک از این فرم‌ها رسم کنید

`plot(x,y,y,x,'rx') , plot(x,y,x,y,'rx')` .

۲- مقادیر `log10(x)` را به ازای `x = 0.1: 0.1: 1` را به صورت یک بردار در یک فایل متن نگه‌داری کرده، پس از بار کردن آن فایل منحنی مربوطه را رسم کنید.

۳- در یک مدار سری `R=5,C=100e-6,L=4e-3` و `emf` ورودی 2 ولت است. شدت جریان را در محدوده صفر تا ۲۰۰۰ هرتز رسم کنید.

٤- توابع زیر را به صورت حذف پارامتر با `ezplot()` رسم کنید.

```
sin(3*t)*cos(t),   sin(3*t)*sin(t)'    , range [0,pi]
t*cos(t),          t*sin(t)'           ,range [0,4*pi]
```

۵- مختصات قطبی آرایه‌ای ده عنصری از بردار- هندسی‌های هم‌طول وچرخنده به صورت مدور را تعریف و بزرگی هر بردار- هندسی را برحسب زاویه‌اش رسم کنید. مختصات قائمِ این بردار- هندسی‌ها را به دست آورده، آن‌ها را با دستور `compass()` در مختصات قطبی رسم کنید.

٦- برای تابع `y1 = sinα` ، مقادیر `y1` را برحسب چند مقدار α روی مختصات قطبی نشان داده و با جدول `y1` برحسب α مطابقت کند.

۷- منحنی یک معادله درجه دو (سهمی) را بر مختصات قائم و بر مختصات قطبی رسم کنید.

۸- تابع $\frac{\sin x}{x}$ را با `ezplot()` رسم کنید.

۹- تابع $y=\sin\frac{1}{x}$ را با `fplot()` رسم کنید. راهنما: $\sin\frac{1}{x}$ وقتی به یک رشته نسبت داده می‌شود `sin(1/x)` نوشته شود، نه `sin(1./x)`

۱۰- نویز توزیع- یک‌نواختی با فرکانس‌های بین ۱۰۰۰ تا ۱۰۰۰۰ ایجاد کنید و نمودار آن را در حوزه فرکانس نمایش دهید.

۱۱- چند تابع مثلثاتی را به‌وسیله `fill()` را با رنگ‌های زیر امتحان کنید:

`'r','g','b','c','m','y','w','k'`

فصل ۶ برنامه نویسی

۶-۱ ام- فایل

تشکیل ام- فایل

با زدن دکمه New M-File از میله ابزار یا انتخاب File_New از میله منیو یک ادیتور برنامه نویسی موسوم به ادیتور ام- فایل باز می‌شود. برنامه‌ی شامل مجموعه‌ای از دستورات است که در M-File نوشته، و با نام Filename.m (بدون فضای خالی در بین حروف) ضبط می‌شود. Filename یک کلمه اختیاری است). برای اجرای برنامه می‌توانید پس از ضبط، نامِ ام- فایل را مانند یکی از فرامین متلب در پنجره فرمان بنویسید (ذکر پسوند .m ضروری نیست) یا از میله ابزار M-File دکمه Save and Run را کلیک کنید. در هر حال نتیجه به روی پنجره فرمان خواهد آمد.

در مثال‌های برنامه نویسی نام ام- فایل در ابتدای برنامه به صورت توضیح (درمقابل علامت %) ذکر شده که به هنگام اجرا از پنجره فرمان این نام (معمولاً بدون پسوند .m) تایپ و اجرا می‌شود. echo off باعث قطع نمایش دستورات برنامه بر روی پنجره فرمان می‌شود.

ام- فایلِ اسکریپت و ام-فایلِ تابعی

یک ام- فایل که شامل برنامه (مجموعه‌ای از دستورات) باشد ***ام- فایل اسکریپت*** Script M-File نام دارد و با وارد کردن نام آن از پنجره فرمان اجرا می‌شود. اما ***ام- فایل تابعی*** Function M-File شامل یک تابع کاربر- تعریف است و معمولاً از داخل برنامه‌های دیگر صدا call زده می‌شود. برای اطلاع بیشتر به مبحث ام- فایل تابعی مراجعه شود. تا قبل از این مبحث کلیه ام- فایل‌های مورد استفاده اسکریپت هستند.

۶-۲ مثال های ریاضی

محاسبه سرمایه نهائی بر حسب سرمایه اولیه

برنامه‌ای با نام BankIn.m بنویسید که جدول سرمایه نهائی بر حسب سرمایه اولیه (تومان) را با نرخ سود r پس از n سال برای پنج نمونه از مقدار سرمایه اولیه به دست دهد.

```
% BankIn.m
echo off;
format bank
A = [75000 100000 300000
  500000 1000000]; % مقادیر سرمایه اولیه
r = 0.09;
n = 10;
B = A * (1+r)^n;
nama = [A' B']

>> BankIn.m
```

```
nama =  75000.00      177552.28
       100000.00      236736.37
       300000.00      710209.10
       500000.00     1183681.84
       1000000.00     2367363.67
```

در محاسبه مقدار B به علامت ضرب بدون نقطه (ضرب بردار در سکالر)، و اولویت عملگرها توجه کنید.

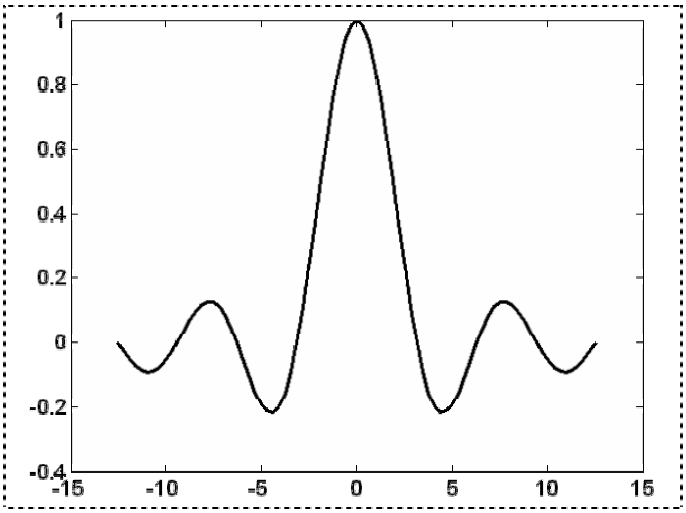
تقسیم صفر بر صفر

منحنی $\frac{\mathrm{Sin}\ x}{x}$ را در فاصله -4π تا 4π رسم کنید.

به علت وجود x در مخرج خطای تقسیم صفر بر صفر حادث می‌شود، که برای رفع آن می‌توان از روش ضرب بردار منطقی در eps استفاده کرد.

```
%zbz.m
echo off;
x = -4*pi : pi/10 : 4*pi;
y = sin(x)./x;
% Warning: Divide by zero.
x = x + (x == 0)*eps ;
% equivalent to
% x = x + (~x)*eps
% this one is OK:
% x = x + eps
y = sin(x)./x; % No Warning
plot(x,y)

>> zbz
```



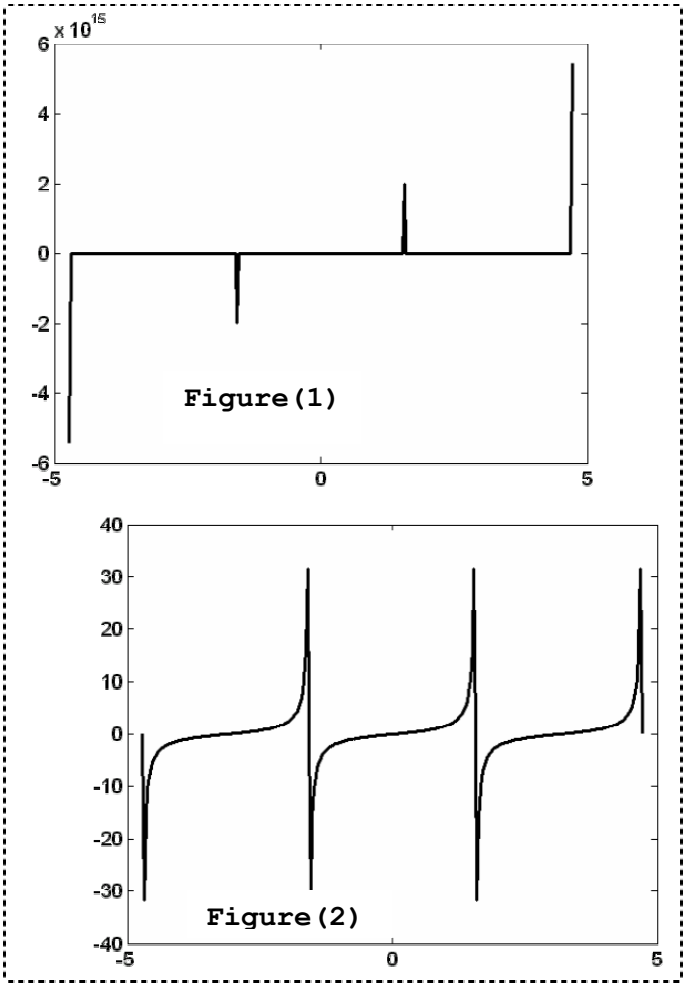
شکل ٦-١

تقسیم بر صفر

در یک برنامه با نام tn.m منحنی y = tan(x) را در فاصله -3π تا 3π رسم کنید. همین منحنی را با استفاده از بردار منطقی در حذف مقادیر بزرگ y مجدداً رسم کنید.

```
%tn.m
echo off;
x = -3*pi/2:pi/100:3*pi/2;
y = tan(x);
figure(1)
plot(x,y)
% results in very large y's
% because of small x's
y = y .* (abs(y) < 1e6);
%removes large y's
figure(2)
plot(x,y)
% good looking graph

>> tn
```

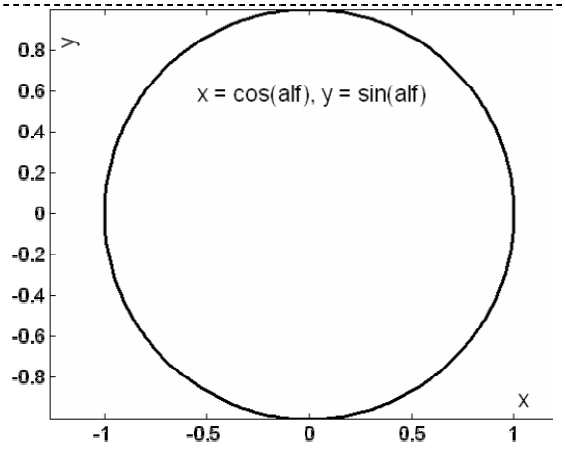


شکل ٦-٢

رسم پارامتریک

با استفاده از روش پارامتریک یک دایره رسم کنید.

```
x1 = 'cos(alf)';
y1 = 'sin(alf)';
ezplot(x1,y1)
```



شکل ۳-۶

تعریف عمل ریاضی برای نوع **single**

اغلب عملیات ریاضی برای نوع single تعریف نشده اند. ما عمل جمع را برای نوع single به صورت یک متد (تابع) تعریف و در دیرکتوری @single (منشعب از دیرکتوری work) قرار می‌دهیم.

```
>> si1 + si2
??? Error using ==> +
Function '+' is not defined for values of class 'single'.
```

```
function F = plas(a,b)
F =  double(a)+double(b);
```

```
>> si1 = 2.2; si2 = 3.5;
>> plas(si1,si2)
ans = 5.7000
```

۳-۶ مثال های آماری

تحلیل آماری نمرات دانشجویان

نمره‌های دانشجویان یک درس را در برداری قرار داده و از لحاظ آماری آنالیز کرده و نتایج را نمایش دهید.

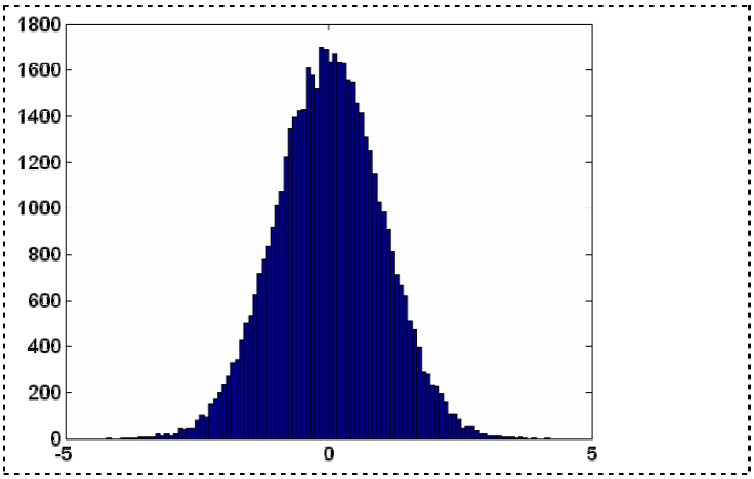
```
% Snum.m
scr = [12 14.56 18.44 16 8.3 19.1 18.23 16.67 5.3 7.8...
          15 12.3 14.6 8.8 17 11.2 13.25 12 13 9 14 ...
          11 12 11.5 15 15 7 4 6 11 12 8 9];
N1 = length(scr);N2 = sum(scr)/N1;N3 = mean(scr);
N4 = max(scr);N5 = min(scr);N6 = std(scr);
S1='Number of Stds';S2 = 'Class Average';
S3 = 'Class Mean';S4 = 'Class Max';
S5 = 'Class Min';S6 = 'Class Std';
fprintf('%-16s %7.3f\n%-16s %7.3f\n%-16s %7.3f\n%-16s %7.3f\n%-16s
      %7.3f\n%-16s %7.3f\n',S1,N1,S2,N2,S3,N3,S4,N4,S5,N5,S6,N6);
```

```
>> Snum
Number of Stds    33.000
Class Average     12.062
Class Mean        12.062
Class Max         19.100
Class Min          4.000
Class Std          3.908
```

پیشینه نگار `hist()`

۵۰۰۰۰ عدد تصادفی با توزیع نرمال ایجاد کرده و کلاً در ۱۰۰ ظرف گنجانده و نمایش دهید (شکل ٦-٤).

```
%hst.m
x = randn(1,50000);
hist(x,100)
>> hst
```



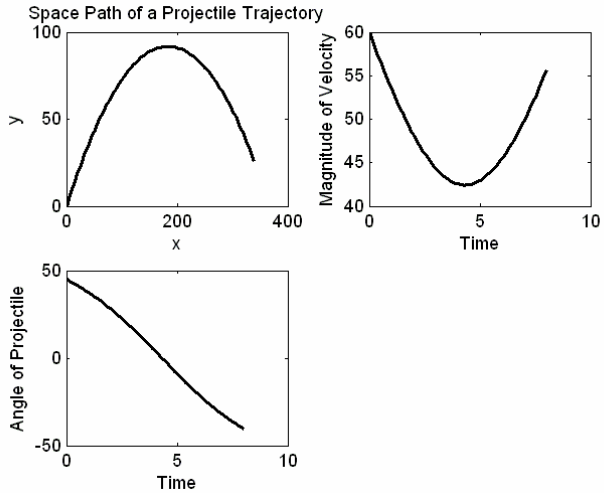
شکل ٦-٤

۶-۴ مثال های مکانیک

آنالیز پرتابه `projectile`

پرتابه‌ای را با زاویه ٤٥ درجه و سرعت اولیه ٦٠ متر بر ثانیه پرتاب می‌کنیم. در فاصله زمانی بین صفر و ۸ ثانیه، مسیر پرتابه در فضا، سرعت پرتابه، و زاویه پرتابه را برحسب زمان رسم کنید (شکل ٦-٥).

```
% projectile.m
echo off;
d0 = 45; v =  60; g = 9.8; % constant values
a = d0 * pi / 180;    % convert to radians
t =  0 : 0.1 : 8;
x = v * t * cos(a);  % horizontal displacement
y = v * t * sin(a) - 0.5 * g * t .^ 2;    % vertical displacement
subplot(2,2,1), plot(x,y),xlabel('x'),ylabel('y')
title('Space Path of a Projectile Trajectory')
vx = v * cos(a);            % horizontal velocity
vy = v * sin(a) - g * t;    % vertical velocity
V = sqrt( vx^2 + vy.^2 );  % Magnitude of velocity
subplot(2,2,2), plot(t,V,'g')
xlabel('Time'), ylabel('Magnitude of Velocity')
d = 180 / pi * atan2( vy, vx );          % angle at time t
subplot(2,2,3), plot(t,d,'r')
xlabel('Time'),ylabel('Angle of Projectile')
>> projectile
```



شکل ٦-٥

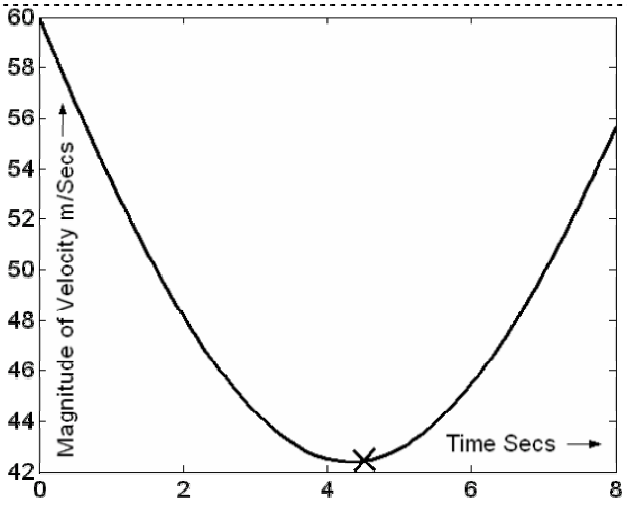
سرعت پرتابه و یافتن مینیمم آن

ام- فایلی با نام trj.m بنویسید که:

الف) بزرگای سرعت یک پرتابه را با نخستینه‌های: زاویه پرتاب d0 = 45، سرعت اولیه v = 60، و شتاب گرانش g = 9.8 در فاصله زمانی t = 0 : 0.1 : 8 رسم کند. روی می‌نیمم سرعت علامت × بزنید.

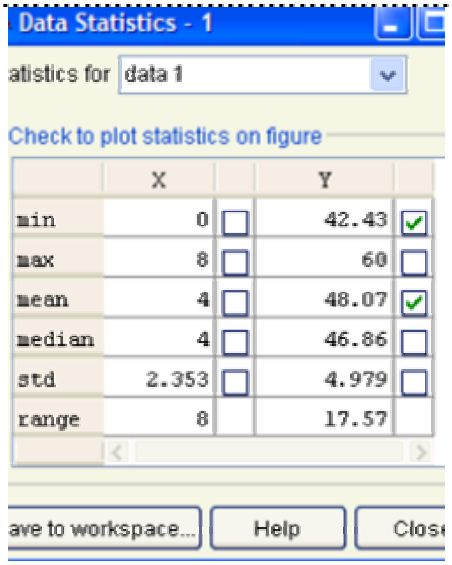
ب) مقادیر مینیمم و میانگین منحنی را با استفاده از پنجره Data Statistics نشان دهید.

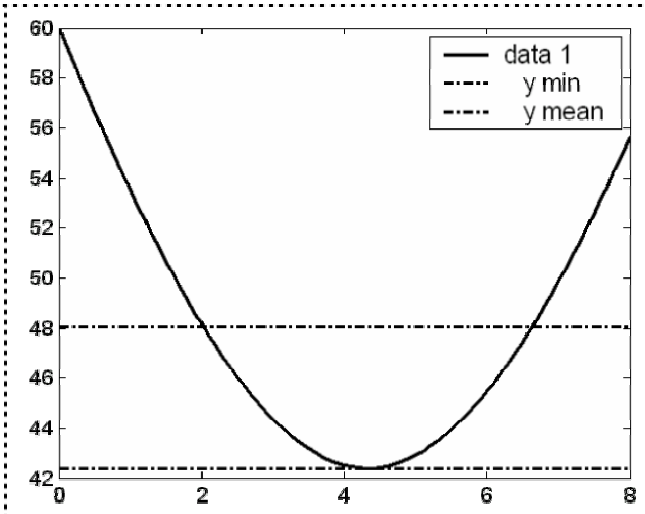
```
% trj.m
v =  60; g = 9.8;
d0 = 45; t = 0 : 0.1 : 8;
a = d0 * pi / 180;
% converts to radians
vx = v * cos(a);
vy = v * sin(a) - g * t;
V = sqrt( vx^2 + vy.^2 );
plot(t,V,'g')
xlabel('Time Secs')
ylabel ...
'Magnitude of Velocity m/s'
k = find(V == min(V));
hold on,
plot(t(k),V(k),'x')
% V(k) is min(V)
hold off
```



شکل ٦-٦

عبارت k = find(V == min(V)) اندیس نقطه می‌نیمم را پیدا می‌کند.





مقادیر کمینه و میانگین خط‌چین شده‌اند.

شکل ٦-٧

٦-٥ مثال های الکتریکی

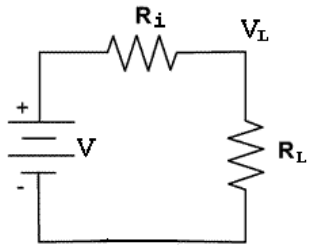
توان مصرفی مقاومت

در شکل ٦-٨ با داشتن k ، توان مصرفی، ولتاژ بار، و توان اتصال کوتاه باطری این‌گونه نوشته می‌شوند(تحقیق کنید):.

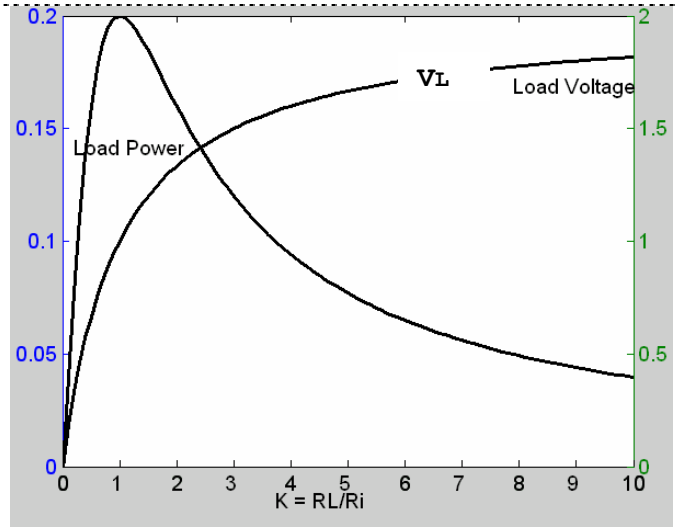
$$k=\frac{R_L}{R_i} \quad , \quad P_L=\frac{V^2}{Ri}\times\frac{k}{(1+k^2)} \quad , \quad V_L=V\times\frac{k}{k+1} \quad , \quad Psc=\frac{V^2}{Ri}$$ توان اتصال کوتاه ثابت است

توان مصرفی و ولتاژ بار را برحسب k (که نمایش مقدار بار است) رسم کنید. V = 12, Ri = 10 (شکل٦-٩).

```
% rp.m
V = 2; Ri = 10;
RL = 0:100;
k = RL./Ri;
PL = ...
=(V^2/Ri)*(k./(1+k.^2));
VL = V*(k./(1+k));
plotyy(k,PL,k,VL),
>> rp
```



شکل ٦-٨



شکل ٦-٩

مدار معادل چند مقاومت موازی

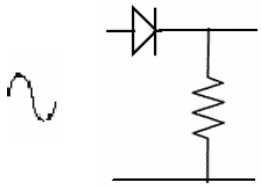
مقاومت معادل سه مقاومت موازی R1 = 15, R2 = 25, R3 = 80 را پیدا کنید.

```
% resi.m
RR = [15 25 80];
n1 = ones(1,3) % creates a vector of three ones
RI = n1./RR; % reverses all three elementds of RR
disp(['RI = ' num2str(RI)]);
SRI = sum(RI); % sums up the reverse elements
RT = 1/SRI; % gives the equivalent resistance of three
disp(['RT = ' num2str(RT)]);
>> resi
n1 = 1.00          1.00          1.00
RI = 0.066667      0.04          0.0125
RT = 8.3916
```

یک سوساز نیمه موج

خروجی یک سوساز نیمه موج را برای ورودی سینوسی بین صفر تا 3π رسم کنید (شکل ٦-١٢)

```
% hf.m
alf = ...
linspace(0, 3*pi);
y = sin(alf);
y = y .* (y > 0);
plot(alf,y)
>> hf
```



شکل٦-١٠

مدار تی T-Network

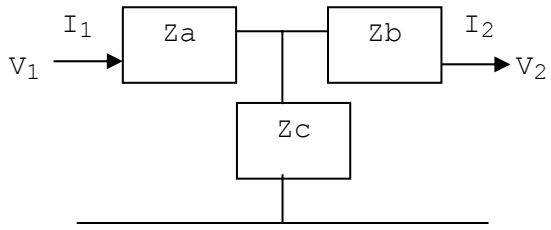
مقدار، زاویه، و فازورِ شدت جریان‌های ورودی و خروجی را برای یک مدار تی با مشخصه‌های زیر به‌دست آورید:

ω = 1000rad/s Ra=3.05K, Ca = 0.01μF, Rb=4.5K, Cb = 0.05μF
Rc = 5.52K, Lc =0.1mH, V1 = 2300, V2 = 450

مدار تی با پارامترهای Z در شکل ٦-١٠ نشان داده شده است. فرم ماتریسی معادلات مدار این گونه است:

$$\begin{vmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{vmatrix} \times \begin{vmatrix} I_1 \\ I_2 \end{vmatrix} = \begin{vmatrix} V_1 \\ V_2 \end{vmatrix}$$

```
Z       *    I    =    V    ,     I = Z \ V
```



شکل ٦-١١

$$V_1 = z_{11}I_1 + z_{12}I_2$$
$$V_2 = z_{21}I_1 + z_{22}I_2$$

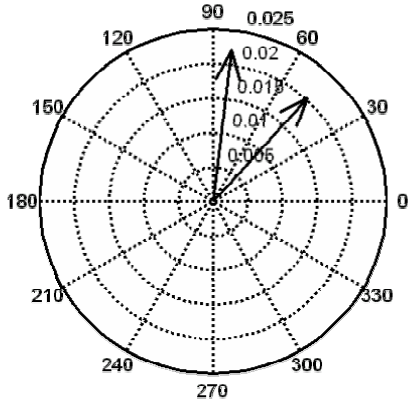
$$z_{11} = V_1/I_1 \;|I_2 = 0$$
$$z_{21} = V_2/I_1 \;|I_2 = 0$$
$$z_{12} = V_1/I_2 \;|I_1 = 0$$
$$z_{22} = V_2/I_2 \;|I_1 = 0$$

$$z_{11} = Za + Zc$$
$$z_{22} = Zb + Zc$$
$$z_{12} = z_{21} = Zc$$

حال برنامه‌ای برای انجام محاسبات با مقادیر داده شده مینویسیم:

```
% Tnet.m
echo off
omeg = 1000;
Ra = 3050; Ca = 0.01e-6;
Za = Ra+1/(omeg*Ca*j);
Rb = 4500; Cb = 0.05e-6;
Zb = Rb+1/(omeg*Cb*j);
Rc = 5520; Lc = 0.0001;
Zc = Rc+omeg*Lc*j;
z11 = Za + Zc;
z22 = Zb + Zc;
z12 = Zc; z21 = Zc;
Z = [z11  z12; z21  z22];
V1 = 2300; V2 = 450;
V = [V1; V2];
I = Z \ V
>> Tnet
>> Iabs = abs(I)'
>> Iang = angle(I)'
>> compass(I)
```

```
I = 0.0027 + 0.0220i
    0.0136 + 0.0150i
Iabs = 0.0222  0.0202
Iang = 1.4482  0.8339
```

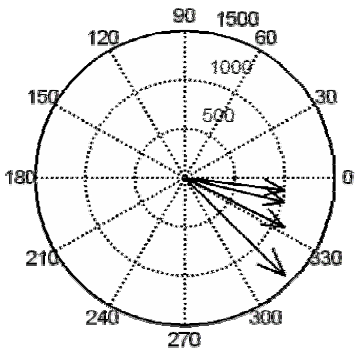


شکل ٦-١٢

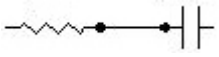
امپدانس معادل مدار RC

در مدار زیر اندازه خازن هر بار دوبرابر می‌شود. امپدانس معادل‌ها را در فرکانس زاویه‌ای $\omega = 100$ rad/s همراه با نمایش بردار- هندسی‌ها آن مقایسه کنید.

```
% rc.m
omeg = 100;
R = repmat(1000,1,4); c = [10e-6 20e-6 40e-6 80e-6];
z = R + 1./(j*omeg*c);
compass(z)
disp('   C/micF   Angle/Deg   Z/Kohm    R/Kohm');
disp([1e6*c'  (180*angle(z)/(2*pi))' (abs(z)/1e3)' R'/1e3]);
>> rc
```



C/micF	Angle/Deg	Z/Kohm	R/Kohm
10.0000	-22.5000	1.4142	1.0000
20.0000	-13.2825	1.1180	1.0000
40.0000	-7.0181	1.0308	1.0000
80.0000	-3.5625	1.0078	1.0000

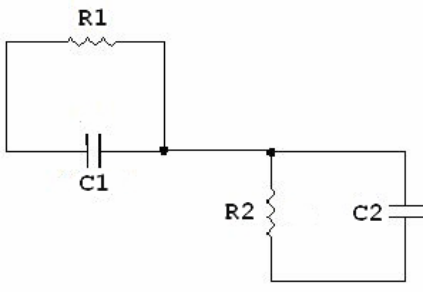


شکل ٦-١٣

ترسیم مشخصه پراب اسیلوسکوپ، دستور **semilogx()**

مدار زیر، پراب یک اسکوپ را برای اندازه‌گیری فرکانس‌های پائین نشان می‌دهد، امپدانس وردی بر حسب مگااهم و ضریب تبدیل امپدانس آن را در بازه فرکانسی 20Hz تا 20KHz با گام 20Hz رسم کنید.

R1=9MΩ C1=100pF
R2=1MΩ C2=100pF



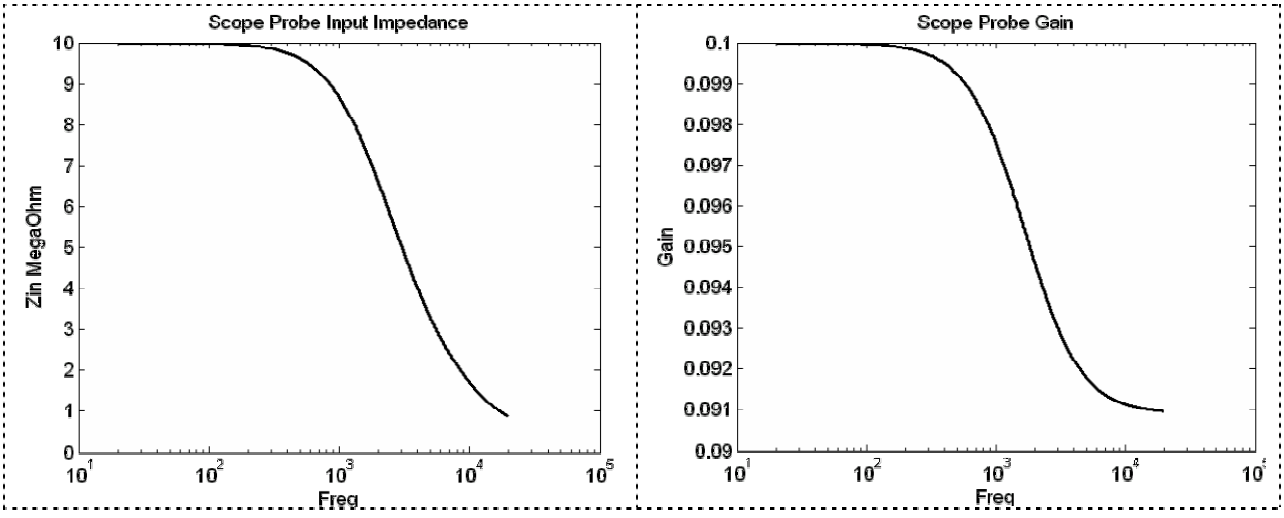
$$Z_{in}=\left[\frac{1}{j\omega C_1}||R_1\right]+\left[\frac{1}{j\omega C_2}||R_2\right]=\frac{R_1}{1+j\omega C_1R_1}+\frac{R_2}{1+j\omega C_2R_2}$$

شکل ٦-١٤

```
% ProbeGain.m
c = [10e-12  100e-12];
r = [9e6  1e6];
freq = [20:20:20000];
omg = 2*pi*freq;
z1 = r(1)./(1+j*omg*c(1)*r(1));
z2 = r(2)./(1+j*omg*c(2)*r(2));
zin = z1 + z2;
figure(1)
semilogx(freq,abs(zin)/1e6)
xlabel('Freq')
ylabel('Zin MegaOhm')
title('Scope Probe Input Impedance')
figure(2)
gain = z2 ./ zin  ;
```

کاربرد علامت / (بدون نقطه) ماتریس zin را معکوس کرده، و به نتیجه غلط منجر می‌شود. %

```
semilogx(freq,abs(gain))
xlabel('Freq')
ylabel('Gain')
title('Scope Probe Gain')
>> ProbeGain
```



شکل ٦-١٥

۶-۶ تمرین

برنامه‌های زیر را در ام- فایل‌های جداگانه بنویسید:

۱- در مثال مدار تی همان مقادیر را برای یک مدار پی در نظر گرفته با استفاده از پارامترهای Y مدار را حل کنید.

۲- در مثال پراب اسکوپ، زاویه تبدیل مدار را با مقیاس درجه بر حسب لگاریتم فرکانس رسم کند. برچسب‌های ضروری را در گراف بگنجانید.

۳- x و y مثال دایره مثلثاتی را قسمت‌های حقیقی و موهومی یک تابع مختلط بگیرید zc = x + i*y . قسمت موهومی را بر حسب قسمت حقیقی رسم کنید.

٤- r را برداری چرخان و متساوی‌العنصر با tet بگیرید (r = repmat(1,1,length(tet)) ، ترسیم را به طریق قطبی polar(r,theta) انجام دهید.

۵- درجه حرارت سحر، صبح، ظهر، غروب، عصر، و شبِ هرروز از یک هفته را در یک ماتریس ۷ × ٦ وارد کرده و گراف‌های ستونی و منحنی مربوطه را رسم کنید.

٦- یک‌بار ۲۰ و یک‌بار ۵۰۰ عدد تصادفی از هر دو نوع rand(), randn() تولید کنید. میانگین mean() ماکزیمم، مینیمم، و انحراف میانه std() اعداد تصادفی از هر دو نوع را پیدا کنید. با استفاده از بردار منطقی تعداد اعداد بالا و پائین میانگین را شمارش کنید.

راهنما: تابع randn(1,n)، n عدد تصادفی با توزیع نرمال (گوسی) حول 0 (میانگین 0) تولید می‌کند، با افزایش تعداد اعداد تصادفی توزیع به نرمال نزدیک‌تر می‌شود.

۷- قسط پرداخت وام خانه از این فرمول به دست می‌آید. $$P=\frac{rL(1+r/12)^{12N}}{12[(1+r/12)^{12N}-1]}$$

مبلغ L = 10e6 وام با سود r = 0.15 سالانه در مدت N = 5:20 سال بازپرداخت می‌شود، قسط ماهانه را برای زمان بازپرداخت بین ۵ تا ۲۰ سال به صورت جدول نشان دهید و به صورت نقطه به نقطه (نه خط پیوسته)، و نمودار ستونی رسم کنید است. مبلغ کل بازپرداخت را بر حسب N به صورت جدول نشان دهید و به صورت نقطه‌ای، خط‌چین، ستاره‌ای، و نمودار ستونی Bar Chart رسم کنید است.

فصل ۷ گرافیک سه بُعدی

۷-۱ ترسیم منحنی فضائی

تابع plot3()

این تابع یک منحنی دو بعدی را به بعد سوم می‌برد، در نتیجه یک منحنی فضائی (سه‌بعدی) رسم می‌شود (مثلاً یک دایره به صورت مارپیچ درمی‌آید). عبارت `plot3(x1,y1,z1)` نقاطی را با مختصات `x1,y1,z1` با خط منحنی به هم وصل می‌کند. سه بردار می‌توانند به نحو جبری به هم وابسته بوده یا کاملاً مستقل از یک‌دیگر باشند، اما تعداد عناصر بردارهای `x1,y1,z1` باید برابر باشند.

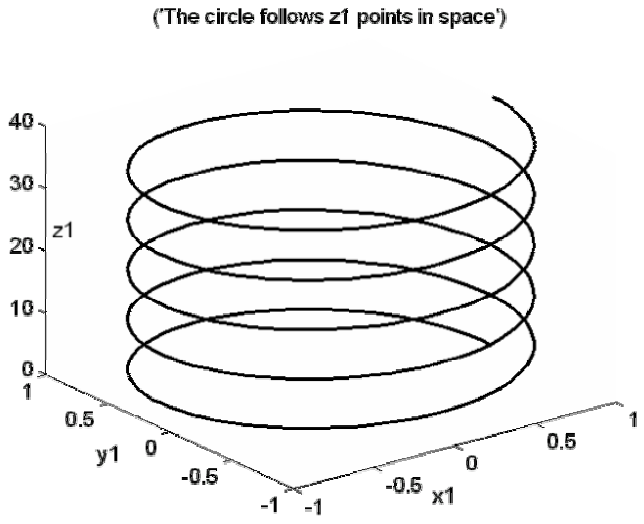
مثال:

یک دایره با روش غیرمستقیم تولید و آنرا در جهت عمودی حرکت داده و یک مارپیچ استوانه‌ای ایجاد کنید.

دراین مسأله `x,y` باید تابع دایره‌ای یک‌دیگر باشند، اما `z` مستقل ازآن دو بوده و صرفاً در جهت عمودی حرکت می‌کند.

```
% p3.m
echo off;
alf = linspace(0,10*pi,400);
x1 = cos(alf);
y1 = sin(alf);
z1 = linspace(0,40,400);
plot3(x1,y1,z1),
title ...
('The circle follows
 z1 points in space')

>> p3
```



شکل ۷-۱

منحنی فضائی نتیجه اتصال نقاطی با مختصات `x1,y1,z1`

تابع comet3()

تابع `comet3()` همان کار `plot3()` را با پویا نمائی انجام می‌دهد. مثال فوق را با `comet3(x1,y1,z1)` اجرا کنید.

۷-۲ ترسیم سطوح فضائی

دستور meshgrid(a,b)

اگر `a` و `b` به‌ترتیب بردارهای `n` و `m` عنصری باشند. عبارت `[X,Y] = meshgrid(a,b)` ماتریس `X` را با ردیف‌هائی مساوی `a` در `m` ردیف و ماتریس `Y` را با ستون‌هائی مساوی `b` در `n` ستون می‌سازد. در نتیجه `X` و `Y` دو ماتریس هم‌سان خواهند بود. دستور `[X,Y] = meshgrid(a)` است و `X` و `Y` را ترانهاد می‌سازد.

مثال:

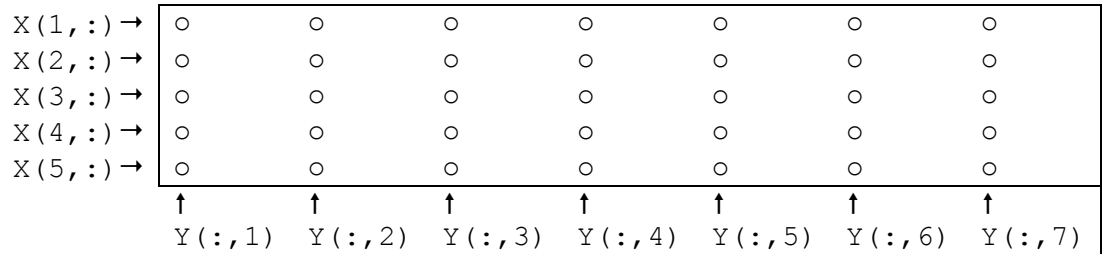
```
>> x = -3:3 ;
>> y = -2:2 ;
>> [X,Y] = meshgrid(x,y)
X =   -3    -2    -1     0     1     2     3
```

```
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3

Y =  -2     -2     -2     -2     -2     -2     -2
     -1     -1     -1     -1     -1     -1     -1
      0      0      0      0      0      0      0
      1      1      1      1      1      1      1
      2      2      2      2      2      2      2
```

صفحه مختصات

چنان‌چه در مثال فوق مشاهده می‌شود، ردیف‌های ماتریس X پنج خط افقی هفت نقطه‌ای در صفحه X-Y ایجاد می‌کنند. ستون‌های ماتریس Y هفت خط عمودی پنج نقطه‌ای در صفحه X-Y ایجاد می‌کند. از برخورد این خطوط مطابق شکل زیر سی‌وپنج نقطه حاصل می‌شود که نقاط صفحه X-Y را تشکیل می‌دهند (دقت کنید که تعداد عناصر هر یک از ماتریس‌های X و Y نیز سی‌وپنج است). اگر مقادیر و تعداد عناصر بردارهای x و y را تغییر دهیم، تعداد نقاطی را که دستور [X,Y] = meshgrid(x,y)، برروی صفحه ایجاد می‌کند تغییر خواهد کرد.



مثال:

بردار a = -3:3 را تعریف، و با آن صفحه X-Y را پدید آورید. تعداد نقاطِ صفحه X-Y چندتا است؟

حل:

```
>> a = -3:3 ;
>> [X,Y] = meshgrid(a)
```

```
X = -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
    -3     -2     -1      0      1      2      3
Y = -3     -3     -3     -3     -3     -3     -3
    -2     -2     -2     -2     -2     -2     -2
    -1     -1     -1     -1     -1     -1     -1
     0      0      0      0      0      0      0
     1      1      1      1      1      1      1
     2      2      2      2      2      2      2
     3      3      3      3      3      3      3
```

نقاط حاصل برروی صفحه X-Y هفت‌درهفت یعنی چهل‌ونُه است که برابر با تعداد عنصر هریک از ماتریس‌ها می‌باشد.

دستور mesh(X,Y,Z)

چنان‌چه یک تابع فضائی از X و Y مانند Z داشته باشیم، دستور mesh(X,Y,Z) نقاط فضائی با مختصات X,Y,Z پدید می‌آورد و آن‌ها را طوری به‌هم وصل می‌کند که یک شکل فضائی پدید آید. مختصات سه‌گانه هرنقطه‌ی این شکل

عناصر متناظر از ماتریس‌های فوق می‌باشند (هر سه ماتریس باید هم‌سان باشند). ماتریس‌های X و Y که متغیرهای مستقل هستند می‌توانند با دستور meshgrid () یا به طریق دیگری ایجاد شوند، اما Z باید تابع X,Y باشند.

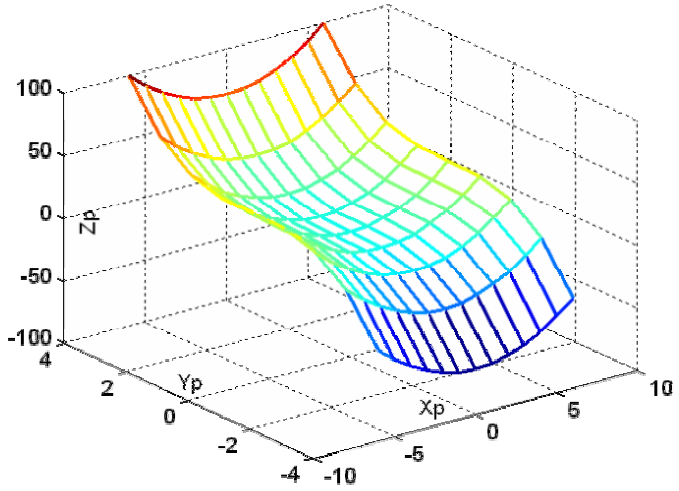
مثال:

با بردارهای a = -6:6 و b =-4:4 ماتریس‌های Xp و Yp را بسازید.

تابع فضائی Zp = Xp.^2 + Yp.^3 را ایجاد کرده، سپس با دستور mesh () صفحه‌ای پدید آورید که مختصات سه‌گانه نقاط آن عناصر متناظر از ماتریس‌های Zp و Xp,Yp مثال فوق باشد.

```
% ThreeD.m
echo off;
[Xp,Yp] = ...
meshgrid(-6:6, -4:4);
Zp = Xp.^2 + Yp.^3 ;
mesh(Xp,Yp,Zp)
xlabel('Xp')
ylabel('Yp')
zlabel('Zp')

>> ThreeD
```



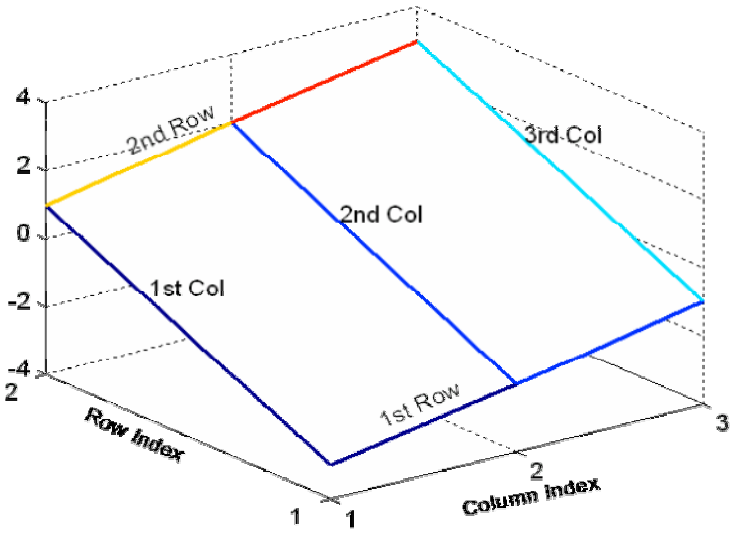
شکل ۷-۲

دستور mesh(M) ، نمایش سه بعدی یک ماتریس

دستور mesh (M) اندیس‌های ستون و ردیف M را به‌ترتیب روی محورهای x و y ، و مقادیر عناصر را بالای مختصات مسطح هر عنصر (x,y) روی محور z می‌برد.

مثال:

```
>> M = [-3 -2 -1;
        1  2  3];
>> mesh(M)
```



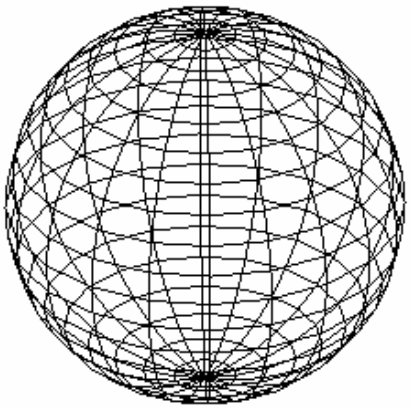
شکل ۷-۳

۷-۳ توابع فضائی کتاب خانه ای

رسم کُره با sphere و ایجاد افکت های تصویری

عبارت `[X,Y,Z] = sphere` مختصات فضائی یک کره را داخل سه ماتریس قرار می‌دهد. در زیر یک کره را رسم، و بعضی افکت‌های تصویری را روی گراف ایجاد کرده‌ایم:

```
% sp
[X,Y,Z] = sphere;
mesh(X,Y,Z)
axis equal % گراف را متقارن می‌کند.
axis off % محورها را محو می‌کند.
hidden off
% قسمت‌های مخفی گراف را ظاهر می‌کند.
>> sp
```



شکل ۷-٤

رسم استوانه با cylinder

عبارت `[X,Y,Z] = cylinder` مختصات فضائی یک استوانه را داخل سه ماتریس قرار می‌دهد. که با استفاده از آن‌ها می‌توان استوانه را با شگردهای تصویری مختلف ایجاد کرد.

رسم قله ها با تابع نمونه peaks

تابع `peaks` یکی از توابع نمونه MATLAB است. که سطحی را با تعدادی قُلّه رسم می‌کند.

با `type` هریک از توابع فضائی کتاب‌خانه‌ای، روتین برنامه‌های آن‌ها را مشاهده کنید.

۷-۴ بعضی از قابلیت های گرافیکی متلب

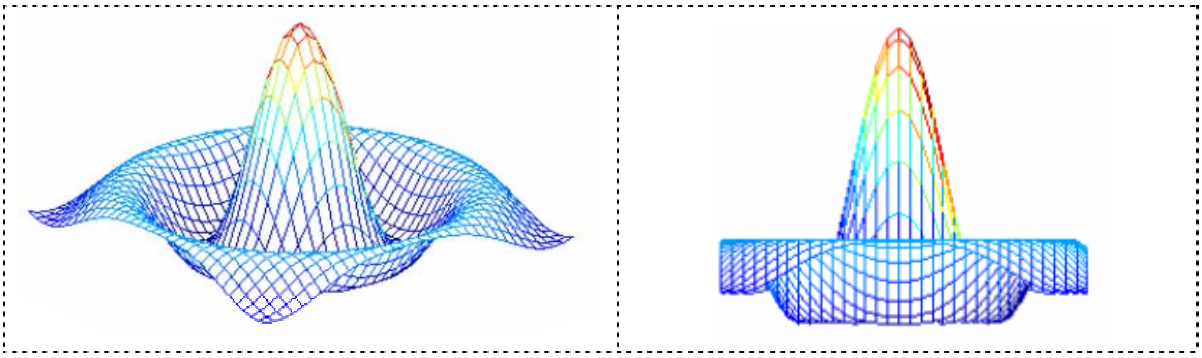
زاویه دید یک تصویر

عبارت `[a b] = view` زاویه دید فعلی ترسیم را می‌دهد. عدد a زاویه چرخش افقی و عدد b زاویه چرخش عمودی شکل را نسبت به دید مستقیم از روبرو می‌دهند. مقادیر پیش فرض زاویه دید به هنگام رسم هر گراف `-37.5` درجه چرخش افقی (در خلاف جهت عقربه‌های ساعت) و چرخش عمودی به اندزه `30+` درجه (درجهت عقربه‌های ساعت) است. دستور `view(a,b)` زاویه دید را دستی تغییر می‌دهد. عدد a زاویه چرخش افقی جدید و عدد b زاویه چرخش عمودی جدید شکل را ایجاد می‌کنند. با انتخاب‌هائی از منیوی `Tools` زاویه دید را با ماوس تغییر دهید

مثال:

کلاه مکزیکی متلب را رسم کنید. زاویه چرخش افقی و عمودی شکل را پیدا کرده، سپس زاویه را به دید روبرو ببرید.

```
% mex.m
[x y ] = meshgrid(-8 : 0.5 : 8.5);
r = sqrt(x.^2 + y.^2) + eps;
z = sin(r) ./ r;
mesh(z)
[a b] = view
view(0,0)
>> mex
a = -37.5000   b = 30
```



شکل ۷-۵

تصویربرداری با getframe

با دستور getframe می‌توان از هر یک از زوایای گراف یک عکس گرفت.

مثال:

تصویر کلاه مکزیکی را ایجادکرده، برنامه زیر را برای آن اجرا کنید.

```
% mexmv.m
[x y ] = meshgrid(-8 : 0.5 : 8.5);
r = sqrt(x.^2 + y.^2) + eps;
z = sin(r) ./ r;
mesh(z)
grid off
axis off
hidden off
for k = 1: 150
 view(-37.5 + k,30 - k)
 getframe;
end
>> mexmv
```

هم‌چنین می‌توان مجموعه فریم‌ها را داخل یک ماتریس ریخت، و در صورت لزوم با دستور save آن را روی دیسک نگه‌داری کرد.

مثال:

تابع داخلی peaks را به حرکت درآورده و فیلم آن را در ماتریسی به‌نام mi نگه‌دارید.

```
% pk.m
peaks
axis off, grid off
for k = 1:20
 view(-37.5+30*k,30+30*k);
 mi(k) = getframe; % mi(:,k) and mi(k,:) are also valid
end
>> pk
```

باز نمایش فیلم بادستور ()movie

اگر متغیر mi در حافظه مانده باشد، یا از دیسک به حافظه متلب بار گذاری شده باشد، تصاویر نگه‌داری شده در آن با دستور movie(mi) به نمایش مجدد درمی‌آید.

مثال:

برای مثال فوق قاب‌های تصویر را در یک فایل mat. ضبط کرده، و در اجرای بعدی متلب به نمایش درآورید.

```
% Ldm.m
clear x y r z k % removes all but mi for safety
```

```
delete mi.mat % if already exists in the current directory
save mymov mi % mi variable is saved into mymov.mat file
clear mi % removes mi
>> Ldm
```

در اجرای بعدی متلب:

```
>> load mymov
>> movie(mi)
```

۷-۵ تغییرات روی قسمتی از سطح ترسیم سه بعدی

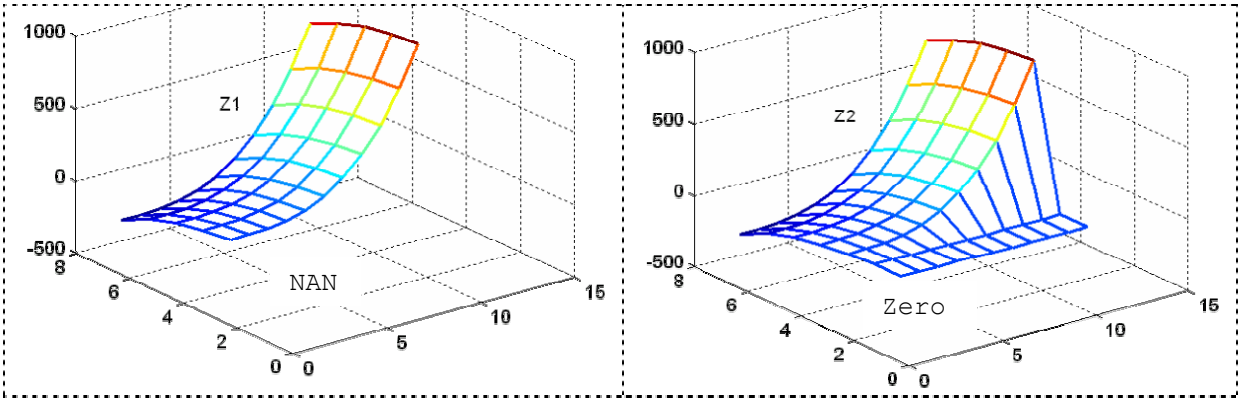
حذف قسمتی از سطح

این کار با استفاده از nan ، ضرب در صفر، و ماتریس تهی انجام می‌شود. هریک از روش‌ها به نوعی عمل حذف را انجام می‌دهند.

مثال:

در یک ام- فایل تابع فضائی $z = x^3 - y^3$ را رسم و سپس قسمتی از آن را حذف کرده، نتیجه را رسم کنید.

```
% nn.m
a = 0:10;
b = 0:6;
[x y] =  meshgrid(a,b);
z = (x.^3 - y.^3);
mesh(z)
z1 = [nan*z(1:2,:); z(3:size(z,1),:)];
figure(2)
mesh(z1), title 'Z1'
z2 = [0*z(1:2,:); z(3:size(z,1),:)];
figure(3)
mesh(z2), title 'Z2'
z(1:2,:)=[]
figure(4)
mesh(z)>> nn
```



شکل ۷-۶

۷-۶ سایر دستور های ترسیم سه بعدی

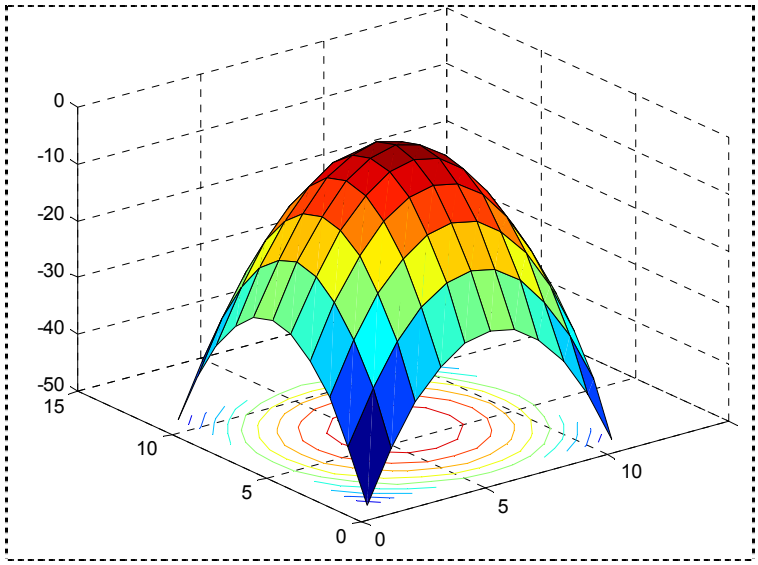
رسم سطح surf () شبیه mesh () است، اما سطح را به صورت شطرنجی رسم می‌کند.

اگر برآمدگی مرتفعی را از بالا به طور عمودی نگاه کنیم، ارتفاع سطوح مختلف آن را می‌توان به صورت دوایری متحدالمرکز نشان داد. این دوایر در اصطلاح جغرافیائی کنتور نام دارند. دستور contour () این ترسیم را به‌دست می‌دهد (امتحان کنید). دستورهای surfc () , meshc () حجم را همراه با کنتور در زیر آن نمایش می‌دهد.

مثال:

رابطه مثال فوق را هم‌راه با کنتور در زیر رسم کنید.

```
>> a = -5:5;
>> [x y] = meshgrid(a);
>> z = (-x.^2 - y.^2);
>> surfc(z)
```



شکل ۷-۷

۷-۷ تمرین

۱- دایره‌ای مسطح ایجاد کنید. z1 را تابعی از x1,y1 یا هردو بگیرید (مثلاً z1 = x1.^2). گونه‌گونی‌های منحنی را با تغییر جای x1,y1,z1 رسم کنید.

راهنما: plot3(x1,y1,z1),plot3(x1,z1,y1),...

۲- با استفاده از فرمول‌های x2 = exp(-0.03*alf).*cos(alf) و y2 = exp(-0.03*alf).*sin(alf) یک مارپیچ مخروطی فضائی ایجاد کنید. سپس زاویه دید را با استفاده از انتخاب Tools_Rotate 3D از منوی گراف تغییر دهید. پس از هر بار تغییر مقدار زوایا را با استفاده از دستور [a b] = view روی پنجره فرمان مشاهده کنید.

۳- ماتریس M را بسازید و دستورهای mesh(M) و mesh(X,Y,Z) را اجرا کنید.

٤- با استفاده از v = -5:5 ماتریس‌های Xp و Yp و سپس Zp را (به صورت توابع چند جمله‌ای از Xp و Yp) تعریف کنید.Zp را یک‌بار همراه با این دو ماتریس و یک‌بار به تنهائی mesh کنید.

٥- ماتریس Zp = Xp.^2 + Yp.^3 را با مقدار دهی مناسب mesh(Zp) کنید. اندیس ردیف‌ها (9) و اندیس ستون‌های (13) آن را روی محورها مشخص کنید.

٦- معادل موازی دو مقاومت [R1 R2]= meshgrid(10:10:100) را به صورت سطح رسم کنید.

۷- با دستور sphere یک کُره ایجاد کرده، آن را به چرخش درآورده، و فیلمی ازآن تهیه نمائید.

۸- ماتریس‌های مختصات استوانه‌ای را ایجاد کرده و رسم کنید. سپس با فرمول‌هائی نظیر [X,Y,Z] = cylinder(cos(t)); شگردهای تصویری بسازید. از axis square استفاده کنید.

۹- سطح peaks را رسم کنید. سپس axis off; hidden off را اجرا کنید.

فصل ۸ ساختارهای تصمیم و تکرار

۸-۱ ساختارهای تصمیم و عوامل آن

عملگرهای نسبتی (رابطه ای) Relational Operators

علامت	==	~=	>	<	>=	<=
عمل	تساوی شرطی	نامساوی شرطی	بزرگ‌تر	کوچک‌تر	بزرگ‌تر یا مساوی	کوچک‌تر یا مساوی

عمل‌گرهای نسبتی (رابطه ای) براساس جدول فوق مقایسه مابین دو آرایه انجام می‌دهند. نتیجه هر مقایسه 0 یا 1 منطقی است. این‌گونه عملیات معمولاً در دستورهای شرطی نظیر if کاربرد دارند.

مثال:

```
>> A = [-2 0 2 4];
>> B = [-2 0 1 3];
>> A == B
```

```
ans = 1     1     0     0
```

عملگرهای منطقی Logical Operators

علامت	&	|	~
عمل	AND	OR	NOT

عمل‌گرهای منطقی مابین صفر و یک‌های منطقی عمل می‌کنند، و نتیجه آن‌ها 0 یا 1 منطقی است. ترکیب عملیات مقایسه‌ای و منطقی معمولاً در دستورهای شرطی نظیر if کاربرد دارند.

مثال:

```
>> CL = (A == B)
>> (A < B) & CL
>> (A < B)|CL
```

```
CL  = 1     1     0     0
ans = 0     0     0     0
ans = 1     1     0     0
```

بلوک if

فرم کلی بلوک if در زیر آمده است. اگر نتیجه شرط condition منطق یک (درستی) بود دستورات زیر if یا دستورات زیر elseif اجرا می‌شوند، اگر نتیجه شرط منطق صفر (نادرستی) بود دستورات زیر if یا دستورات زیر elseif اجرا نمی‌شوند. در صورت درستی چند شرط دستورات زیر اولین شرط درست اجرا شده و برنامه پس از خروج از بلوک if از اولین دستور بعد از end ادامه می‌یابد. اگر نتیجه همه شرط‌ها منطق صفر (نادرستی) بود دستورات زیر else اجرا می‌شوند. یک بلوک if می‌تواند elseif ها و/ یا else را نداشته باشد.

```
if condition1
   statements1
elseif condition2
   statements2
      .
      .
else
   statementsN
end
```

بلوک switch

فرم کلی switch در زیر آمده است. در مقابل switch نام یک متغیر تک عنصری به نام متغیر مبنا می‌آید. در مقابل هر case یک مقدار یا چند مقدار (داخل آکلاد) به متغیر مبنا نسبت داده می‌شود، و در صورت صدق آن مقدار (یا یکی از چند مقدار) دستورات زیر همان case اجرا می‌شوند.

```
var = ...
switch var
case value of var
 statements1
case {values of var}
 statements2
otherwise
 statements3
end
```

مثال:

در یک ام- فایل عددی تصادفی بین صفر و نُه تولید و زوج، فرد یا صفر بودن آن را تعیین کنید.

```
% sw.m
d = floor(10*rand);
disp(d);
switch d
    case 0
        disp('Zero');
    case {1,3,5,7,9}
        disp('Odd');
    otherwise
        disp('Even');
end
>> sw
>> sw
>> sw
```

```
2          Even
9          Odd
0          Zero
```

۸-۲ ساختارهای تکرار

حلقه for

در این حلقه شمارنده‌ای index از مقدار اولیه تا مقدار نهائی خود را با گام معین طی می‌کند، و حلقه برای هر گام تغییرِ شمارنده تکرار می‌شود. می‌توان شمارنده را مساوی یک بردار قرار داد.

مثال ها:

جدول توان

```
% fr.m
d = 1:2:9;
for k = d
    disp(k^2)
end
>> fr
```

```
1     9     25     49     81
```

جدول سینوس، کسینوس

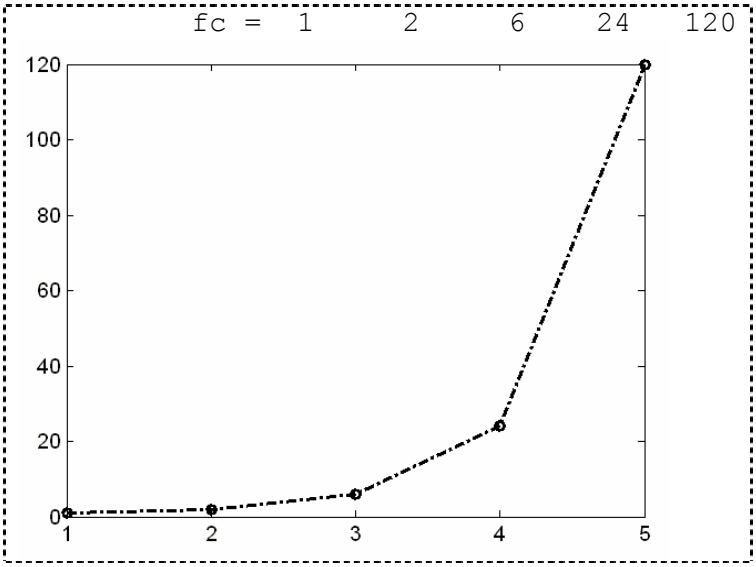
```
% sn.m
v = [0 : pi/6 : pi];
fprintf('\n');
disp(['Angle    Sine  Cosine'])
for k = v
    fprintf('%6.2f %6.2f %6.2f\n',180*k/pi, sin(k), cos(k));
end
```

```
>> sn
Angle    Sine  Cosine
  0.00   0.00   1.00
 30.00   0.50   0.87
 60.00   0.87   0.50
 90.00   1.00   0.00
120.00   0.87  -0.50
150.00   0.50  -0.87
180.00   0.00  -1.00
```

تعییین فاکتوریل عدد صحیح مثبت، اعداد ۱ تا ۵

```
% fac.m
clear
 % متغیرها را از حافظه پاک می‌کند
n = 5; fact = 1;
ss = 1;
for k = ss : n
    fact = k*fact;
    fc(k) = fact;
end
fc
plot([ss:n],fc,'Or')
>> fac
```

```
fc =   1     2     6    24   120
```



شکل ۸-۱

بلوک **try...catch**

مثال:

اعضای یک ماتریس سلولی را جداگانه چاپ کنید. وقتی به آخر رسید برنامه تمام وپیغام Not Found دیده شود.

```
%TryCth.m
echo off
n=1:10;
A={'Number' 234 'next'  9} ;
for k=n
 try
     disp(A(k))
 catch
 disp('Not Found'), break
 end
end
>> TryCth.m
```

به دست آوردن طول زمان محاسبه حاصل‌جمع اعداد بین صفر تا یک بیلیون با گام صد

```
t0 = clock;
s1 = 0;
for n = 0 : 100 : 1e9   % n یک شمارنده است با مقدار نهائی یک بیلیون با گام صد
   s1 = s1 + n ;
end
elaps1 = num2str(etime(clock,t0));
disp( ['For loop duration =  ', elaps1, ' Secs'])
disp(['sum1 = ' num2str(s1)])
```

تابع etime () فاصله زمانی بین آرگومان‌هایش را می‌دهد، برای آوردن در disp به رشته تبدیل شده است.

بردار به جای حلقه for

مثال فوق را بدون استفاده از for حل کنید. طول زمان این دو روش را برای کامپیوتر خودتان مقایسه کنید.

```
t0 = clock;
m = 0 : 100 : 1e9;
s2 = sum(m);
elaps2 = num2str(etime(clock,t0));
disp( ['Vectorization duration =  ', elaps2 ', Secs'])
disp(['sum2 = ' num2str(s2)])
```

در بعضی موارد استفاده از حلقه for الزامی است و نمی‌توان به جای حلقه for از بردار استفاده کرد.

حلقه while و دستور break

حلقه while مانند for دستورات داخل حلقه را تکرار می‌کند، اما شمارنده ندارد. دستورات داخل حلقه while تازمانی‌که شرط مقابل کلمه while برقرار باشد اجرا خواهند شد.

مثال:

تعیین اعداد اول

یک M-File به‌نام primen.m بنویسید که اعداد اول کوچک‌تر از n (n >= 5) را به دست آورد.

این مثال کاربردهای عبارات و توابع زیر را نیز توضیح می‌دهد. به توضیحات توجه شود:

input(), fprintf(), while, for, if, break, end, rem()

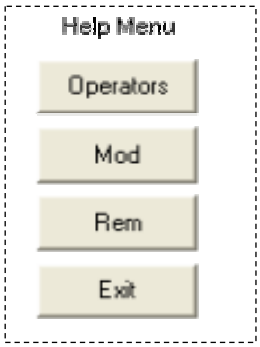
```
% primen.m
n = input('Enter upper limit: ');
fprintf('%d %d',2,3); %    برنامه، ۲ و ۳ را محاسبه نمی‌کند لذا آن‌ها را جدا چاپ می‌کنیم
ii = 5;
   از ۵ شروع می‌کنیم و تقسیم پذیری ii بر کلیه اعداد ماقبلش را امتحان می‌کنیم تا نهایتاً ii به n برسد %
while ii < n
 for jj = 2 : ii-1
  if rem(ii,jj)== 0 %
     break
                          اگر تقسیم پذیر بود عدد، اول نیست و برنامه به عبارت ii=ii+1 می‌رود %
  end
  if jj == fix(sqrt(ii))
                اگر jj تا ریشه دوم ii بالا بیاید و مقسوم‌علیه آن نباشد، عدد، اول است و چاپ می‌شود %
     fprintf(' %d',ii);
     break
   end
 end
  ii = ii + 2; % excludes odd numbers
end
>> primen
enter upper limit: 50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

۳-۸ منیو

منیو یک GUI ساده است که امکان انتخاب چند گزینه را فراهم می‌کند.

مثال:

```
% mnu.m
k = 0;
while k < 4;
    k = menu('Help Menu','Operators','Mod','Rem','Exit');
    if k == 1
        help \
    elseif k == 2
        help mod
    elseif k == 3
        help rem
    else
    a = input('Really Exit? (Y/N) ','s');
        if (a == 'n') || (a =='N')
            k = 3;
             continue;
```



اجرا را به ابتدای حلقه while منتقل و شرط را مجدد تست می‌کند %

```
        end
    end
end
>> mnu
```

۸-۴ تمرین

۱- برنامه‌ای بنویسید که یک عدد را با input() دریافت، و تعیین کند که اول است یا نه.

۲- فاکتوریل ۵ را با استفاده از تابع prod() که حاصل‌ضرب پشت‌هم عناصر یک ماتریس را حساب می‌کند به‌دست آورید.

۳- برنامه‌ای بنویسید که ۱۰ عدد تصادفی بین صفر و ۹ تولید و زوج، فرد یا صفر بودن هر یک را پس از تولید تعیین کند.

راهنما: فرمول d = floor(10*rand) و switch d را داخل یک حلقه for قرار دهید.

٤- یک منیو بنویسید که با انتخاب هر دکمه یک تابع مثلثاتی را رسم کند.

فصل ۹ نکاتی پیرامون توابع و تابع کاربر- تعریف

۹-۱ تابع خط فرمان inline function

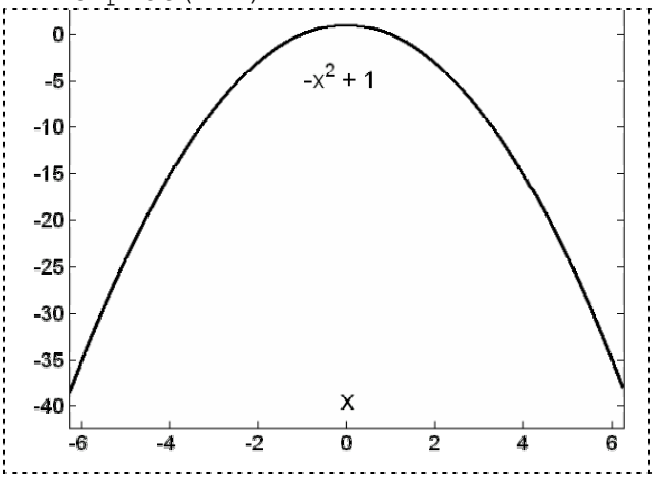
مثال:

تابع `-x^2 + 1` را به صورت `inline` تعریف و بدون مقدار دهی به متغیر آن را رسم کنید.

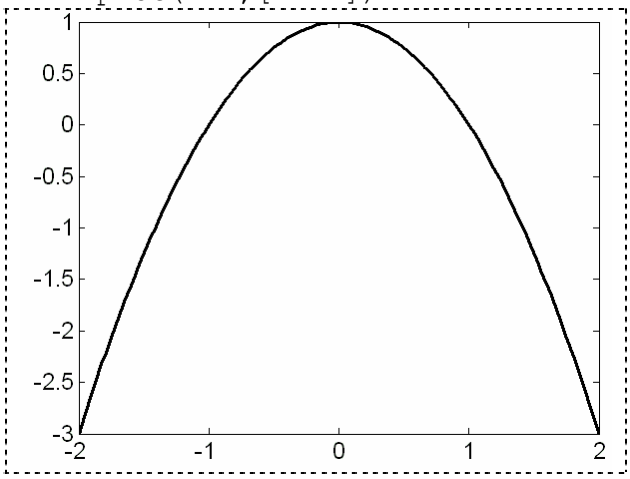
این تابع بایستی به صورت آرگومان تابعی به `ezplot()` ارسال شود. چون بدون مقدار دهی رسم می‌شود، لذا دامنه پیش‌فرضِ $-2\pi, +2\pi$ برای `x` در نظر گرفته می‌شود. عنوان گراف و برچسبِ محور، اتوماتیک گذاشته می‌شوند.

در `fplot()` باید دامنه `x` را تعیین کرد و عنوان و برچسب گذاشته نمی‌شوند.

```
>> fin = inline('-x^2 + 1')
fin = inline function:
      fin(x) = -x^2 + 1
>> ezplot(fin)
```




```
>> fplot(fin,[-2 2])
```



شکل ۹-۱

۹-۲ ام- فایل تابعی function M-file

تابعی که به صورت برنامه نوشته شده و در داخل یک فایل ضبط شود ام- فایل تابعی `function M-file` نام دارد، و صورت کلی آن این‌گونه است:

```
function [out_arg1,out_arg2,...] = func_name(in_arg1,in_arg2,...)
% Help Statemants
 statemants
 out_arg1 = ...
 out_arg2 = ...
 ...
```

۱- نام‌گذاری `out_arg` ها دلخواه است، در سطر آخر مساوی مقادیری که نتیجه عملیات سطور قبل است قرار می‌گیرند، و مقدار تابع را به صورت یک بردار برمی‌گردانند. اگر تابع تک مقداری باشد، فقط یک `out_arg` دارد.

۲- `func_name` نام تابع و دلخواه است. نام فایل نگه‌دارنده‌ی تابع باید با تابع هم‌نام باشد، یعنی این تابع باید در فایلی به‌نام `func_name.m` ضبط و با همان نام از پنجره فرمان یا از داخل ام- فایل دیگری فراخوانی (اجرا) شود(فراخوانی تابع - `function calling`).

۳- `in_arg` ها پارامترهای وردودی تابع هستند. توابع می‌توانند ورودی‌های برداری از نوع رشته یا عدد داشته باشند.

٤- اجرای help سطور بعداز علامت توضیح % را نمایش می‌دهد.

٥- statements عملیاتی هستند که بر روی in_arg ها و متغیرهای داخلی تابع انجام و منجر به مقدار (مقادیر) برگشتی می‌شوند.

مثال ها:

مقدار معادله درجه دو $n_2x^2 + n_1x + n_0$ و نوشتن راهنما برای تابع.

تابعی به نام y1 را در یک ام- فایل تابعی نوشته و آن را با نامِ y1.m در دیرکتوری جاری ضبط می‌کنیم. متغیرهای x,n2,n1,n0 آرگومان‌های ورودی تابع و p1 آرگومان خروجی آن هستند.

```
function p1 = y1(x,n2,n1,n0)
% Calculates the value of a second order sentence.
p1 = n2*x^2 + n1*x + n0;
```

```
>> help y1
Calculates the value of a second order sentence.
```

```
>> y1(-2,3,4.6,-5.8)
ans = -3.0000
```

محاسبه فاکتوریل از طریق خود فراخوانی (بازگشتی) یک تابع **Function Recursivity**

خود فراخوانی اجرای یک تابع از داخل خود آن تابع است، و مانند یک حلقه تا زمانی‌که خروجی به یک مقدار معین برسد ادامه می‌یابد.

برنامه محاسبه فاکتوریل را از طریق خود فراخوانی در محیط متلب اجرا می‌کنیم.

```
function ff = facto(n)
if n > 1
    ff = n * facto(n - 1);
else
    ff = 1;
end
```

```
>> facto(6)
ans = 720
```

ام- فایل تابعی با چند آرگومان خروجی

یکی از ویژگی‌های متلب این است که در آن می‌توان توابعی با چندین خروجی که هر خروجی هم می‌تواند بردار باشد تعریف کرد. مابین نام آرگومان‌های خروجی تابع باید کاما قرار گیرد مثل: function [p1, p2]

مثال ها:

تابع درجه دو در یک ام- فایل تابعی

```
function [p1, p2] = y2(x,a,b,c)
p1 = a*x^2 + b*x + c;
p2 = 2*a*x + b;
```

```
>> [a b] = y2(-2,3,4.6,-5.8)
a = -3.0000
b = -7.4000
```

اگر x بردار باشد p1,p2 هم بردار خواهند بود %

حل معادله با راه بُرد نیوتن

راه‌برد نیوتن یکی از روش‌های ساده محاسبات عددی برای حل معادله چند جمله‌ای f(x) = 0 است. با روش تکرار برای نزدیک شدن به ریشه از راه‌بُرد نیوتن به این شکل می‌توان استفاده کرد: x → x - f(x)/f'(x) .

یعنی در هر بار تکرار به جای x عبارت x - f(x)/f'(x) جای‌گزین می‌شود. مقدار دهی به x در اولین دور حلقه اختیاری است و حدس اولیه نام دارد. راه‌برد نیوتن برای حدس‌های اولیه مختلف ریشه‌های مختلف می‌دهد.

برای حصول ریشه‌های معادله درجه دو $f(x) = 3x^2 + 4.6x - 5.8$ با راه‌بُرد نیوتن برای ریشه‌های حقیقی با دو حدس اولیه حقیقی، و برای ریشه‌های موهومی با دو حدس اولیه موهومی شروع می‌کنیم. از تابع y2 در این برنامه استفاده می‌کنیم.

```
function [p1, p2] = y2(x,a,b,c)
p1 = a*x^2 + b*x + c;
p2 = 2*a*x + b;
```

```
% M-File script secndegree.m
% solves equation f(x) = a*x^2 + b*x -c
x = input('Enter initial guess: '); % initial guess
n = input('Enter 3 coefficients: ');
ero = 1;
iero = 1e-8; % permissible error
while ero > iero
    x1 = x;
    [y yp] = y2(x, n(1), n(2), n(3));
    x = x - y/yp;
    ero = abs((x-x1)/x);
end
format
disp(['root = ' num2str(x)])
```

```
>> secndegree
Enter initial guess: -1
Enter 3 coefficients: [3  4.6 -5.8]
root = -2.3545

>> secndegree
Enter initial guess: 1
Enter 3 coefficients: [3  4.6 -5.8]
root = 0.82113

>> secndegree
Enter initial guess: i
Enter 3 coefficients: [3  4.6 5.8]
root = -0.76667+1.16i

>> secndegree
Enter initial guess: -i
Enter 3 coefficients: [3  4.6 5.8]
root = -0.76667-1.16i
```

تابع بدون مقدار، متغیر Persistent

یک ام-فایل تابعی (که به اختصار تابع خوانده می‌شود) می‌تواند بدون مقدار برگشتی (آرگومان خروجی) باشد. این‌گونه توابع معمولاً برای نمایش نتایج به کار می‌روند.

معرفی یک متغیر با پیش‌وند persistent باعث می‌شود که مقدار متغیر در مراجعات بعدی به تابع حفظ شود.

آوردن نام تابع بعد از clear باعث می‌شود که مقدار متغیرهای عددی persistent آن صفر شود.

مثال:

نام یک دانشجو را به عنوان ورودی دریافت و همراه با شماره ردیف نمایش می‌دهیم.

ابتدا یک تابع با ورودی نام دانشجو می‌نویسیم. این تابع نام دانشجو را از صفحه کلید دریافت می‌کند و هر بار که اجرا شود نام دریافتی را با شماره ردیف در کنار نام دانشجو (متغیر رشته‌ای wr) چاپ می‌کند. رشته‌ی wr که به تابع sst (ns) ارسال می‌شود یک بردار است که هر عنصر آن یک حرف الفبا است.

عبارت persistent nn باعث می‌شود که مقدار n که شمار ردیف را نشان می‌دهد برای اجرای بعدی تابع حفظ شود.

```
% Function M-File stt.m
function stt(ns)
persistent nn
if isempty(nn)
    nn = 1; % دراجرای اول شماره ردیف را یک می‌دهیم
else
    nn = nn + 1; % در هر بار تکرارِ اجرا شماره ردیف یک واحد افزایش می‌یابد
end
mm = num2str(nn);
disp(['Student number ' mm ' is ' ns])
```

حال برنامه‌ای به نام stdno.m می‌نویسیم که تابع فوق را در درون یک حلقه چند بار اجرا می‌کند.

```
% Script M-File stdno.m
clear stt
```

این دستور در اجرای اولِ تابع stt() مقدار شمارنده را صفرمی‌کند. %

```
wr = 'h';
```

یک مقدار اولیه دل‌خواه برای نام دانشجو در نظر می‌گیریم که حلقه برای بار اول اجرا شود. %

```
while isempty(wr) == 0
```

حلقه تا زمانی‌که نامی را وارد کنیم (wr خالی نباشد)، ادامه می‌یابد. با زدن <Enter> حلقه تمام می‌شود. %

```
  wr = input('Enter the student name: ','s');
```

قرار دادن 's' به عنوان آرگومان دوم input() باعث می‌شود که بتوانیم یک رشته را به wr نسبت دهیم %

```
  if isempty(wr)
```

اگر بدون وارد کردن نام (wr خالی)، کلید <Enter> را بزنیم برنامه تمام می‌شود %

```
     break
  end
  stt(wr)%
end
```

```
>> stdno
Enter the student name: Ali
Student number 1 is Ali
Enter the student name: Maryam
Student number 2 is Maryam
Enter the student name: Masud
Student number 3 is Masud
Enter the student name:
```

زیر تابع subfunction

می‌شود در داخل یک تابع توابع فرعی دیگری را به نام زیرتابع تعریف کرد، زیرتابع‌ها فقط توسط تابع اصلی دیده و فراخوانده می‌شوند، و در خارج از تابع اصلی قابل فراخوانی نیستند. نام فایلِ نگه‌دارنده‌ی ام- فایل تابعی باید نام تابع اصلی باشد.

مثال:

برای حل معادله درجه سه b^3 - 2*b^2 + 3*b - 4 یک ام-فایل تابعی به نام newss() بنویسید، که در آن تابع و مشتق آن به صورت دو زیر تابع با نام‌های ny() و nyp() تعریف شوند.

```
% newss.m
function p1 = newss(a)
p1 =  ny(a)/nyp(a); %

function p2 = ny(b)
p2 = b^3 - 2*b^2 + 3*b - 4 ;

function p3 = nyp(d)
p3 = 3*d^2 - 2*2*d + 3;
```

سپس از درون یک ام- فایل با نام eus.m، تابع newss () را فرامی‌خوانیم:

```
% eus.m
x = input('Enter initial guess: ');
ero = 1;
while ero > 1e-6
    x1 = x ;
    x  = x- newss(x);
    ero=abs((x-x1)/x);
end
disp(['root = ' num2str(x)])
```

```
>> eus
Enter initial guess: 1
root = 1.6506

>> eus
Enter initial guess: i
root = 0.17469+1.5469i

>> eus
Enter initial guess: -i
root = 0.17469-1.5469i
```

۹-۳ تابعِ تابع

گیره تابع function handle ، feval()

اگر در مقابل نام تابع علامت @ بیاید گیره آن تابع را ایجاد کرده‌ایم (چیزی شبیه اشاره‌گر pointer در C++). برای مراجعه به تابع از طریق گیره باید از feval () استفاده کنیم مفهوم گیره امکان استفاده از تابعی به عنوان آرگومان تابع دیگر را فراهم می‌کند و از توانائی‌های مهم متلب محسوب می‌شود.

مثال ها:

نسبت دادن گیره تابع به متغیر دیگر

```
>> hs = @sin;
>> feval(hs,pi/6)
ans = 0.5000
```

کاربرد مستقیم گیره تابع

```
>> feval(@sin,pi/6)
ans = 0.5000
```

تابعِ تابعِ کاربر- تعریف

توابعی که تابع دیگری را به عنوان آرگومان می‌پذیرند تابعِ تابع نام دارند و آرگومان آن‌ها می تواند یک گیره تابع باشد. به تابعِ تابع کاربر- تعریف مثال توجه کنید.

مثال:

یک فایل تابعی ایجاد می‌کنیم که آرگومان ورودیش تابع دیگری باشد.

یک تابع به نام yd.m می‌نویسیم که جواب معادله‌های درجه دو و سه را برای مقدار معینی از x برگرداند.

```
function [p2,p3] = yd(x)
 p2 = x.^2 + x  ;
 p3 = x.^3 + x.^2 + x;
```

حالا فایل تابعی ydh.m را ایجاد می‌کنیم که آرگومانش تابع فوق است:

```
function [out1, out2] = ydh(hy,x)
 [out1 out2] = feval(hy,x);
                              % hy باید گیره تابع باشد.
```

```
>> [a b] = ydh(@yd,1)
a = 2
b = 3

>> x = [1 2];
>> [a b] = ydh(@yd,x)
a = 2     6
b = 3    14
```

چون x بردار است، a و b بردار هستند.

تابع تابع کتاب خانه ای

توابع کتاب خانه ای که تابع دیگری را به عنوان آرگومان می‌پذیرند تابعِ تابع نام دارند و آرگومان آن‌ها می تواند یک تابع inline رشته، یا گیره باشد. برخی از این‌گونه توابع عبارتند از: feval, ezplot, fplot().

مثال ها:

رسم $\sin^3 x$ به صورت تابعِ تابع

```
>> isn = inline('(sin(x))^3');
>> fplot(isn,[-pi pi])
>> ezplot(isn)
```

رسم تابع **humps(x)** (از توابع نمونه داخلی متلب)

برای اطلاع بیشتر در مورد این تابع از پنجره فرمان help humps را اجرا کنید.

مثال ها:

فراخوانی مستقیم گیره

```
>> fplot(@humps,[0 1])
```

نسبت دادن به یک مغیر رشته‌ای

```
>> hu = 'humps(x)';
>> fplot(hu,[0 1])
```

۹-۴ تبدیل فایل های متلب

تبدیل ام- فایل به پرونده پی- کد pcode file

می‌توان جهت پنهان کردن سطور برنامه آن را به صورت pcode در آورد. فایل‌های pcode در محیط MATLAB قابل اجرا هستند، اما قابل ادیت و بازبینی نیستند. با اجرای دستور pcode fun.m یا mcc -B pcode fun.m فایلی به نام fun.p ایجاد و در دیرکتوری جاری ذخیره می‌شود. نتیجه اجرای fun.m و fun.p یکسان است.

مثال:

ام- فایل زیر را به نام pascall.m نوشته و آنرا به صورت فایل pcode در آورید. فایل pascall.p را به دیرکتوری دیگری منتقل کرده، نتیجه اجرای آنرا ملاحظه کنید.

```
ncr = 1;
r = 2;
n = 6;
for k = 1:r
 ncr=ncr*(n-k+1)/k;
end
disp(ncr)
```

```
>> pcode pascall
```

فایل رمزگذاری‌شده با نام

```
pascall.p
```

پدید می‌آید.

تولید برنامه C با کامپایلر متلب MATLAB Compiler

دستور mcc -m functin name از یک ام- فایل تابعی یک برنامه به زبان C و هم‌چنین فایل اجرائی مربوطه را که از پنجره Command Promt ویندوز قابل اجرا است می‌سازد، و در دیرکتوری جاری ذخیره می‌کند. معمولاً برنامه‌های ساخته شده با کامپایلر احتیاج به MATLAB Compiler Run-Time Libraries، و برنامه‌های گرافیکی اجرائی نیاز به C/C++ Graphics Library دارند. یعنی برای اجرا تمام یا مؤلفه‌هائی از نرم‌افزار MATLAB و Visual C++ باید روی کامپیوتر شما نصب شده باشد.

مثال:

ام- فایل تابعی زیر را به یک برنامه به زبان C تبدیل و فایل اجرائی آن را بسازید. کد حاصل شده به زبان C را در دیرکتوری جاری باز کنید و ببینید. فایل اجرائی را در محیط متلب از Command Window با گذاشتن علامت ! در مقابل نام آن اجرا کنید. فایل اجرائی را در محیط Windows از پنجره Command Promt اجرا کنید.

```
function dispp
xs='6.5';
disp(['MATLAB Version is ', xs]);
```

```
>> mcc -m dispp
>> ! dispp.exe
```

```
MATLAB Version is 6.5
```

فایل dispp.c را باز کرده و مشاهده می‌کنیم. پنجره Command Promt را درمحیط ویندوز باز کرده و از داخل آن دستور dispp.exe را اجرا می‌کنیم.

تبدیل برنامه گرافیکی به زبان C++

دستور mcc -B sgl functin name از یک ام- فایل تابعیِ دارای دستورات گرافیکی یک برنامه به زبان C++ و هم‌چنین فایل اجرائی مربوطه را که از پنجره Command Promt ویندوز قابل اجرا است می‌سازد.

مثال:

از ترکیب دو ام- فایل تابعی زیر که یکی دیگری را فراخوانی می‌کند، یک برنامه به زبان C++ هم‌راه با فایل اجرائی بسازید. فایل اجرائی را در محیط MATLAB و از پنجره Command Promt اجرا کنید.

```
function y = sawt(t,T)
y=10*rem(t,T)/T;
```

```
>> mcc -B sgl tsawt
>> !tsawt
```

```
function tsawt
n=200;T=2;t=linspace(0.01,3*T,n);
f=sawt(t,T);
F=fft(f);
f1=ifft(F);
plot(t,abs(f1),'x',t,f)
```

خلاصه دستورات کامپایلر

در کلیه دستورات زیر نام ام- فایل بعداز دستور می‌آید.

mcc -x ترجمه یک M-file به C و تولید C MEX-file که از درون متلب قابل اجراست

mcc -S تولید C و Simulink S-Function مربوطه

mcc -m ترجمه یک M-file به C و تولید فایل اجرائی آن، قابل اجرا بدون نیاز به متلب

mcc -p ترجمه یک M-file به C++ و تولید فایل اجرائی آن، قابل اجرا بدون نیاز به متلب

mcc -B sgl تولید یک فایل کاربردیِ C Graphics Library Application و فایل اجرائی آن قابل اجرا بدون نیاز به متلب، از یک ام- فایل گرافیکی

mcc -B sglcpp تولید یک فایل کاربردیِ C++ Graphics Library Application و فایل اجرائی آن قابل اجرا بدون نیاز به متلب، از یک ام- فایل گرافیکی

mcc -m -W lib:libfoo -T link:lib foo.m تولید یک کتاب‌خانه C از ام- فایل foo.m

mcc -p -W lib:libfoo -T compile:lib تولید یک کتاب‌خانه C++

تولید یک کتاب‌خانه مشترک با C برای محاسبات خاص که از برنامه شما فراخوانی می‌شود

mcc -W lib:mylib -L C -t -T link:lib -h Function1 Function2 ...

mcc -B pcode تولید P-Code برای محیط MATLAB

سازنده اکسل Excel Builder

دستور mxltool یک واسط گرافیکی GUI در اختیار قرار می‌دهد که با آن در محیط متلب می‌توان برنامه‌ها و ماکروهائی جهت استفاده در داخل اکسل نوشت. پس از اجرای این دستور از منیوی Help آن اطلاعات بیش‌تر را به دست آورید.

۹-۵ تمرین

۱- حاصل‌ضرب یک تابع نمائی و یک تابع مثلثاتی را به صورت inline تعریف و با ezplot() رسم کنید.

۲- مقدار، مقدار مشتق، و مقدار مشتق مشتق یک معادله درجه چهار را با یک ام-فایل تابعی با چند آرگومان خروجی برگردانید. همین عمل را این‌بار به صورت زیر تابع انجام دهید. برای ام-فایل های تابعی راهنما بنویسید. معادله را با راه بُرد نیوتن و با استفاده از ام-فایل ها حل کنید

۳- فاکتوریل را با یک تابع، بدون استفاده از بازگشتی محاسبه کنید.

٤- تابعی با ورودی نام بنویسید، که در هر بار اجرا نام دریافتی را به ترتیب با حرف الفبا (A,B,C,...) کنار نام چاپ کند.

۵- یک فایل تابعی ایجاد کنید که آرگومان ورودیش تابع دیگری باشد.

٦- با توابعی که تاکنون نوشته‌اید و با استفاده از کامپایلر متلب برنامه C و برنامه C++ تولید کنید، و آن‌ها را از پنجره فرمان ویندوز اجرا کنید.

فصل ۱۰ ریاضیات نمادین Symbolic Math (Symbolic Math Tool Box)

نوشتن عبارات و فرمول‌ها با حروف و علائم، در متلب ریاضیات نمادین گفته می‌شود. در جبر و آنالیز روابط ریاضی با حروفی نظیر x, y, a ,b نشان داده می‌شوند که روشی کارآمد و آشنا به ذهن و چشم است. متلب با استفاده از ساختار و موتور Maple این کار را انجام می‌دهد. البته خط‌نوشته‌ی کامپیوتری به خوانائی فرمول‌نوشته‌های چاپی و دستی نیست (مثلاً عبارت $\sin^2 2x$ به صورت نمادینِ sin(2*x)^2 نوشته می‌شود). اما کوشش شده است که حداکثر مشابهت با روش نوشتاری جبری فراهم شود. برای کار با ریاضیات نمادین باید از متغیرهای نمادین استفاده کرد.

۱۰-۱ مقایسه چند نوع داده

آرایه یا متغیر عددی با دقت افزوده double array

متغیری که به‌صورت x = 7 تعریف شود، آرایه یا متغیر عددی با دقت افزوده است (پیش‌فرض اعداد تعریف شده دقت افزوده یا double است). پاسخ دستور isnumeric(x) درستی یا منطقِ یک خواهد بود.

آرایه یا متغیر کاراکتری

متغیری که به صورت d = '7' تعریف شود، متغیر کاراکتری یا رشته‌ای است. پاسخ دستور ischar(d) درستی خواهد بود.

شیئ یا متغیر نمادین

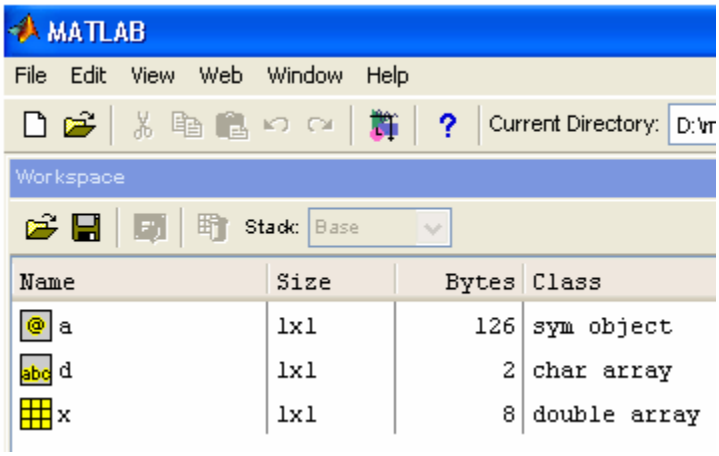
متغیری که به صورت a = sym(7) تعریف شود، یک شیئ نمادین symbolic است و پاسخ isobject(a) درستی است.

انواع دیگر داده

در متلب انواع دیگری از متغیرها نظیر single (که شرح آن قبلاً آمد) و cell array و sparse matrix نیز وجود دارند. برای اطلاع بیشتر help datatypes را اجرا کنید.

نمایش متغیرها در پنجره workspace

سه متغیر فوق که از سه نوع مختلف هستند در workspace با سه نشانک icon مختلف نمایش داده می‌شوند:



شکل ۱۰-۱

۲-۱۰ متغیرهای نمادین

عمل معکوس کردن را به عنوان نمونه در نظر می‌گیریم. به متغیر عددی x مقدار داده، آن را معکوس می‌کنیم. در پایان دو متغیر عددی خواهیم داشت.

```
>> x = 7;
>> xr = x^(-1)
```

```
xr = 0.1429
```

حال متغیر d را مساوی کاراکتر '7' قرار داده، آن را معکوس می‌کنیم. این عمل مقدار عددی یا کد اسکی متغیر d را که یک متغیر کاراکتری است معکوس می‌کند (کد اسکی کاراکتر '7' عدد 55 است)

```
>> d = '7';
>> dr  =  d^(-1)
```

```
dr = 0.0182 % 1/55
```

سپس متغیر نمادین a را مساوی 7 نمادین گرفته و آن را معکوس می‌کنیم، نمادین بودن مقدار معکوس را امتحان کرده، سپس آن را به مقدار عددی تبدیل می‌کنیم. با عمل روی a یک متغیر (شیئ) نمادین دیگر به نام ar ایجاد می‌شود.

```
>> a = sym(7);
>> ar = a^(-1)
```

```
ar = 1/7
```

چون ar در سمت چپ یک عبارت نمادین قرار دارد، خود به خود معرفی می‌شود و نیاز به معرفی جدا ندارد. نمادین بودن این شیئ را با دستور isobject() امتحان می‌کنیم:

```
>> isobject(ar)
```

```
ans = 1
```

با دستور double می‌توان مقادیر نمادین عددی را نیز به عدد تبدیل کرد:

```
>> bs = double(ar)
>> isnumeric(bs)
```

```
bs = 0.1429
ans = 1
```

همین کار با دستور eval() نیز شدنی است:

```
>> eval(ar)
```

```
ans = 0.1429
```

جای گزینی عدد نمادین در متغیر نمادین

متغیر نمادین a را مساوی 5- نمادین قرار داده b = a^2 + a و z = b^.5 را به دست آورده مقادیر عددی آن‌ها را نیز پیدا می‌کنیم.

```
>> a = sym(-5)
>> b = a^2 + a
>> z = b ^ 0.5
```

```
a = -5
b = 20
z = 20^(1/2)
```

چون به b و z مقادیر نمادین نسبت داده شده قابل تبدیل به عدد هستند:

```
>> bd = double(b)
>> zd = double(z)
```

```
bd = 20
zd = 4.4721
```

یافتن متغیرهای نمادین

دستور findsym() متغیرهای نمادین یک تابع نمادین را به ترتیب الفبا نشان می‌دهد.

```
>> syms x t y z
>> f = x^t; f1 = z + t^x*y;
>> findsym(f), findsym(f1)
```

```
ans = t, x    ans = t, x, y, z
```

تعداد n متغیر اول که از x شروع می‌شوند با findsym(f1,n) به دست می‌آید:

```
>> findsym(f1,3)
```

```
ans = x, y, z
```

نمایش اعداد نمادین

متغیر عددی u = 0.257 را در نظر می‌گیریم و با آن متغیر نمادین us را پدید آورده، به دو صورت کسری و اعشاری (تا پنج رقم) نمایش می‌دهیم.

```
>> u = 0.257;
>> us = sym(u)
```

```
us = 257/1000
```

پارامتر 'd' نمایش را به صورت اعشاری در می‌آورد. دستور digits() که فقط برای متغیرهای نمادین به کار می‌رود تعداد ارقام اعداد نمادین را تعیین می‌کند.

```
>> digits(5)
>> us = sym(u,'d')
```

```
us = .25700
```

نمایش متغیرهای نمادین

برای هرنوع عملیات نمادین بایستی ابتدا متغیرهای مورد استفاده را معرفی کرد. این کار با دستورهای sym() و syms انجام می‌شود. متغیرهائی که بر اثر عملیات بر روی اشیاء نمادین حاصل شوند خودبه‌خود نمادین خواهند بود و نیاز به معرفی اولیه ندارند.

```
>> syms a b
>> (a+b)^(1/2)
```

```
ans = (a+b)^(1/2)
```

اما عبارت زیر تولید خطا می‌کند زیرا w معرفی نشده است:

```
>> (a+w)^(1/2)
```

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

یک متغیر نمادین مقداردار و یک متغیر نمادین بدون مقدار را جمع کرده و رشیه دوم می‌گیریم. به معرفی یک متغیر نمادین (سمبلیک) با مقدار دهی و بدون مقدار دهی توجه کنید. اما متغیر c که از عملیات بر روی متغیرهای قبلاً معرفی شده به دست می‌آید، نیاز به معرفی ندارد

```
>> a = sym(7);
>> syms b
>> c = (a+b)^(1/2)
```

```
c = (7+b)^(1/2)
```

با کاربرد عبارات بالا مقدار نمادین a هفت است و مقدار نمادین b خودِ b است، مقدار اشیاء نمادینِ بدون مقدار با خودشان مساوی است. این موضوع را می‌توان روی پنجره فرمان تحقیق کرد:

```
>> a , b
```

```
a = 7  b = b
```

متغیر مستقل نمادین

به صورت پیش‌فرض متغیر مستقل x است، لذا بدون ذکر پارامتر، عمل ریاضی بر حسب x یا نزدیک‌ترین نام به x انجام می‌شود. مثلاً مشتق x^t را بر حسب x و t پیدا می‌کنیم.

```
>> syms x t
>> f = x^t
>> diff(f)
```

```
ans =x^t*t/x
```

اگر بخواهیم متغیر مستقل را خودمان تعیین کنیم، آن‌را به عنوان پارامتر دوم می‌آوریم:

```
>> diff(f,t)
```

```
ans = x^t*log(x)
```

یا مشتق sin(at + b) بر حسب نزدیک‌ترین نام به x که در این‌جا t است گرفته می‌شود:

```
>> g = sin(a*t + b)
>> dg = diff(g)
```

```
dg =cos(a*t+b)*a
```

جای گزینی عدد نمادین در متغیر نمادین

متغیر نمادین a را مساوی 5- نمادین قرار داده a = a^2 + a و z = b^.5 را به دست آورده مقادیر عددی آن‌ها را نیز پیدا می‌کنیم.

```
>> a = sym(-5),b = a^2 + a,z = b^.5
```

```
a = -5  b = 20  z = 20^(1/2)
```

چون به b و z مقادیر نمادین نسبت داده شده قابل تبدیل به عدد هستند:

```
>> bd = double(b), zd = double(z)
```

```
bd = 20  zd = 4.4721
```

یافتن متغیرهای نمادین

دستور زیر متغیرهای نمادین تابع $f = x^t$ را نمایش می‌دهد:

```
>> findsym(f)
```

```
ans = t, x
```

اولین متغیر مستقل با اضافه کردن پارامتر 1 به دستور فوق به دست می‌آید:

```
>> findsym(f,1)
```

```
ans = x
```

نمایش اعداد نمادین

با متغیر عددی u = 0.257 متغیر نمادین us را پدید آورده، به دو صورت کسری و اعشاری (تا پنج رقم) نمایش می‌دهیم.

```
>> u = 0.257;
>> us = sym(u)
```

```
us = 257/1000
```

پارامتر 'd' نمایش را به صورت اعشاری در می‌آورد. دستور digits() که فقط برای متغیرهای نمادین به کار می‌رود تعداد ارقام اعداد نمادین را تعیین می‌کند.

```
>> digits(5)
>> us = sym(u,'d')
```

```
us = .25700
```

۳-۱۰ عملیات ریاضی

ریشه دوم

ریشه دوم متغیر نمادین a به طریق زیر به دست می‌آید.

```
>> syms a
>> b = sqrt(a)
```

```
b = a^(1/2)
```

توان

ریشه دوم exp(-z) را از طریق به توان رساندن به دست می‌آوریم.

```
>> syms z
>> b = exp(-z);
>> z = b ^ 0.5
```

```
z = exp(-z)^(1/2)
```

مشتق

مشتق اول و دوم عبارت -z^2 + a^2 را پیدا می‌کنیم.

```
>> syms a z
>> d = diff(-z^2 + a^2)
>> dd = diff(d)
```

```
d = -2*z
dd = -2
```

انتگرال

انتگرال a^2 + a را تعیین می‌کنیم.

```
>> syms a
>> b = a^2 + a
>> c = int(b)
```

```
c = 1/3*a^3+1/2*a^2
```

انتگرال محدود

انتگرال exp(-z) را مابین صفر تا بی‌نهایت پیدا می‌کنیم.

```
>> syms z
>> b = exp(-z);
>> c = int(b,z,0,inf)
```

```
c = 1
```

تبدیل به کسرهای جزئی و ریشه و قطب یک تابع تبدیل

دستور residue() یک کسر را از فرم ۱ به فرم ۲ (کسرهای جزئی) تبدیل می‌کند.

$$\frac{b(s)}{a(s)} = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \ldots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \ldots + a_{n+1}}$$

فرم ۱

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots + \frac{r_n}{s-p_n} + k(s)$$

فرم ۲

مثال:

برای حصول ریشه و قطبِ تابع تبدیل فرم ۱ ، آن را به کسرهای جزئی تبدیل می‌کنیم. عبارت تبدیل شده به شکل فرم ۲ خواهد بود.

$$\frac{5s^3 + 3s^2 - 2s + 7}{-4s^3 + 8s + 3}$$

فرم ۱

$$\frac{-1.4167}{s-1.5737} - \frac{0.6653}{s+1.1644} + \frac{1.3320}{s+0.4093} - 1.25$$

فرم ۲

```
>> b = [ 5 3 -2 7]
>> a = [-4 0 8 3]
>> [r, p, k] = residue(b,a)
```

```
r = -1.4167   -0.6653    1.3320
p =  1.5737   -1.1644   -0.4093
k = -1.2500
```

تبدیل عبارت جبری به کسر متعارفی گویا

[N,D] = numden(A) عناصر ماتریس A را به کسر متعارفی گویا تبدیل می‌کند. چندجمله‌ای‌های صورت و مخرج حتی‌الامکان تجزیه‌ناپذیر بوده و ضرائب آن‌ها اعداد صحیح هستند. اگر عبارت $\frac{x}{y}+\frac{y}{x}$ را به صورت کسر متعارفی گویا درآوریم.

```
>> syms x y
>> [n,d] = numden(x/y + y/x)
```

```
n = x^2 + y^2
d = y*x
```

عبارت تبدیل شده به این شکل می‌باشد: $\frac{x^2+y^2}{yx}$

۱۰-۴ اعداد مختلط نمادین

برای این که بتوانیم قسمت موهومی را به صورت i*y نشان دهیم بایستی y حتماً عدد حقیقی باشد. لذا از پارامتر real استفاده می‌کنیم. حقیقی بودن x و y به معنی مثبت بودن عبارت x^2 + y^2 می‌باشد.

مزدوج یک عدد مختلط

مزدوج یک عدد مختلط نمادین را به دست آورده، در خودش ضرب کرده، و نمایش می‌دهیم. دستور expand() در این‌جا ضرب دو پرانتز را باز می‌کند.

```
>> syms x y real
>> z = x + i*y ;
>> cz = conj(z)
>> az = z*conj(z)
>> expand(z*conj(z))
```

```
cz = x-i*y
az =(x+i*y)*(x-i*y)
ans = x^2+y^2
```

۱۰-۵ توابع نمادین

معرفی یک تابع کلی

مثال:

تابع f(x) را به صورت نمادین معرفی می‌کنیم.

```
>> f = sym('f(x)')
```

```
f = f(x)
```

جای گزینی یک عبارت به جای x (subs = substitution)

مثال ها:

۱)

با عبارت subs(f,x,x+h) ، x+h را به جای x قرار می‌دهیم

```
>> syms x h y
>> fs = subs(f,x,x+h)
```

```
fs = f(x+h)
```

۲)

مقدار نمادین $\frac{fs-f}{h}$ را به دست آورید.

```
>> df = (fs-f)/h
```

```
df = (f(x+h)-f(x))/h
```

تابع نمادین مختلط

مثال:

مشتق e^{ix} را به دست می‌آوریم.

```
>> ep = exp(i*x)
>> ep1 = diff(ep)
```

```
ep1 = i*exp(i*x)
```

۱۰-۶ حد تابع

در گرفتن حد، اگر جهت میل متغیر را ننویسیم، پیش فرضِ حد یعنی $x \to 0$ اعمال می‌شود.

مثال ها:

تعیین $\lim_{h\to\infty} \frac{\cos(x+h)-\cos x}{h}$

```
>> syms x h
>> lm = '(cos(x+h) -
cos(x))/h';
>> li = limit(lm, h, inf )
```

h متغیر میل کننده و inf (∞) مقصد تمایل است %

```
li = 0
```

می‌توان به این صورت نوشت: lm = sym('(cos(x+h) - cos(x))/h') ، اما متلب اجازه می‌دهد که تابع را به صورت رشته تعریف کرده، سپس عملیات نمادین را انجام دهیم. به این ترتیب نوشتن توابع راحت‌تر و نمایش آن‌ها واضح‌تر می‌شود.

سؤال: با توجه به رشته‌ای بودن lm متغیر li از چه نوع است؟

تعیین مشتق از طریق یافتن حد

```
>> syms x h
>> lm = '(cos(x+h) -
cos(x))/h'
>> lz = limit(lm, h, 0 )
```

```
lz = -sin(x)
```

حد و گرایش حد

حد تابع 1/x را به ازای x → 0 بدون گرایش

```
>> limit(1/x)
```

```
ans = NaN
```

حد تابع 1/x را به ازای x → 0 با گرایش چپ

```
>> limit(1/x,x,0,'left')
```

```
ans = -inf
```

حد تابع 1/x را به ازای x → 0 با گرایش راست

```
>> limit(1/x,x,0,'right')
```

```
ans = inf
```

۱۰-۷ تابع ام- فایلی نمادین

مثال:

تابع ام- فایلی بنویسید که به ازای یک آرگومان نمادین مقدار $\frac{\sin x}{x}$ را برگرداند، آن را در دیرکتوری جاری ضبط کنید. این تابع را با یک آرگومان نمادین بدون مقدار و یک آرگومان نمادین مقدار دار از پنجره فرمان اجرا کنید. هم‌چنین برای آن یک راهنما بنویسید و آن را نیز اجرا کنید.

```
% Function M-File sinc.m
function z = sinc(x)
%SINC The symbolic sinc function sin(x)/x.
%This function receives a symbolic variable as the input argument.
if isequal(x,sym(0))
 z = 1;
else
 z = sin(x)/x;
end
```

```
>> syms q
>> sinc(q)
```

```
ans = sin(q)/q
```

```
>> x = sym(0);
>> sinc(x)
```

```
ans = 1
```

```
>> help sinc
```

```
SINC The symbolic sinc function sin(x)/x.
This function receives a symbolic variable as the input argument.
```

۱۰-۸ سری ها

سری با دستور symsum(s,a,b) محاسبه می‌شود. a,b مقادیر ابتدائی و انتهائی پارامتر سری را معین می‌کنند

مثال:

محاسبه سری: $\sum_{k=1}^{\infty} \frac{1}{k^2}$

```
>> syms x k
>> s1 = symsum(1/k^2,1,inf)
```

```
s1 = 1/6*pi^2
```

۱۰-۹ توابع آسان ساز

برای نمایش واضح‌تر عبارات نمادین از این توابع استفاده می‌کنیم. استفاده از توابع آسان ساز همیشه با موفقیت همراه نیست و بستگی به هوشمندی نرم‌افزار دارد.

تابع pretty()

یک عبارت را به فرم جبری خوانا نمایش می‌دهد.

مثال:

```
>> syms x
>> f1 = (5+4*cos(x))^3*sin(x)^2*(1+sin(x));
>> pretty(f1)
                                   3         2
                  (5 + 4 cos(x))   sin(x)   (1 + sin(x))
```

توابع collect() و expand()

مکمل یک دیگر هستند و عبارات نمادین را باز یا بسته می‌کنند.

مثال:

```
>> syms x y
>> f = x^2*y + y*x - x^2 - 2*x;
>> collect(f)

ans = (y-1)*x^2+(y-2)*x

>> g = (2*x+5)^2
>> expand(g)

ans = 4*x^2+20*x+25
```

فاکتورگیری factor()

مثال:

از عبارت x^3-1 و عدد نمادین 625 فاکتور می‌گیریم.

```
>> syms x
>> g = x^3-1
>> gf = factor(g)

gf =(x-1)*(x^2+x+1)

>> y = sym(625);
>> factor(y)

ans = (5)^4
```

ساده کردن با simplify()

سعی می‌کند هر عبارت نمادین را به ساده‌ترین فرم نمادین نمایش دهد.

مثال:

عبارات $e^y e^x$, $\sin^2 x + \cos^2 x$, $x^2 x^5$ را ساده می‌کنیم.

```
>> syms x y
>> simplify(exp(y)*exp(x))

ans = exp(y+x)

>> simplify(sin(x)^2+cos(x)^2)

ans = 1

>> simplify(x^2*x^5)

ans = x^7
```

ساده کردن با simple()

این دستور انواع دستورهای ساده‌سازی را که بر روی یک عبارت قابل اجرا است، اجرا می‌کند. خودتان simple را برای sin(x)*cos(x) اجرا کنید.

۱۰-۱۰ ماتریس های نمادین

ماتریس با عناصری از متغیرهای نمادین

یک ماتریس با عناصر نمادین به طریق زیر تعریف می‌شود.

```
>> syms a b c
>> M = [a b c; a/2 b/2 c/2; c a b]
```

```
M = [     a,     b,     c]
    [ 1/2*a, 1/2*b, 1/2*c]
    [     c,     a,     b]
```

```
>> s2 = sum(M(2,:)) % مجموع عناصر ردیف دو
```

```
s2 = 1/2*a+1/2*b+1/2*c
```

```
>> a = sym(-6); b = sym(-6); c = sym(-6); % مقادیر نمادینی به متغیرها نسبت می‌دهیم
>> sv = eval(s2) % عدد نمادین است
>> isobject(sv)
```

```
sv = -9

ans = 1
```

با ()double قابل است:

```
>> svd = double(sv) % تبدیل به عدد محاسباتی
>> isnumeric(svd)
```

```
svd = -9

ans = 1
```

ماتریس با عناصری از توابع نمادین

ماتریس زیر را تشکیل می‌دهیم، مشتق می‌گیریم، در ماتریسِ مشتق، a را برابر مقدار نمادین 1- و x را برابر مقدار نمادین pi/4 قرار می‌دهیم.

$$\begin{bmatrix} \cos(a*x) & \sin(a*x) \\ -\sin(a*x) & \cos(a*x) \end{bmatrix}$$

```
>> syms a x
>> M = [cos(a*x) sin(a*x ); -sin(a*x ) cos(a*x)];
>> Md = diff(M)
```

```
Md = [ -sin(a*x)*a,  cos(a*x)*a]
     [ -cos(a*x)*a, -sin(a*x)*a]
```

```
>> a = sym(-1);
>> Mde = eval(Md)
```

```
Mde = [ -sin(x), -cos(x)]
      [  cos(x), -sin(x)]
```

```
>> x = sym(pi/4);
>> MM = eval(Mde)
```

```
MM = [ -1/2*2^(1/2), -1/2*2^(1/2)]
     [  1/2*2^(1/2), -1/2*2^(1/2)]
```

```
>> double(MM)
```

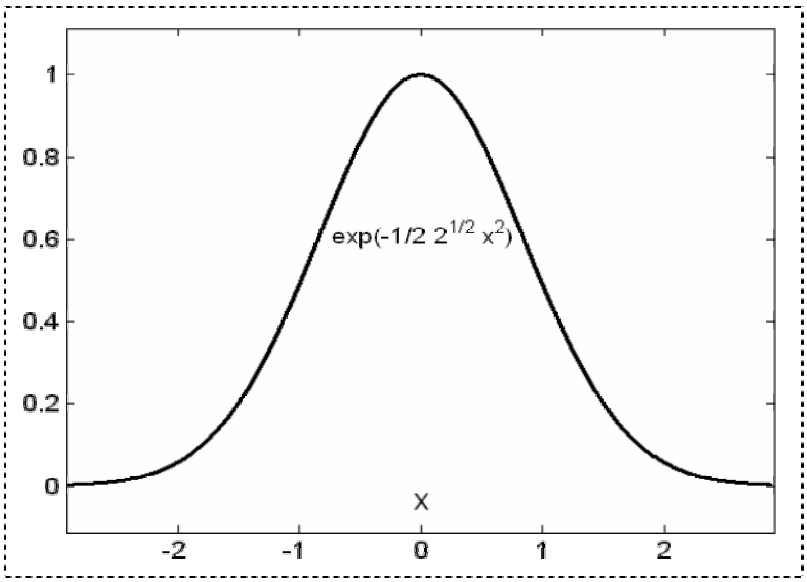
```
ans = -0.7071   -0.7071
       0.7071   -0.7071
```

۱۰-۱۱ رسم تابع نمادین با ezplot()

مثال:

رسم تابع $e^{-\frac{1}{\sqrt{2}}x^2}$:

```
>> syms x
>> k = sym(-
1/sqrt(2));
>> f = exp(k*x^2);
>> ezplot(f)
```



شکل ۱۰-۲

۱۰-۱۲ دریافت راهنما در مورد ریاضیات نمادین

برای دریافت راهنما در مورد ریاضیات نمادین این دستورها را از پنجره فرمان اجرا کنید:

```
help mfunlist, mhelp index[packages], mhelp diff
```

پیشوند m در کلمات فوق به‌جای Maple آمده است.

هم‌چنین می‌توانید منوی Help_MATLAB Help را اجرا و راهنمای Symbolic Math Toolbox را مطالعه کنید.

۱۰-۱۳ تمرین

۱- تابع `tan(y/x)` را تعریف کرده مشتق‌های آن را نسبت به هریک از دو متغیر جداگانه به دست آورید.

۲- مشتق‌های اول و دوم تابع `exp(i*x)` را تعیین کنید.

۳- متغیر نمادین `a` را مساوی 5- نمادین قرار داده `b = a^2 + a` و `z = b^.5` را به دست آورده مقادیر عددی آن‌ها را نیز پیدا کنید.

٤- ماتریس `Mgi = magic(3)/5` را ایجاد کنید. آن را به صورت نمادین `Mgs = sym(Mgi)` درآورید. سپس دترمینان `det(Mgs)` و ماتریس معکوس `inv(Mgs)` آن را به دست آورید.

٥- مشتق `x^2+2*x` را با استفاده از جای‌گزینی و حد به دست آورید.

٦- این حد را $\lim_{n\to\infty}(1+\frac{x}{n})^n$ تعیین کنید.

۷- مشتق تابع `r = sqrt(x^2 + y^2 + z^2)` را به دست آورید. تابع `ri = int(r)` را به دست آورده، و با دستور `pretty(ri)` آن‌را خواناتر نمایش دهید.

۸- منحنی تابع زیر و منحنی مشتق آن را رسم کنید:

```
f2 =32/(5+4*cos(x))^3*sin(x)^2+4/(5+4*cos(x))^2*cos(x)
```

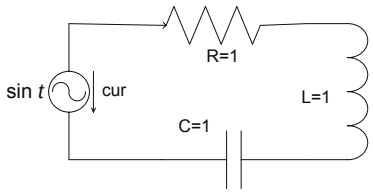
۹- سری $\sum_{k=0}^{\infty} x^k$ را برای `0 < x < 1` حساب کنید.

۱۰- ماتریس `[a b c; a/2 b/2 c/2; c a b]` را تشکیل دهید. حاصل‌جمع عناصر ردیف ۲ آن‌را به صورت نمادین حساب کنید. مقدار عددی این حاصل‌جمع را در ازای `a = b = c = -6` به دست آورید.

۱۱- معادله `a*x^3-2*a*x+1` را حل کنید، سپس با دستور `pretty()` ریشه‌ها را نمایش دهید. به ازای `a = -1` مقادیر عددی ریشه‌ها را به دست آورید.

۱۲- معادله دیفرانسیل $\frac{d^3u}{dx^3}=u$ را با شرائط اولیه $u(0)=1,\ \frac{du}{dx}(0)=-1,\ \frac{d^2u}{dx^2}(0)=\pi$ حل کنید.

۱۳- شدت جریان یک مدار RLC را برای `ω = 1` با مقادیر یک برای عناصر مدار، ورودی سینوسی، و با شرائط اولیه صفر به‌دست آورده و رسم کنید. **راهنما:** برای نشانه شدت جریان به‌جای `i` از `cur` استفاده کنید.



۱٤- فرمول زیر، حل یک دستگاه با شرائط اولیه `f(0)=0, g(0)=1` را می‌دهد. پاسخ‌ها را رسم کنید.

```
[f,g] = dsolve('Df=3*f+4*g,Dg =-4*f+3*g','f(0) = 0,g(0)= 1')
```

۱۵- تابع تبدیل زیر را به کسرهای جزئی تبدیل کنید و دوباره به کسر کلی برگردانید. تغییر در ضریب‌ها و نرمال شدن ضریب ترم اول مخرج را ملاحظه کنید:

$$\frac{5s^3+3s^2-2s+7}{-4s^3+8s+3}$$

۱٦- با استفاده از فاکتورگیری، با نوشتن یک ام- فایل توان‌های ۱ تا ٦ عدد ۳ را در یک جدول نمایش دهید.

فصل ۱۱ عملیات محاسباتی

۱۱-۱ حل معادلات

معادله چند جمله ای، دستورهای roots() و poly()

دستور roots(d) ریشه‌های چندجمله‌ای را به دست می‌دهد که ضرائب آن اعضاء بردار d = [n1 n2 ...] باشند. این دستور از روش‌های پیشرفته‌ی محاسبات عددی برای رسیدن به ریشه‌ها استفاده می‌کند و نیازی به حدس اولیه ندارد. roots() ریشه‌های موهومی و حقیقی را به دست می‌آورد.

دستور poly(b) چند جمله‌ای را به دست می‌دهد که ریشه‌های آن اعضاء بردار b باشند. این دو توابع معکوس یک-دیگر هستند.

مثال ها:

ریشه‌ی معادله‌ی چند جمله‌ای

```
>> r1 = roots([3,4.6,-5.8])
r1 = -2.3545
      0.8211
>> r2 = roots([3,4.6,5.8])
r2 = -0.7667 + 1.1600i
     -0.7667 - 1.1600i
```

ضرائب چند جمله‌ای از روی ریشه‌ها

```
>> rr = roots( [1.0000   -2.0000    3.0001   -4.0001])
rr = 1.6506
     0.1747 + 1.5469i
     0.1747 - 1.5469i
>> poly(rr)
ans = 1.0000   -2.0000    3.0001   -4.0001
```

حل معادله با تابع کتاب خانه ای fzero()

fzero() سعی می‌کند که مقدار ریشه واقعی را حول و حوش یک حدس اولیه یا نقطه شروع starting point که دستی وارد می‌شود، پیدا کند fzero() در واقع محل تغییر علامت تابع را پیدا می‌کند، لذا بهتر است برای توابع پیوسته به‌کار رود.

این تابع که به‌این شکل نوشته می‌شود fzero(f,x0) دو آرگومان اصلی دارد. تابع f که می‌تواند به صورت رشته، گیره، یا خط فرمان وارد شود. حدس اولیه x0 ، مقداری است حتی‌الامکان نزدیک به ریشه و دستی وارد می‌شود.

پارامترهای دیگر:

P که به این صورت وارد می‌شود fzero(f,X0,P) . در صورت مراجعه به گیره تابع، P مقدار ورودی تابع f خواهد بود.

options که به این صورت وارد می‌شود fzero(f,X0,options) برای درک بهتر options به مثال زیر یا help optimset مراجعه کنید.

مثال ها:

ارسال به صورت رشته

```
>> y = 'cos(x)+sin(x)+log(x)';
>> X0 = fzero(y,1)
X0 = 0.2885
```

ریشه را امتحان می‌کنیم:

```
>> cos(X0) + sin(X0) + log(X0)
ans = 0
```

ارسال به صورت گیره

```
function y = fz(q)
y = exp(q)*sin(q)+log(q);
```

```
>> x0 = 1;
>> rt = fzero(@fz,x0)
rt = 0.4771
```

اگر مایل باشیم تعداد دفعات تکرار برای رسیدن به نتیجه نمایش داده شود:

```
>> options = optimset('Display','iter'); x0 = 1;
>> rt = fzero(@fz,x0,options)
```

نتیجه را خودتان امتحان و مشاهده کنید.

حل دستگاه معادلات غیر خطی با()fsolve (جعبه ابزار بهینه سازی Optimization Toolbox)

فرم کلی این تابع به شکل زیر است:

```
x = fsolve(fun,x0,options,P1,P2, ... )
```

fun تابع مخصوصی است که ***فرم*** معادلات غیرخطی خود را در آن می‌گذاریم، و ویژه کاربرد در آرگومان fsolve() یا نظائرش مانند ode45() است. بقیه پارامترها مشابه fzero هستند. اگر سمت چپ تساوی [x,fval] قرار دهیم، جواب معادلات دستگاه نیز در ازای ریشه های به دست آمده چاپ خواهند شد.

مثال:

دستگاه زیر را حل می‌کنیم.

$$x_2 - x_1^{-x_2} = 0$$
$$2x_1 - x_2^{-x_1} = 0$$

```
function F = fsl(x)
 F(1) = x(2)-x(1) ^ -x(2);
 F(2) = 2*x(1)- x(2) ^ -x(1);
 % The following form of writing F is also accepted
 % F = [x(2)-x(1) ^ -x(2);
 %     2*x(1)- x(2) ^ -x(1)];
```

```
>> format bank
>> x0 = [2 2];
>> [a,fv] = fsolve(@fsl,x0);
a = 2.98    0.55              fv = -0.00      -0.00
```

نوشتن fv به عنوان پارامتر مشابه عمل زیر است:

```
>> fsl(a)
ans = -0.00      -0.00
```

حل دستگاه معادلات خطی

بهتر است دستگاه معادلات خطی با استفاده از تقسیم راست به چپ ماتریسی حل شود.

مثال:

دستگاه معادلات زیر را در برنامه اصلی متلب حل کنید.

$$x + \frac{1}{2}y + \frac{1}{3}z = 1$$
$$\frac{1}{2}x + \frac{1}{3}y + \frac{1}{4}z = 1$$
$$\frac{1}{3}x + \frac{1}{4}y + \frac{1}{5}z = 1$$

صورت ماتریسی دستگاه فوق چنین است:

$$\begin{vmatrix} 1 & 1/2 & 1/3 \\ 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 \end{vmatrix} \times \begin{vmatrix} X1 \\ X1 \\ X1 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}$$

که به صورت M * X = B نمایش داده شده و با فرمول X = M\B حل می‌شود:

```
>> M = [ 1 1/2 1/3; 1/2 1/3 1/4; 1/3 1/4 1/5];
>> B = [1 1 1]'   ;
>> X = M\B
```

```
X = 3.0000  -24.0000  30.0000
```

حل معادلات با دستور solve() (جعبه ابزار ریاضیات سمبلیک Symbolic Math Toolbox)

از دستور solve(fun) برای تعیین سمبلیک ریشه‌های معادلات استفاده می‌شود. fun دستگاهی از معادلات است.

متغیر مستقل نزدیک‌ترین حرف به x است، اما در فرم solve(fun,var) پارامتر var تعیین کننده متغیر مستقل است.

معادله fun می‌تواند به صورت سمبلیک یا به صورت رشته نوشته شود. اگر طرف راست معادله صفر باشد می‌توانیم آن-را ننویسیم، در غیر این صورت باید طرف راست بعد از علامت تساوی نوشته شود.

اگر معادله دارای ضرائب عددی باشد، پاسخ‌های عددی سمبلیک حاصل می‌شود، که قابل تبدیل به عدد است. اگر معادله دارای ضرائب سمبلیک باشد، پاسخ‌های سمبلیک حاصل می‌شود.

مثال ها:

پاسخ عددی

```
>> syms x
>> f1 = (5+4*cos(x))^3*sin(x)^2*(1+sin(x))
>> z = solve(f1)
```

```
z = [ pi-acos(5/4)]
    [             0]
    [      -1/2*pi]
```

ریشه‌ها را از حالت نمادین به عدد تبدیل می‌کنیم:

```
>> zd = double(z)
```

```
zd = 3.1416 - 0.6931i
     0
     -1.5708
```

پاسخ سمبلیک

```
>> syms a b c x
>> S = a*x^2 + b*x + c;
>> X = solve(S)
```

```
X = [1/2/a*(-b+(b^2-4*a*c)^(1/2))]
    [1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

تغییر متغیر مستقل

متغیر مستقل را b می‌گیریم و برحسب b حل می‌کنیم، لذا x به صورت پارامتر عمل می‌کند:

```
>> solve(S,b)
ans = -(a*x^2+c)/x
```

حل دستگاه دو معادله با رد معادلات به صورت رشته

$x^2y^2 = 0$

$x - \frac{y}{2} - a = 0$

```
>> q = 'x^2*y^2, x-y/2-a' ;
>> [x,y] = solve(q)
x = [ 0 0 a a]    y = [ -2*a  -2*a  0   0]
```

رد معادلات به صورت سمبلیک و محاسبه مقادیر عددی

```
>> syms x y a
>> [x,y] = solve(x^2*y^2, x-y/2-a)
>> a = sym(1);
>> eval(x),  eval(y)
ans = [ 0 0 1 1]    ans =[ -2 -2  0  0]
```

حل معادله دیفرانسیل عادی با ODE45

شکل کلی این دستور این گونه است:

```
[T,X] = ODE45(odefun,tspan,X0,options,p1,...)
```

odefun تابع مخصوصی است که ***فرم*** معادلات دیفرانسیل خود را در آن می‌گذاریم، و ویژه کاربرد در آرگومان ode45() یا نظائرش مانند fsolve() است.

tspan فاصله‌ای است که در آن تابع انتگرال گیری شده وجود دارد (منحنی یا شکل موج جواب معادلات در آن فاصله ترسیم می‌شود).

X0 بردار شرائط اولیه است

options یک بردار از نوع ساختار structure است و شامل انتخاب هائی است که توسط مقادیر odeset تعیین می‌شوند. برای اطلاع بیشتر به راهنمای آن مراجعه شود.

p1,... بردارهائی هستند که در صورت وجود داشتن در آرگومان odefun به آن رد می‌شوند.

X بردار ستونی معادله های پاسخ است که برحسب بردار T باید در نظر گرفته شود. مثلاً اگر دو معادله داشته باشیم، X(:,1) بردار معادله اول و X(:,2) معادله دوم هستند.

مثال:

دستگاه معادلات دیفرانسل زیر را حل کنید.

$dx_1/dt = x_2$

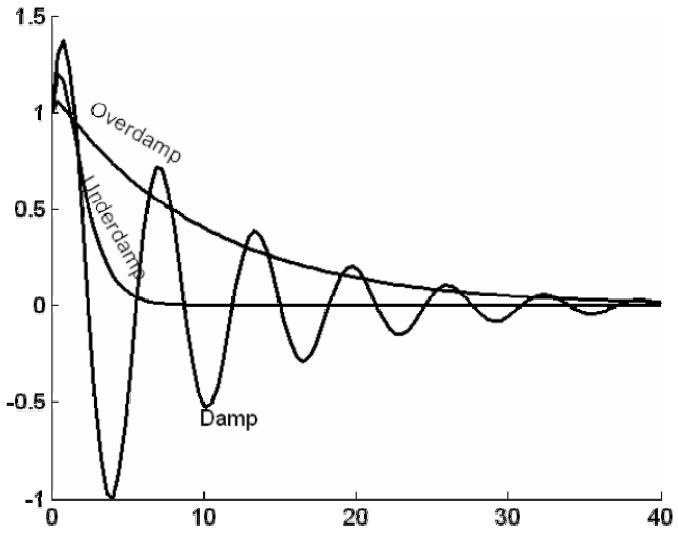
$dx_2/dt = -2\delta\ dx_1/dt - x_1$

ابتدا هر دو معادله را در تابع زیر شبیه سازی می‌کنیم:

```
function dx = frosc(t,x,del)
dx = zeros(2,1,1)
dx(1) = x(2);
dx(2) = -2*del*x(2)-x(1);
% dx = [x(2);  -2*del*x(2)-x(1)]; % this form is also correct
```

پارامترهای ضروری را در ODE45 جای‌گزین می‌کنیم، چون options نداریم به جای آن بردار تهی می‌گذاریم.

```
% odee.m
echo off;
odefun = @frosc;
tspan = linspace(0,40);
X0 = [1 1];
options = [];
p = [0.1 1 5];
%underdamp, damp,
%over damp
clf
hold on
for k = 1:3
 [T,X] = ...
 ODE45(odefun,tspan,...
       X0,options,p(k));
 plot(T,X(:,1))
end
hold off
```



شکل ۱۱-۱

سؤال: دو معادله فوق را به صورت تک معادله $d^2x/dt^2 - 2\delta\, dx/dt + x = 0$ در آورید و تحقیق کنید که یک معادله نوسان کننده (نظیر مدار LC یا بار- فنر) است. ضریب δ نشان دهنده کدام عامل فیزیکی است؟

حل معادلات دیفرانسیل عادی (جعبه ابزار ریاضیات سمبلیک Symbolic Math Toolbox)

دستور زیر برای حل سمبولیک معادلات دیفرانسیل عادی به کار می‌رود:

```
r = dsolve('eq1','eq2',...,'cond1','cond2',...,'v')
```

eq ها معادلات، cond ها شرایط اولیه، و v متغیر مستقل است، در صورت نبود آن نزدیک ترین حرف به x متغیر مستقل فرض می‌شود.

مثال ها:

حل معادله دیفرانسیل درجه یک مرتبه اولِ $\frac{dy}{dx} = 1 + y^2$ بدون شرط اولیه و با ثابتِ **C1**

```
>> y = dsolve('Dy = 1 + y^2')
y = tan(t+C1)
```

حل معادله با شرط اولیه **y(0) = 1**

```
>> y = dsolve('Dy = 1 + y^2', 'y(0) = 1')
y = tan(t+1/4*pi)
```

سؤال: جواب فوق را با مشتق‌گیری امتحان کنید.

حل معادله دیفرانسیل مرتبه اول درجه دو $(\frac{dx}{dt})^2 + x^2 = 1$ با شرط اولیه **x(0) = 0**

```
>> x = dsolve('Dx)^2 + x^2 = 1','x(0) = 0')
x = [-sin(t)]
    [ sin(t)]
```

معادله دیفرانسیل مرتبه دوم $\frac{d^2y}{dx^2} = \cos 2x - y$ با شرائط اولیه $y(0) = 1, \frac{dy}{dx}(0) = 0$

```
>> y = dsolve('D2y = cos(2*x) - y', 'y(0) = 1', 'Dy(0) = 0', 'x');
>> simplify(y)
y =-2/3*cos(x)^2+1/3+4/3*cos(x)
```

حل دستگاه معادلات دیفرانسیل

$$\begin{cases} \frac{df}{dt} = 3f + 4g \\ \frac{dg}{dt} = -4f + 3g \end{cases}$$

```
>> [f g] = dsolve('Df = 3*f+4*g', 'Dg = -4*f+3*g')
f = exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)
g = -exp(3*t)*(sin(4*t)*C1-cos(4*t)*C2)
>> pretty(f)
exp(3 t) (cos(4 t) C1 + sin(4 t) C2)
```

تابع معکوس یک تابع با finverse()

این دستور تابع معکوس تابعی یک تابع ریاضی را بر می‌گرداند.

مثال ها:

```
>> finverse(sin(x))
asin(x)
>> finverse(exp(u-2*v),u)
2*v+log(u)
```

ترکیب تابعی با compose

عبارت `compose(f,g)` تابع `g = g(y)` را در `f = f(x)` ترکیب می‌کند به نحوی که `f(g(y))` به دست آید.

مثال:

```
>> syms x y
>> f = 1/x + x^2; g = sin(y);
>> compose(f,g)
>> u = compose(f,g)
u = 1/sin(y)+sin(y)^2
```

۲-۱۱ تقریب جبری منحنی معادلات

گاهی ضروری است که رابطه‌ی نامشخصی را که مابین تعدادی داده وجود دارد به صورت یک معادله جبری چند جمله-ای درآوریم. توابع چندجمله‌ای زیر برای انجام این کار در نظر گرفته شده‌اند.

برخوراندن یک منحنی در معادله چند جمله ای Curve Fitting with polyfit(x,y,n), polyval()

اگر `x` و `y` دو بردار با تعداد عناصر مساوی و رابطه تابعی جبری یا نامشخص باشند، یعنی `y = f(x)`. دستور `pf = polyfit(x,y,n)` ضرائب معادله درجه `n`، `y1 = f1(x)` را در بردار `pf` قرار می‌دهد (طبعاً `pf` دارای `n+1` عضو خواهد بود). `y1` منحنی برازش است و مقادیر بردار `y1` به مقادیر `y` نزدیک هستند، به نحوی که:

$$y = f(x) \approx$$
$$y1 = f1(x) =$$
$$pf(1)x^{n} + pf(2)x^{n-1} + \ldots + pf(n)x + pf(n+1)$$

مقدار `n` که انتخابش به عهده کاربر است در میزان برازش دو منحنی تأثیر دارد.

پس از به دست آوردن ضرائب `pf`، اجرای دستور `y1 = polyval(pf,x)`، بردار `y1` را با تعداد عناصر مساوی عناصر `x` و `y` ایجاد می‌کند.

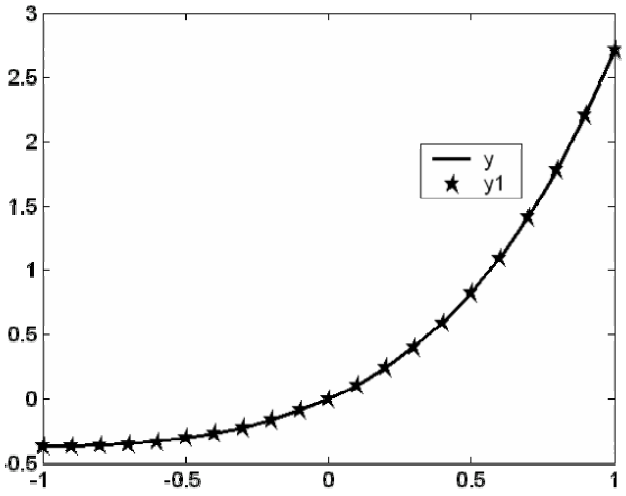
مثال:

تابع `y = exp(x).*x` را در یک چند جمله‌ای درجه چهار بخورانید. مقادیر بردار چندجمله‌ای را با نام `yy` با حلقه `for` تعیین و تعدادی از آن‌ها را با بردار `y` مقایسه کنید. `y1` را با `polyval()` به دست آورید دقت کنید که `y1` همان `yy` خواهد بود. منحنی‌های `y` و `y1` را مقایسه کنید.

```
% polyf.m
x = -1:0.1:1;  y = exp(x).*x; n = 4;
pf = polyfit(x,y,n);
```

```
for k=1:21
 yy(k)=...
 pf(1)*x(k)^4+pf(2)*x(k)^3+pf(3)*x(k)^2+pf(4)*x(k)+pf(5);
end
er = abs(y-yy)
disp('    y          yy         y-yy')
disp([y(1:5)' yy(1:5)' er(1:5)'])
y1 = polyval(pf,x); % yy1 is the same as yy
plot(x,y,x,y1,'p')
>> polyf.m
```

```
 y          yy         y-yy
-0.3679   -0.3649    0.0030
-0.3659   -0.3673    0.0014
-0.3595   -0.3621    0.0027
-0.3476   -0.3498    0.0021
-0.3293   -0.3301    0.0008
```



شکل ۲-۱۱

دریافت مختصات نقاط منحنی با ginput

با اجرای دستور `[X, Y] = ginput` یک تقاطع موئین ظاهر می‌شود. این تقاطع را با ماوس به هر نقطه گراف برده و کلیک کنیم مختصات آن نقطه در بردارهای X و Y ذخیره می‌شوند. با این‌کار یک یا تعدادی نقطه روی گراف انتخاب می‌کنیم. پس از انتخاب نقاط کلید <Enter> را می‌زنیم. سپس می‌توانیم هریک از بردارهای به‌دست آمده را مطالعه کنیم.

دستور `[X, Y] = ginput(n)` فقط امکان انتخاب n نقطه را می‌دهد.

حل ترسیمی

ابتدا منحنی معادله را رسم و سپس با دستور `ginput()` مختصات نقطه یا نقاط برخورد منحنی با محور x را تعیین می‌کنیم. با اجرای `ginput()` امکان انتخاب تعدادی نقطه روی گراف با کلیک ماوس و قرار دادن مختصات آن‌ها در یک بردار فراهم می‌شود. کلید <Enter> کار انتخاب نقطه‌ها را تمام می‌کند.

مثال ها:

ریشه معادله با روش ترسیمی

```
>> y = 'cos(x)+sin(x)+log(x)';
>> fplot(y,[0.1,2*pi])
                چون log(0) بی‌نهایت است، محور x را از 0.1 شروع کرده‌ایم %
>> [X, Y]= ginput %          روی گراف در محل صفر تابع کلیک می‌کنیم
```

```
X = 0.2830
Y = -0.0015
```

ریشه معادله فوق تقریباً X = 0.2830 است.

ریشه معادله با روش ترسیمی، و تحقیق آن با **fzero()**

```
>> y = 'exp(x)+20*x'
>> ezplot(y)
>> [X, Y] = ginput
X = -0.0601
Y = -0.2715
>> X0 = fzero(y,X)
X0 = -0.0477
>> y0 = exp(X0)+20*X0
y0 = -1.1102e-016
```

۳-۱۱ تمرین

۱- تابع sin(8*a)+sin(9*a) را به صورت inline تعریف و آنرا با fplot() و ezplot() رسم کنید.

۲- ضرائب معادله درجه سه را به یک ام- فایل تابعی با دو آرگومان خروجی رد کرده و ریشه‌ها را با راه‌برد نیوتن برگردانید. تابع را با عدد گذاری به جای آرگومان‌ها از پنجره فرمان اجرا کنید.

۳- ریشه‌های معادله فوق را با استفاده از دستور roots() به دست آورده و با نتیجه بالا مقایسه کنید.

٤- تابع peaks را که از توابع نمونه متلب است از پنجره فرمان اجرا و ساختار آن را ببینید. نام صحیح تابعِ رسمِ آسان سه بعدی را جستجو کنید و تابع peaks را به عنوان آرگومان آن (با رد کردن گیره) رسمِ آسان کنید.

۵- یک فایل C و فایل اجرائی آنرا از یک تابع محاسباتی متلب که خودتان نوشته‌اید، پدید آورید.

٦- یک فایل C++ و فایل اجرائی آنرا از یک تابع گرافیکی متلب که خودتان نوشته‌اید، پدید آورید.

۷- از راه سعی و خطا بزرگ‌ترین عددی را که متلب می‌تواند روی کامپیوتر شما فاکتوریل آنرا برگرداند پیدا کنید.

فصل ۱۲ مباحثی پیرامون رشته ها

۱۲-۱ رشته به مثابه آرایه (بردار)

در متلب رشته، برداری است که عناصر آن از کاراکترهای اسکی تشکیل می‌شوند، علاوه بر توابع و دستورات برداری بعضی توابع خاص نیز در مورد رشته‌ها وجود دارند.

دسترسی به حروف رشته

مثال:

تابعی بنویسید که یک رشته را دریافت و تک تک عناصر آن را با یک فاصله در بین‌شان چاپ کند. آن را اجرا کنید. تابع را با نام bl.m در دیرکتوری جاری ضبط کرده و آن‌را از پنجره فرمان با یک آرگومان دل‌خواه فراخوانی کنید.

یادآوری: بلوک try...catch...end برای به دام انداختن خطا error trapping به کار می‌رود. دستورات زیرِ catch در صورت وقوع خطا اجرا می‌شوند. در این مثال خطا وقتی روی می‌دهد که k از اندیس بالاترین عنصرِ ss بیشتر شده باشد، دراین‌صورت سطر try ss(k) خطا را می‌بیند و دستور زیر catch یعنی break اجرا می‌شود.

```
function = bl(ss)
k = 1;
while 1          % حلقه ادامه‌دار تا بی‌نهایت
 try ss(k);      %  اگر خطا نداشتیم، دستورات زیر اجرا می‌شوند
   fprintf('%c ',ss(k));
   k = k + 1;
 catch           % اگر خطا داشتیم برنامه تمام می‌شود
    break
 end
end
>> bl('Azadeh')
A z a d e h
```

عدد اسکی یک کاراکتر

از توابع single() و double() می‌توان برای نمایش عدد اسکی یک کاراکتر استفاده کرد. تابع char() عدد اسکی را به صورت کاراکتر نمایش می‌دهد. اگر رشته‌ای به توابع single() و double() ارسال شود، کدهای اسکی کاراکترهای آن چاپ می‌شود.

مثال ها:

```
>> single('pi')
ans = 112   105
>> Mt = double('LAB')
Mt = 76    65    66
>> Mt1 = [77    65    84    76    65    66];
>> char(Mt1)
ans = MATLAB
```

رشته m × n

برای ایجاد ماتریس رشته‌ای از تابع char() استفاده می‌شود. با تعریف ماتریس رشته‌ای m × n طول همه ردیف‌ها برابر طول درازترین ردیف می‌شود.

مثال:

یک رشته چند سطری تولید، و توابع اندازه‌گیر را در مورد آن اجرا کنید طول سطر اول را به دست آورید.

```
>> NA = char('Mostafa','Hemmatabadi','Iran')
NA = Mostafa
     Hemmatabadi
     Iran
>> ndims(NA)
ans = 2
>> size(NA)
ans = 3    11
>> length(NA)
ans = 11 % length is max(size())
>> N1 = NA(1,:) % first row
ans = Mostafa
>> length(N1)
ans = 11
```

۲-۱۲ مرتب سازی رشته

یک رشته اگر با کالن (بدون استفاده از گیومه) تعریف شود، کاراکترهای عضو آن کد اسکی (عدد) خواهند بود. مرتب شده‌ی چنین رشته‌ای به صورت عددی نمایش داده می‌شود، برای نمایش آن به صورت کارکتر از تابع char() استفاده می‌کنیم.

مرتب سازی با تابع کتاب خانه ای sort()

مثال ها:

تعریف رشته به صورت آرایه کاراکتری

```
>> xs = ['b' 'd' 'a' 'c']
xs = bdac
>> sort(xs)
ans = abcd
```

تعریف رشته به صورت معمول

```
>> xs = 'bdac'
xs = bdac
>> sort(xs)
ans = abcd
```

رشته به صورت اعداد (اسکی)

```
>> ss = 'z' : -1 : 'r'
 ss = 122   121   120   119   118   117   116   115   114
>> stn = sort(ss)
stn = 114   115   116   117   118   119   120   121   122
>> st = char(stn)
st = rstuvwxyz
```

مرتب سازی رشته با تابع حبابی و مقایسه با sort()

روش bubble sort که برنامه آن را نوشته‌ایم، یکی از کندترین روتین‌های مرتب‌سازی است، اما مرتب سازی داخلی متلب sort() از روتین‌های مدرن‌تر و سریع‌تر مانند quick sort استفاده می‌کند.

مثال:

تابع مرتب سازی حبابی bubble sort را بنویسید. سرعت اجرای آن را با سرعت تابع داخلی متلب sort() مقایسه کنید. تابع مرتب سازی حبابی به طریق زیر نوشته می‌شود:

```
% Function M-File bubbles.m
function y = bubbles(x)
n = length(x);
for k = 1 : n;     % count the passes
 for j = 1 : n - k
    if x(j) > x(j+1)
      temp = x(j); x(j) = x(j+1); x(j+1) = temp; % swap
    end
  end
end
y = x;
```

در ام- فایلی به نام bsrt.m یک آرایه از کاراکترها (یک رشته) که شامل a تا z (کد اسکی 97 تا 122) به طور معکوس است را تعریف می‌کنیم. به این ترتیب، چون نامرتب‌ترین سامانه را دارد، بیشترین عبور (تکرار) برای مرتب سازی آن انجام خواهدشد.

```
% Script M-File bsrt.m, calls the function bubbles.m
clc, clear
x = 'z' : -1 : 'a' ;
xr = repmat(x,1,200);% یک آرایه بزرگ تولید می‌کند
tic
xrs = char(sort(xr));
tc = toc;
fprintf('for MATLAB sort: %f Secs\n',tc)
tic
xrs = char(bubbles(xr)) ;
tc = toc;
fprintf('for bubble sort: %f Secs',tc)
```

```
>> bsrt
for MATLAB sort: 0.060000 Secs
for bubble sort: 2.213000 Secs
```

۱۲-۳ توابع رشته ای

تابع strcmp(s1,s2) و عملگرهای مقایسه ای برای رشته ها

دو رشته را مقایسه می‌کند و در صورت یکسانی کاملِ آن‌ها، true (منطقِ یک) ودر غیر آن false (منطقِ صفر) برمی‌گرداند.

عملگرهای مقایسه‌ای، عناصر رشته‌ها را تک به تک مقایسه می‌کنند، زیرا رشته‌ها، آرایه یا بردار متشکل از کاراکترها هستند. برای مقایسه دو رشته با عملگرهای مقایسه‌ای بایستی طول آن‌ها مساوی باشد. نتیجه مقایسه منطق است.

مثال ها:

```
>> s1 = 'ABC'; s2 = [65 66 67];
>> s3 = char(s2)
>> as = strcmp(s1,s3)
as = 1
>> islogical(as)
ans = 1
>> s1 = 'ABCD';  s2 = 'ABC'
>> strcmp(s1,s2)
```

```
ans =  0
```

```
>> s1 = 'Arman';
>> s2 = 'arman';
>> s1 == s2
ans = 0     1     1     1     1
>> s2 > s1
ans = 1     0     0     0     0
```

جمع کردن رشته‌ها با دستور strcat()

دستور strcat () چند رشته را سرهم (باهم جمع) کرده و حاصل‌جمع را در یک رشته جدید قرار می‌دهد.

strcat () فضاهای خالی انتهای رشته‌ها را حذف می‌کند، اما می‌توان فضاهای خالی را به اندازه دل‌خواه تعریف و داخل رشته قرار داد.

مثال ها:

```
>> s1 = 'My '; s2 = 'Name'; s3 = ' Is';
>> s = strcat(s1,s2,s3)
s = MyNameIs
```

```
>> sp = repmat([' '],1,4);
>> xb = strcat(['Iran', sp, 'My', sp, 'Country'])
xb = Iran    My    Country
```

۴-۱۲ قالب‌بندی رشته String Formatting

تعیین فرمت برای نمایش رشته با fprintf()

با استفاده از fprintf () یک رشته را می‌توان به انحاء مختلف نمایش داد. برای نمایش کاراکترهائی نظیر تقسیم معکوس \ ، و آپوستروف ' در تعریف رشته دو تا از آن ها را می‌آوریم

مثال:

جمله زیر را با fprintf () تولید کنید.

```
    Mathworks'MATLAB
is a poweful language.          Is'nt it?
>> ss1 = 'Mathworks''MATLAB';
>> tt1 = 'Is''nt it?';
>> fprintf('%20s \nis a poweful language.\t%s',ss1,tt1)
```

```
    Mathworks'MATLAB
is a poweful language.          Is'nt it?
```

نگه داری رشته در یک متغیر با sprintf()

همانند fprintf () که خروجی را به صفحه نمایش می‌فرستد، با استفاده از sprintf () می‌توان خروجی را به داخل یک متغیر رشته‌ای فرستاد.

مثال:

```
% M-File fL.m
na = 'New Book';
ex = 345;
nat = na';
```

چون رشته دو بعدی ستون بعد ستون در داخل متغیر قرار می‌گیرد ابتدا آن‌را ترانهاد می‌کنیم %

```
snew = sprintf('%s\n%d',nat,ex)
>> fL
```

```
snew = New Book
       345
```

۵-۱۲ تمرین

۱- رشته dcba را به صورت آرایه عددی (با عناصری از کدهای اسکی) تعریف، وبا تابع کتاب خانه ای sort() مرتب کنید.

۲- توابع اندازه‌گیر را برای رشته mxn NA = char('Self','Teaching','Book') امتحان کنید. طول ردیف‌ها را به دست آورید.

۳- تابعی بنویسید که دو رشته s1 و s2 را به عنوان آرگومان از صفحه کلید دریافت و بدون دانستن طول آن‌ها، رشته درازتر را با رشته کوتاه‌تر هم‌اندازه کرده، و مقایسه را انجام دهد. سپس حروف حذف شده رشته درازتر را با پیغام truncated: نمایش دهد. این تابع را از پنجره فرمان اجرا کنید.

٤- با استفده از تابع sprintf() عبارت زیر را در متغیر snow قرار داده و آن را به دو صورت کد اسکی و کاراکتری نمایش دهید.

```
MATLAB
  Ver 6.5
```

۵- با for برنامه‌ای بنویسید که معکوس رشته ss = 'Good Student' را چاپ کند. راهنما: برای تعیین حد بالائی شمارنده از length(ss) استفاده کنید.

٦- آرایه متشکل از کدهای اسکی xn = [100 99 98 97] را مرتب کرده به صورت کاراکتری نمایش دهید.

فصل ۱۳ سیگنال، سیستم، فیلتر

۱-۱۳ تبدیلات فوریه

تبدیل فوریه گسسته fft()

دستور `Y = fft(y)` تبدیل فوریه گسسته‌ی بردار `y` را که `N` عنصر دارد به دست می‌دهد. `y` و `Y` متساوی‌العنصر هستند. در تئوری مخابرات `y` و `Y` دو بردار مکمل در حوزه زمان و در حوزه فرکانس نام دارند. معمولاً برای آنالیز سیگنال‌های مخابراتی بین میدان زمان و میدان فرکانس آمد و رفت می‌کنیم.

باید دقت داشت که هر سیگنال در MATLAB به‌صورت یک بردار تعریف می‌شود و مجموعه‌ای از نقاط مجزا یا نمونه-برداری شده (sampled) است. فرض کنید سیگنال `y` دارای `N` عنصر است، عبارت `Y = fft(y)`، بردار `Y` را به صورت زیر ایجاد خواهد کرد:

$$Y(k) = \sum_{n=1}^{N} y(n)*\exp(-j*2*pi*(k-1)*(n-1)/N),\ 1 <= k <= N.$$

`Y(k)` عنصر kام بردار `Y` است، چنان‌چه ذکر شد تعداد عناصر `Y` نیز `N` است.

چون `Y(k)` مختلط است، باید قدرمطلق آن‌را با دستور `bar()` یا `stem()` نمایش دهیم.

تبدیل فوریه گسسته وارون ifft(t)

این تابع بردار `Y` را به میدان معکوس می‌برد (در سیگنال‌های مخابراتی از میدان فرکانس به میدان زمان می‌رویم).

عبارت `y = ifft(Y)`، بردار `y` را به صورت زیر ایجاد خواهد کرد:

$$y(n) = (1/N) \sum_{k=1}^{N} Y(k)*\exp(\ j*2*pi*(k-1)*(n-1)/N),\ 1 <= n <= N.$$

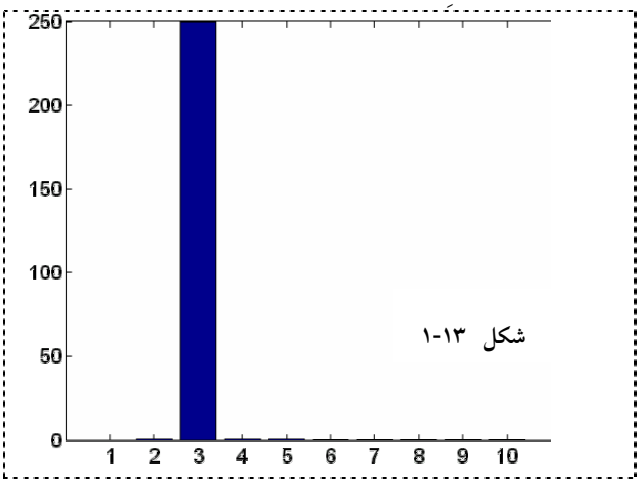
`y(n)` عنصر nام بردار `y` است، چنان‌چه ذکر شد تعداد عناصر `y` مساوی `N` است.

مثال ها:

مشاهده گستره فرکانس یک پالس سینوسی

یک بردار سینوسی شامل 500 درایه را تبدیل فوریه گسسته کرده و طیف فرکانسی آن را تا هارمونیک دهم مشاهده کنید.

```
%ffts
n = 500;
thet = linspace(-2*pi,2*pi,n);
sig = sin(thet);
SIG = fft(sig);
aSIG = abs(SIG) ;
bar(aSIG(1:10),0)
```



شکل ۱۳-۱

سؤال: با تغییر تعداد اعضاء بردار سازنده پالس سینوسی به یک دهم و ده برابر چه تغییری در طیف ایجاد می‌شود

تبدیل فوریه گسسته ی تابع صعودی **Ramp**

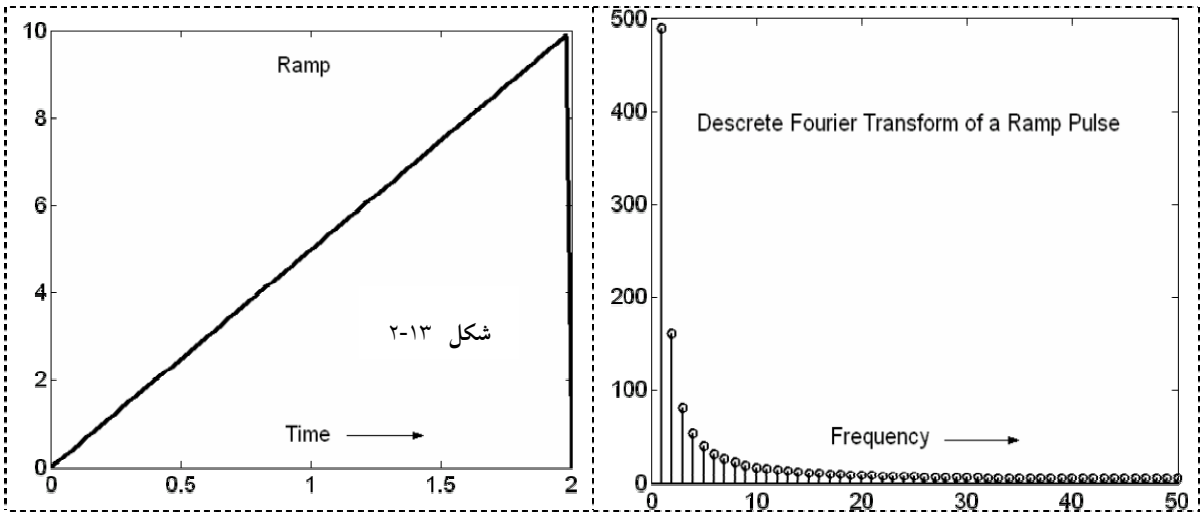
برنامه‌ای برای ایجاد یک تابع صعودی بنویسید، بردار صعودی `y` را به صورت تابع بردار زمان `t` ایجاد و

تبدیل فوریه گسسته کنید. سپس تبدیل وارون کرده دو پالس را مقایسه کنید. نمودارهای مربوطه را رسم کنید.

برای ایجاد شکل ramp تابع زیر را نوشته و در فایلی به نام ramp.m ضبط می‌کنیم.

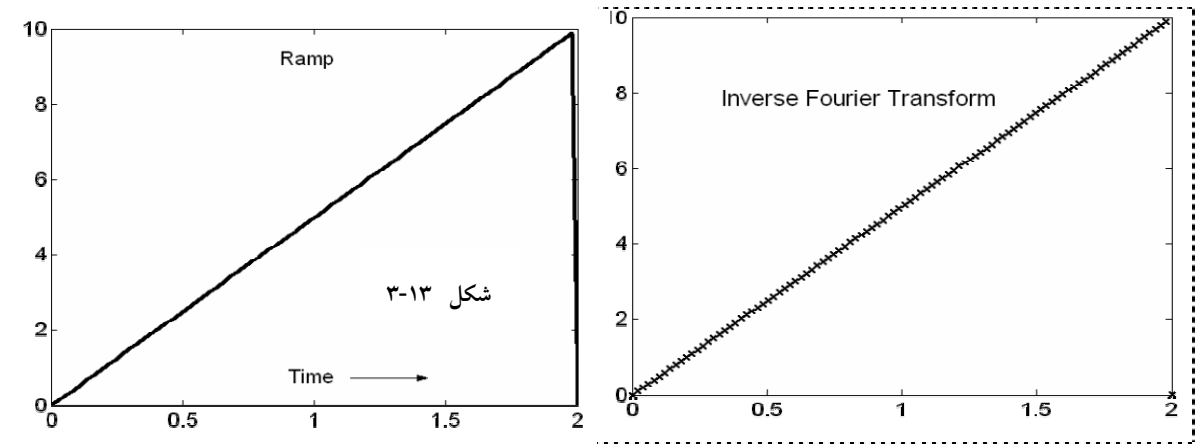
```
function ot = ramp(t,T)
% t is the time duration of the wave and T is its period.
ot = 10*rem(t,T)/T;

% rmp.m
Tp = 2;
n=100
t = linspace(0,Tp);
y = ramp(t,Tp);
plot(t,y);
title('Ramp')
xlabel 'Time'
clf
Y = fft(y);
aY = abs(Y);
stem(aY(1:n/2)),
title 'Descrete Fourier Transform of a Ramp Pulse'
xlabel 'Frequency'
clf
y1 = ifft(Y);
ay1 = abs(y1);
plot(t,ay1,'x')
title 'Inverse Fourier Transform'
```



شکل ۱۳-۲

سؤال: برای نمایش طیف فرکانسی عبارات plot() و bar() را هم امتحان کنید.



شکل ۱۳-۳

۲-۱۳ توابع سیستم ها

دستور ایجاد تابع تبدیل زمان پیوسته tf

این دستور `SYS = tf(num,den)` یک تابع تبدیل زمان پیوسته به نام SYS از کسری با صورت و مخرج num و den پدید می‌آورد. SYS یک شیء از نوع TF خواهد بود. بردارهای num و den از ضرائب s استخراج می‌شوند.

مثال:

بردارهای num و den را از عبارت $\frac{s}{s^2+2s+10}$ استخراج کنید. و تابع تبدیل مربوطه را به‌دست آورید.

```
>> num = [1 0];
>> den = [1 2 10];
>> h = tf(num, den)
```

```
Transfer function:      s
                  --------------
                  s^2 + 2 s + 10
```

ترسیم bode() برای یک سیستم

دستور `bode(SYS)` ترسیم Bode را برای یک سیستم خطی بی‌تغییر با زمان LTI (Linear Time-Invariant) که با `tf()` یا دستورات مشابه آن پدید آمده ایجاد می‌کند. دامنه فرکانس و تعداد نقاط خودبه‌خود تعیین می‌شوند.

واکنش پله ای و واکنش ایمپالسی step(), impulse()

دستور `step(SYS)` واکنش در مقابل پالس پله‌ای و دستور `impulse(SYS)` واکنش در مقابل سیگنال ایمپالس را برای یک سیستم LTI که با `tf()` یا دستورات مشابه آن پدید آمده نمایش می‌دهند. دامنه فرکانس و تعداد نقاط خودبه‌خود تعیین می‌شوند.

دیاگرام نایکوییست Nyquist Diagram

دستور `nyquist(SYS)` دیاگرام مربوطه را برای یک سیستم LTI پیوسته یا گسسته رسم می‌کند. این دیاگرام برای آنالیز سیستم (شامل gain margin، phase margin، و پایداری) به‌کار می‌رود.

تغییر تابع تبدیل به فرم فضای حالت tf2ss

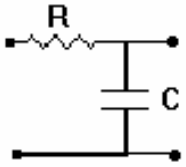
این دستور: `[A,B,C,D] = tf2ss(b,a)` ، پارامترهای فضای حالت را از تابع تبدیل استخراج می‌کند.

مثال ها:

انتگراتور

تابع تبدیل مدار شکل ۱۳-٤ را خودتان پیدا کرده و در محیط MATLAB پدید آورید. ترسیم `bode()` ، دیاگرام نایکوییست، واکنش پله‌ای و واکنش ایمپالسی آن‌را نمایش دهید. با فرض `1/RC = 1`

```
% intr.m
num = [0 1]; den = [1 1];
H = tf(num,den)
subplot(2,2,1),bode(H)
subplot(2,2,2),step(H)
subplot(2,2,3),impulse(H)
subplot(2,2,4),nyquist(H)
```



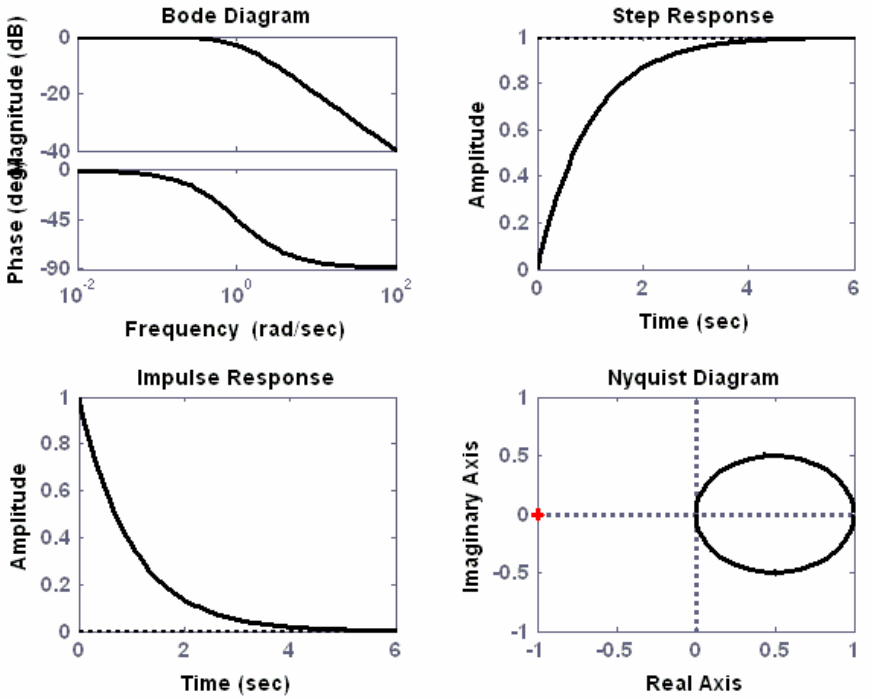
شکل ۱۳-٤

$$\frac{1/RC}{s + 1/RC}$$

```
>> intr
```

```
Transfer function:   1
                   -----
                   s + 1
```

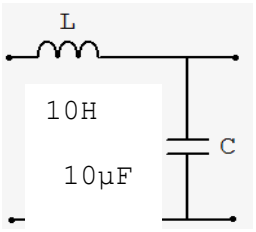
منحنی شارژ خازن متعلق به مدار انتگراتور و پائین‌گذر است



شکل ۱۳-۵

اسیلاتور

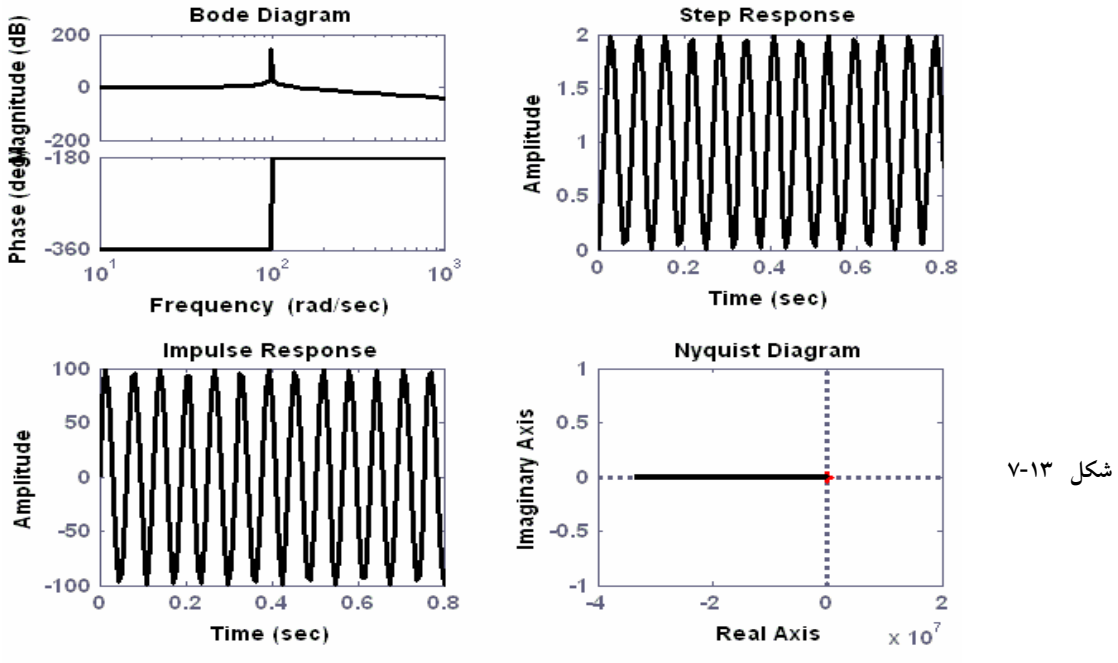
تابع تبدیل زمان پیوسته مدار نوسانی شکل ۱۳-۶ را محاسبه کرده و پدید آورید. ترسیم ( ) bode ، دیاگرام نایکوییست، واکنش پله‌ای و واکنش ایمپالسی آن‌را نمایش دهید. تابع تبدیل مثال فوق را به فرم فضای حالت درآورید.



$$\frac{1/LC}{s^2 + 1/LC}$$

شکل ۱۳-۶

```
% oscil.m
B = [10000];
A = [1 0 10000];
H = tf(B,A);
bode(H),step(H),impulse(H),nyquist(H)
[A,B,C,D] = tf2ss(num,den)
>> oscil
A = -1    B = 1    C = 1    D = 0
```

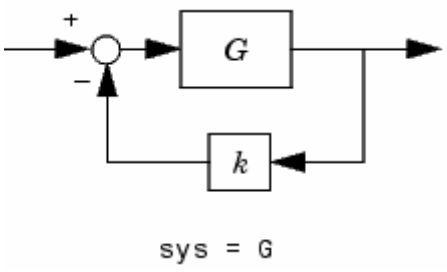


شکل ۱۳-۷

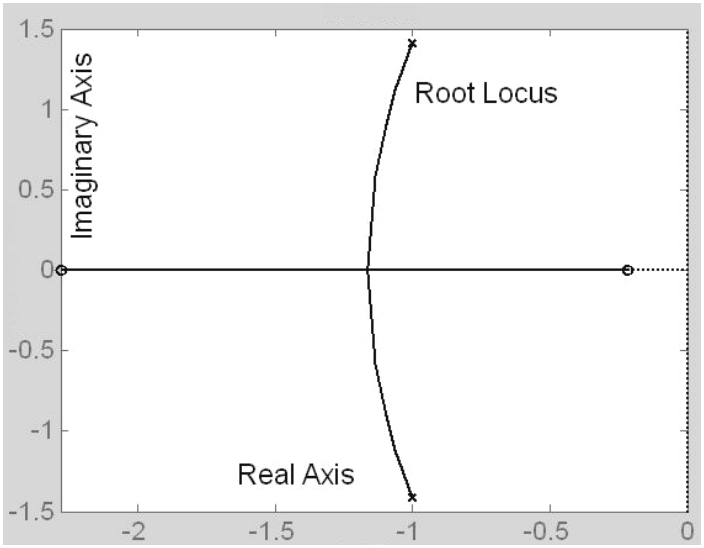
دستور rlocus برای رسم مکان هندسی ریشه ها

این دستور برای بررسی تغییرات ضریب بازخور feedback gains بر روی مکان قطب‌های سیستم‌های حلقه بسته closed-loop مانند سیستم نمونه زیر کاربرد دارد.

مثال:




```
>> B = [2 5 1];
>> A = [1 2 3];
>> h = tf(B, A);
>> rlocus(h)
```



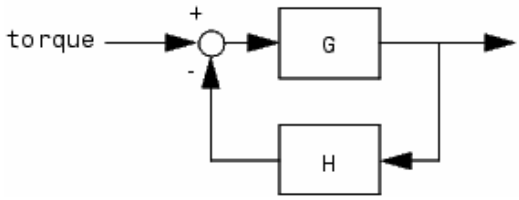
شکل ۱۳-۸

سیستم فیدبک منفی با دستور feedback()

دستور feedback() مدل LTI را برای یک سیستم بازخور منفی به دست می‌دهد.

مثال:

شکل۱۳-۹ مدل LTI یک سیستم کنترل سرعت را نشان می‌دهد.



$$G(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

$$H(s) = \frac{5(s+2)}{s+10}$$

شکل ۱۳-۹

```
G = tf([2 5 1],[1 2 3]);
H = zpk(-2,-10,5);
Cloop = feedback(G,H)
```

```
Zero/pole/gain:
0.18182 (s+10) (s+2.281) (s+0.2192)
-----------------------------------
 (s+3.419) (s^2  + 1.763s + 1.064)
```

۱۳-۳ مدل زمان گسسته Discrete-Time Models

مدل زمان گسسته برای آنالیز سیستم‌هائی که با سیگنال‌های گسسته یا نمونه‌برداشته (Sampled) کار می‌کنند، به‌کار می‌رود. تعیین سیستم برای آن شبیه به سیستم زمان پیوسته است، اما دراین‌جا باید مدت‌زمان نمونه‌برداری (sampling period OR sample time) نیز ذکر شود.

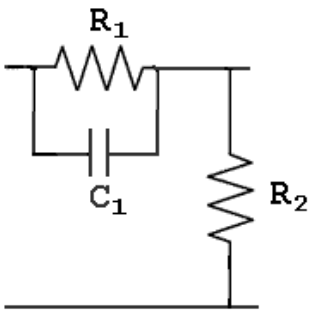
تابع تبدیل زمان گسسته مدار Lead

مثال:

تابع تبدیل زمان گسسته مدار ۱۳-۱۰ را که در کنترل، مدار Lead نام دارد ابتدا محاسبه کرده، سپس پدید آورید.

ترسیم bode () ، دیاگرام نایکوییست، واکنش پله‌ای و واکنش ایمپالسی آنرا نمایش دهید.

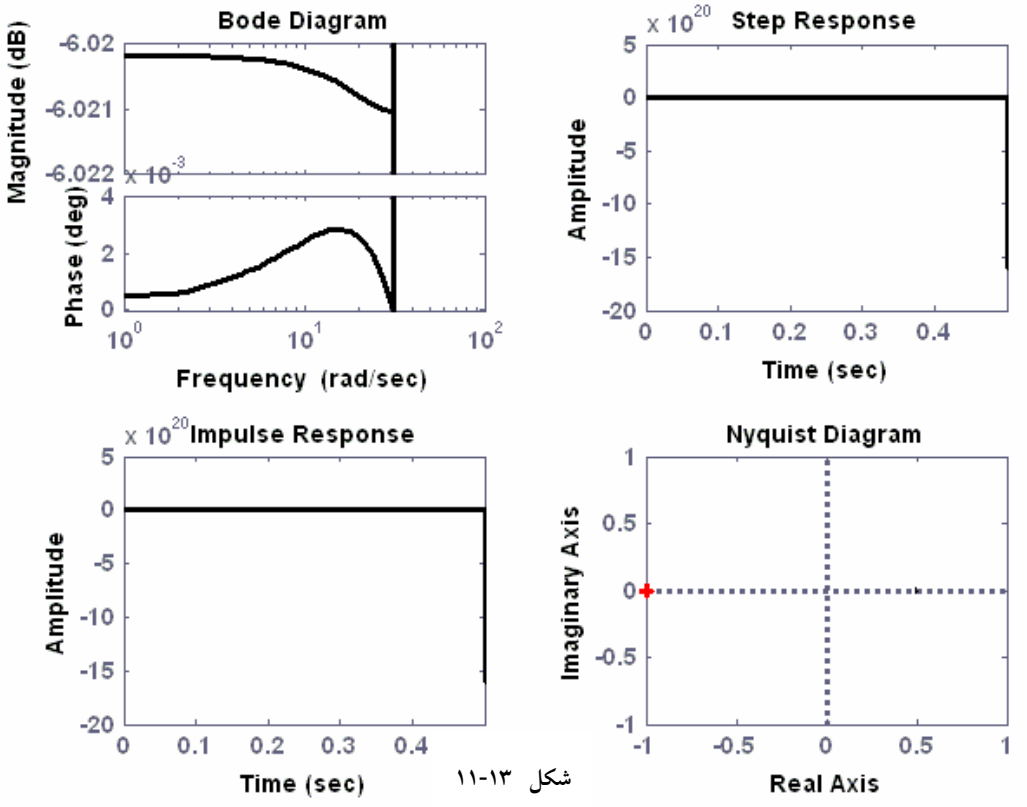
R1 = R2 = 1, C = 100µF, time sample = 0.1



$$\frac{s + (1/R_1C)}{s + (R_1+R_2)/(R_1R_2C)}$$

شکل ۱۳-۱۰

```
% dtm.m
B=[1 10000];
A=[1 20000];
Hz=tf(B,A,0.1)
subplot(2,2,1),bode(Hz, 'k')
subplot(2,2,2),step(Hz, 'k')
subplot(2,2,3),impulse(Hz, 'k')
subplot(2,2,4),nyquist(Hz, 'k')
>> dtm
```



شکل ۱۳-۱۱

به منحنی تغییر فاز **(lead)** . واکنش پله‌ای، و ایمپالسی در مدل گسسته توجه کنید.

۱۳-۴ فیلترها

فیلتر Butterworth آنالوگ و دیجیتال، دستور butter()

دستور `[B,A] = butter(N,w,'s')` یک فیلتر آنالوگ پائین‌گذر درجه N باترورث با فرکانس تقطیع `w rad/sec` را ایجاد می‌کند. B و A به ترتیب صورت و مخرج تابع تبدیل فیلتر هستند.

دستور `[B,A] = butter(N,[w1 w2],'s')` یک فیلتر میان‌گذر مابین w1 و w2 را ارائه می‌دهد،

و دستور [B,A] = butter(N,w,'high','s') یک فیلتر آنالوگ بالاگذر ایجاد می‌کند.

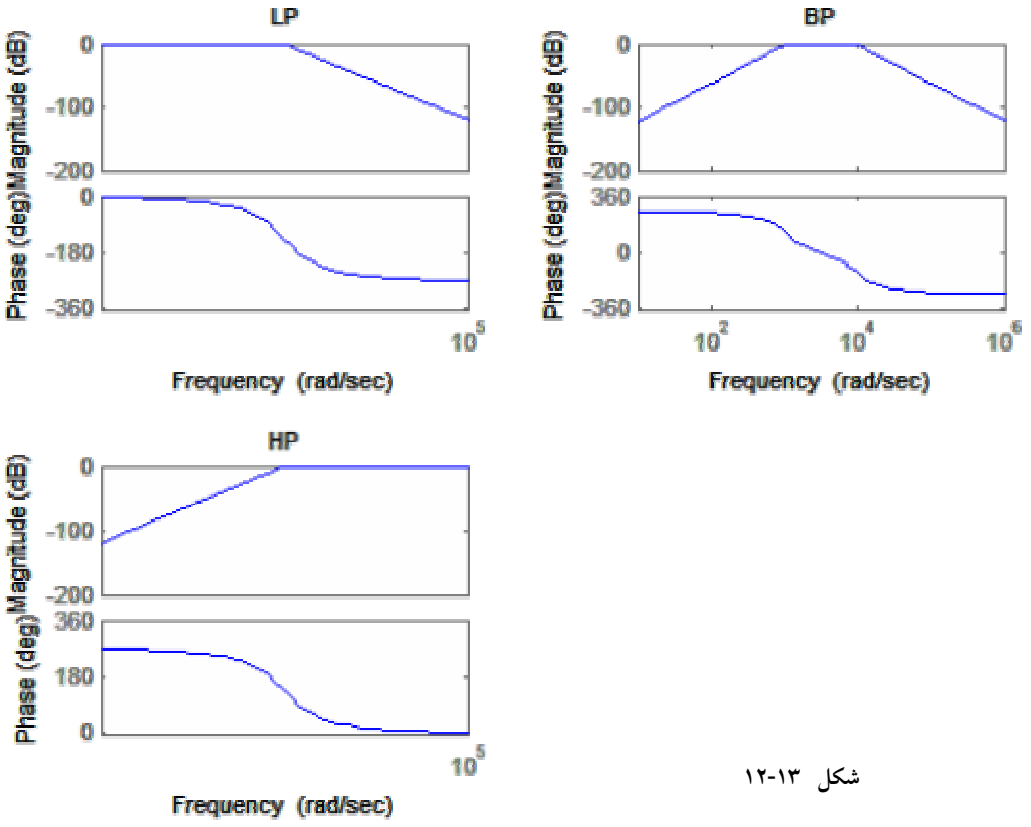
دستور [B,A] = butter(N,Wn) یک فیلتر باترورث پائین‌گذر دیجیتال درجه N با فرکانس قطع Wn طرح می‌کند. دامنه فرکانس قطع 0.0 < Wn < 1.0 است. اگر Wn = 1 بگیریم معادل نصف سرعت نمونه‌برداری sample rate خواهد بود.

مثال ها:

فیلتر آنالوگ پائین گذر باترورث

در یک ام- فایل یک فیلتر آنالوگ پائین‌گذر باترورث درجه 3 با فرکانس تقطیع 1000 rad/sec،
یک فیلتر آنالوگ میان‌گذر درجه 3 با فاصله عبور 1000 rad/sec تا 10000 rad/sec،
و یک فیلتر آنالوگ بالاگذر درجه 3 با فرکانس تقطیع 1000 rad/sec
ایجاد کنید. و نمودار فرکانسی آن‌ها را رسم نمائید.

```
% agfil.m
echo off
[B,A] = butter(3,1000,'s');
L = tf(B,A);
clf, bode(L),title('LP'), pause
[B,A] = butter(3,[1000 10000],'s');
B = tf(B,A);
clf, bode(B),title('BP'), pause
[B,A] = butter(3,1000,'high','s');
H = tf(B,A);
clf, bode(H),title('HP'), clf
```



شکل ۱۲-۱۳

فیلتر پائین گذر دیجیتال

یک فیلتر پائین‌گذر دیجیتال درجه 3 با فرکانس قطع 0.3 طرح کنید.

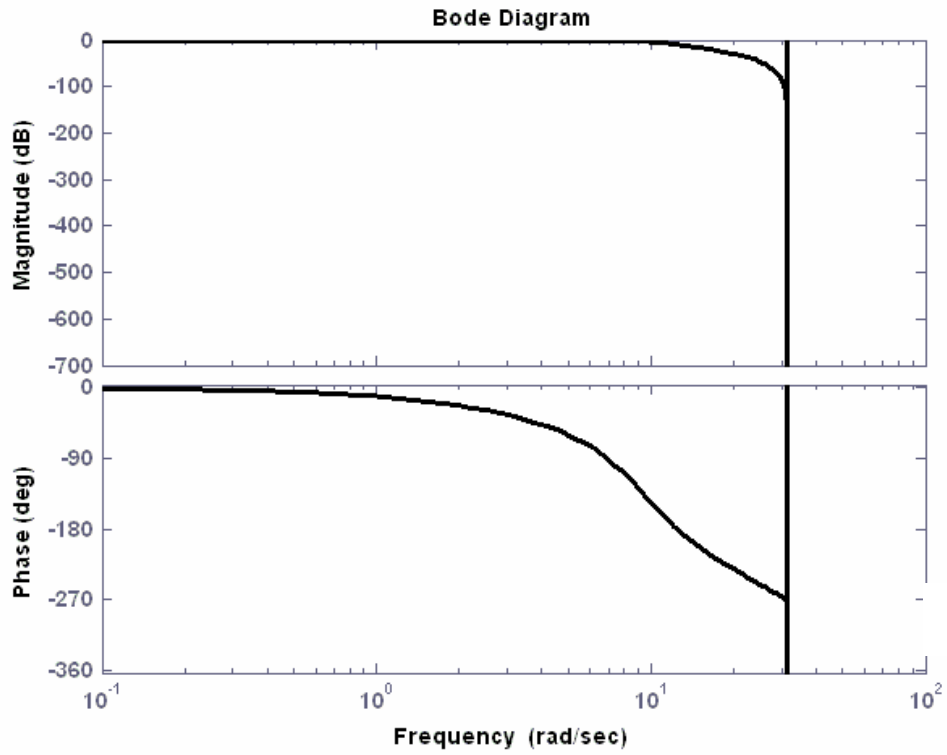
```
>> [B,A] = butter(3,0.3)
```

```
B = 0.0495    0.1486    0.1486    0.0495
A = 1.0000   -1.1619    0.6959   -0.1378
```

```
>> Hb = tf(B,A,0.1)
```

```
Transfer function:
0.04953 z^3 + 0.1486 z^2 + 0.1486 z + 0.04953
----------------------------------------------
     z^3 - 1.162 z^2 + 0.6959 z - 0.1378
Sampling time: 0.1
```

```
>> bode(Hb)
```



شکل ۱۳-۱۳

۱۳-۵ تمرین

۱- تبدیلات فوریه برای پالس مربعی شکل را بنویسید و ترسیم کنید. ترسیم طیف فرکانسی را با `bar(aF)` و `plot(aF)` نیز امتحان کنید. **راهنما:** تبدیل فوریه در حوزه فرکانس به شکل `sin(x)/x` است.

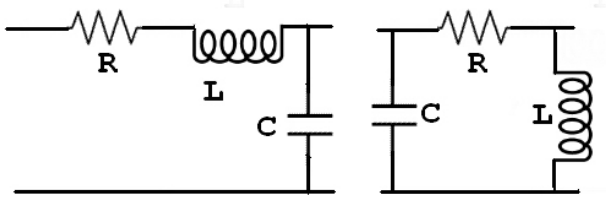
۲- تبدیلات فوریه را برای پالس مثلثی شکل بنویسید و ترسیم کنید.

۳- تبدیلات فوریه را برای شکل موج زیر بنویسید و ترسیم کنید

```
a = linspace(-pi,pi); x = sin(a) - sin(4*a);
```

٤- تابع تبدیل زمان گسسته و زمان پیوسته دو مدار زیر را پدید آورید. ترسیم `bode()`، دیاگرام نایکوییست، واکنش پله‌ای و واکنش ایمپالسی آن‌ها را نمایش دهید

`L = 1 mH, C = 100µF, time sample = 0.1 sec`



۵- فیلترهای پائین‌گذر و بالاگذر آنالوگ و دیجیتال باتروورث درجه4 با فرکانس تقطیع `10000 rad/sec`، و میان‌گذر مابین `1000 rad/sec` و `10000 rad/sec` ایجاد کنید. و نمودار فرکانسی آن‌ها را رسم نمائید.

فصل ۱۴ واسط گرافیکی کاربر

۱۴-۱ واسط گرافیکی کاربر graphical user interface (GUI)

با دستور guide مخف graphical user interface development environment ابزار تولید GUI در اختیار قرار می‌گیرد، که شامل دو پنجره مهم است:

Layout Editor(LE), User Interface Controls (uicontrols)

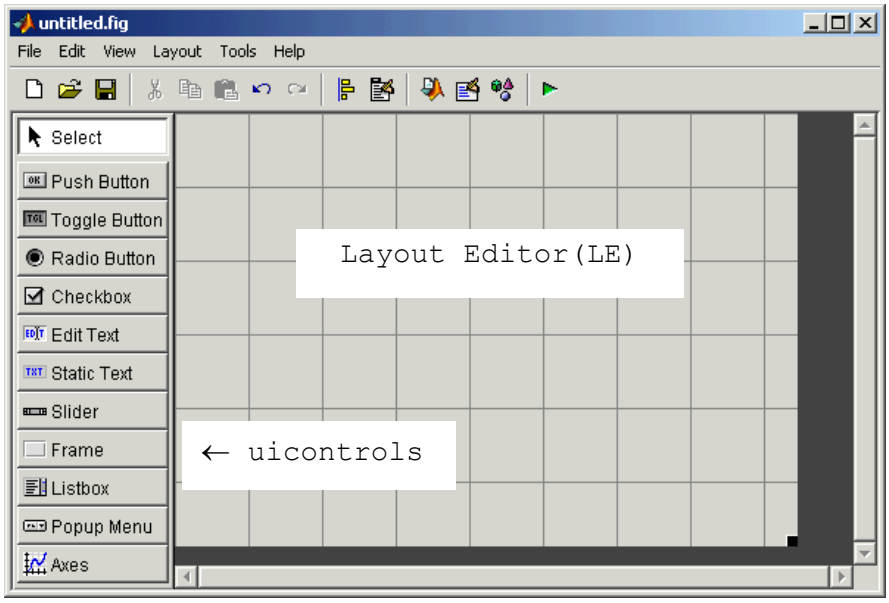
در پنجره اول چهارچوب اصلی GUI را که پس از اجرا مشابه پنجره‌های استاندارد محیط Windows خواهد بود، طراحی می‌کنیم. پنجره دوم یک میله ابزار tools bar است که دکمه‌کنترل‌های ضروری جهت یک برنامه GUI را فراهم می‌کند. روش کار بدین‌گونه است که ابتدا دکمه‌های لازم برای یک برنامه به محوطه LE منتقل می‌شود، سپس به هر دکمه وظیفه خاص خودش از طرق یک زیر برنامه مربوط به آن دکمه که Callback Function نام دارد محول می‌شود. البته در بین راه اعمال فرعی دیگری نظیر تعیین شاخصه‌های properties دکمه‌ها، و پنجره‌ها نیز انجام می‌شود. به مثال‌های زیر و توضیحات مربوطه توجه کنید.

مثال:

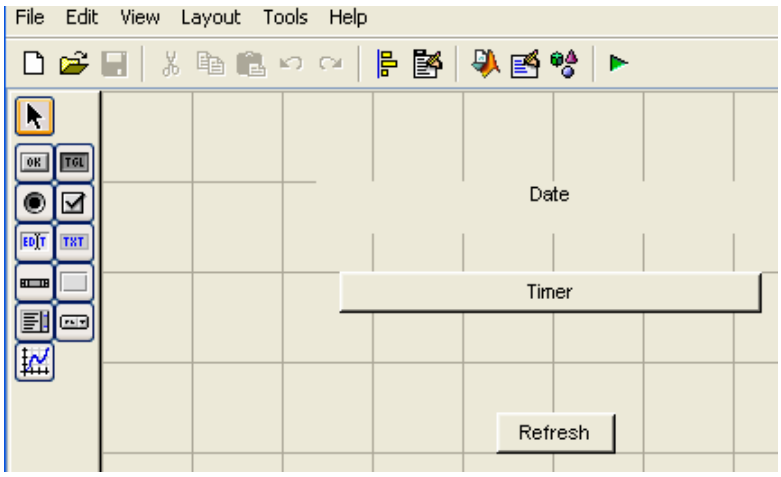
پنجره‌ای بسازید شامل: ۱- یک دکمه فشاری push button که روی آن کلمه Time نوشته شده باشد و پس از کلیک، ساعت را روی خودش نمایش دهد.۲- یک دکمه متن ایستا Static text button که روی آن Date نوشته باشد و پس از فشردن دکمه فشاری قبلی تاریخ را نشان دهد. ۳- یک دکمه فشاری که روی آن Refresh نوشته باشد و دو دکمه قبل را به حالت اول برگرداند.

حل:

ابتدا دستور guide را ازپنجره فرمان اجرا می‌کنیم. پس از ظهور ابزارهای مربوطه (شکل ۱۴-۱) سه دکمه خواسته شده‌ی فوق را از میله ابزار uicontrols انتخاب و به داخل LE می‌آوریم پس از بعضی ویرایش‌ها که ذکر خواهند شد، شکل ۱۴-۲ حاصل می‌شود.



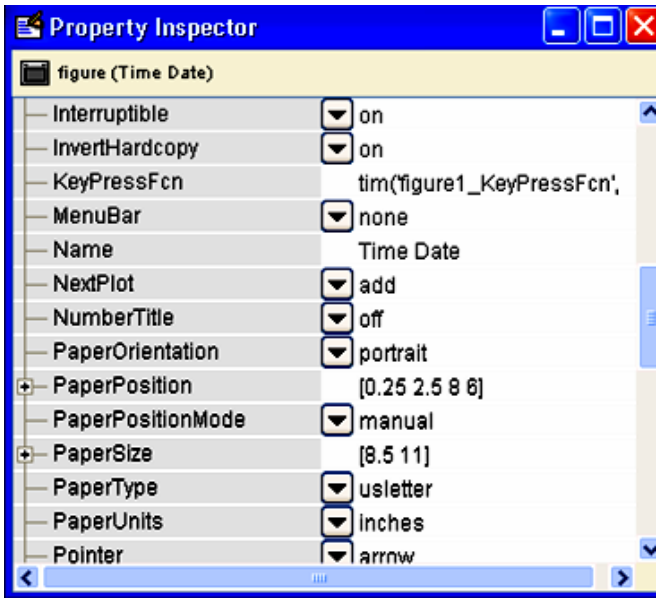
شکل ۱۴-۱



شکل ۱۴-۲

شاخصه یاب property inspector

روی LE ، راست کلیک و از منوی فوری که باز می‌شود انتخاب شاخصه‌یاب property inspector را کلیک می‌کنیم. پنجره مربوطه باز می‌شود.



شاخصه‌ی عنوان ←

شکل ۱۴-۳

شاخصه عنوان Name or String

این شاخصه برای بعضی از اشیاء String و برای بعضی Name گفته شده و روی شیئ حک می‌شود. برای کار با این شاخصه از مثال فوق ۱- در شاخصه‌یاب شاخصه Name را انتخاب و مقابل آن کلمه Time Date را می‌نویسیم. نام پنجره‌ی حاصل پس از اجرا Time Date می‌شود. ۲- دکمه متن را فعال کرده از شاخصه‌یاب برای شاخصه‌ی String کلمه Date را می‌نویسیم. این کلمه روی دکمه متن حک می‌شود. ۳- دکمه فشاری اول را فعال کرده از شاخصه‌یاب برای شاخصه‌ی String کلمه Timer را می‌نویسیم. این کلمه روی دکمه حک می‌شود. ٤- دکمه فشاری دوم را فعال کرده از شاخصه‌یاب برای شاخصه‌ی String کلمه Refresh را می‌نویسیم. این کلمه روی دکمه حک می‌شود.

شاخصه برچسب Tag

شاخصه برچسب Tag نام شیئ است که در برنامه با این نام به شیئ مراجعه می‌شود. شاخصه برچسب را برای هر چهار شیئ، در پنجره شاخصه‌یابِ هرکدام ملاحظه کنید، اما نیازی به تغییر دادن آن‌ها نیست.

بعضی از شاخصه‌های مثال:

شیئ		عنوان String or Name (ادیت شده)	برچسب Tag (ادیت نشده)
پنجره اجرا	Runtime window	Time Date	figure1
دکمه متنِ ایستا	Static text	Date	text1
دکمه فشاری۱	Push Button1	Time	pushbutton1
دکمه فشاری۲	Push Button2	Refresh	pushbutton2

توابع فراخوان Callback Function

در داخل برنامه برای عمل یا اعمالی که با فشردن یک دکمه یا کلیک روی یک پنجره انجام می‌شود تابعی می‌نویسیم که تابع فراخوان آن پنجره یا آن دکمه Callback Function گفته می‌شود.

برنامه نویسی

در پنجره LE انتخاب Run را از میله ابزار یا از منوی Tools اجرا می‌کنیم. یک ام- فایل برای نگه‌داری دستورات برنامه باز می‌شود. نام آن را tim.m می‌گذاریم. نام LE هم خودبه‌خود tim.fig می‌شود. هردو فایل tim.m و tim.fig در دیرکتوری جاری ذخیره می‌شوند. حال برای برنامه نویسی به داخل tim.m می‌رویم. در متن تابع فراخوان دکمه فشاری۱ (دارای برچسب pushbutton1) برنامه زیر را وارد می‌کنیم. به دستورها و سطرهای راهنما comments که در داخل ام- فایل به صورت خودکار نوشته شده کاری نداریم.

```
function pushbutton1_Callback(hObject, eventdata, handles)
t = clock;
d = date;
tm = sprintf('%2.0f: %2.0f: %2.0f:',t(4),t(5),t(6));
```

دو رقم از رشته‌های ساعت، دقیقه و ثانیه را داخل متغیر رشته‌ای tm می‌نویسد %

```
dt = sprintf('%12s',d);
```

متغیر dt را با ۱۲ مکان ایجاد و رشته‌ی تاریخ را در آن می‌نویسد %

```
set(gcbo,'String',tm)
```

شاخصه‌ی عنوانِ شیئ مربوط به تابع gcbo = get callback object را tm (زمان) قرار می‌دهد. %

```
set(handles.text1,'String',char(dt))
```

شاخصه‌ی عنوان شیئ text1 (دکمه متن) را به dt (تاریخ) تغییر می‌دهد. %

در متن تابع فراخوان دکمه فشاری۲ (دارای برچسبِ pushbutton2) برنامه زیر را وارد می‌کنیم.

```
function pushbutton2_Callback(hObject, eventdata, handles)
set(handles.pushbutton1,'String','Time')
```

رشته روی دکمه فشاری ۲ را به کلمه Time تغییر می‌دهد %

```
set(handles.text1,'String','Date')
```

رشته روی دکمه متن را به کلمه Date تغییر می‌دهد %

برنامه‌ی GUI فوق از ۳ طریق قابل اجرا است:

- در پنجره LE با انتخاب Run از میله ابزار
- در پنجره LE با انتخاب Run از منوی Tools
- از داخل ام- فایل به طریق معمول اجرای ام- فایل‌ها

هر برنامه GUI دارای دو فایل است، یکی با پسوند fig . که گرافیکی است و ساختار پنجره خروجی و اشیاء داخل آن را دربر می‌گیرد، و دیگری با پسوند m . (ام- فایل) که رفتارهای هر شیئ را به صورت برنامه code نگه می‌دارد.

اجرای GUI و باز کردن قسمت گرافیکی آن برای ادیت، دو عمل متفاوت هستند.

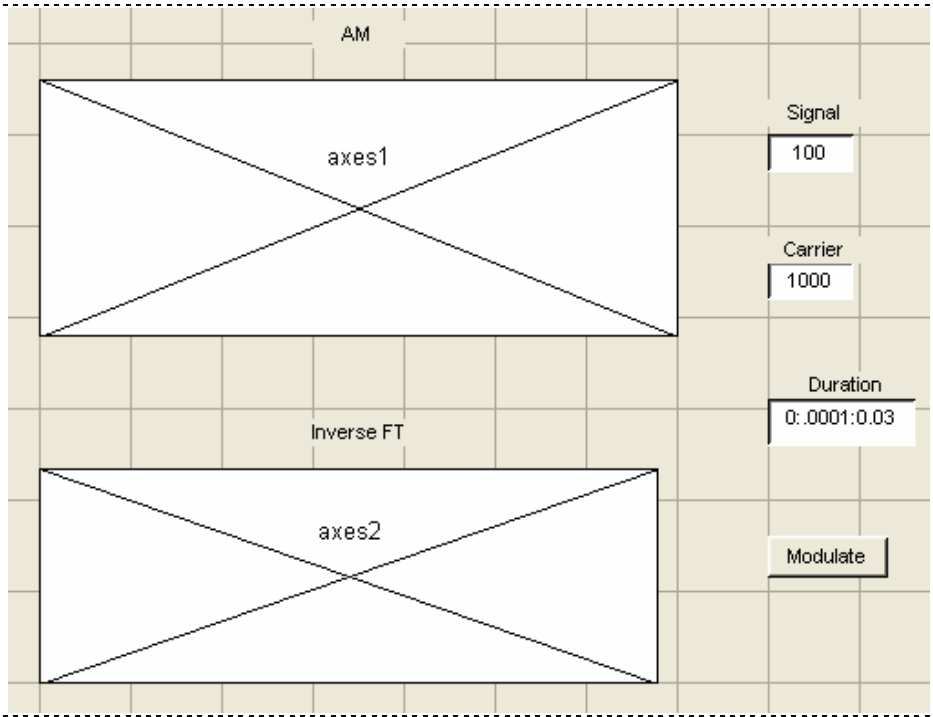
سؤال: پنجره GUI مثال فوق را باز کنید (قسمت گرافیکی یک GUI با اجرای guide و انتخاب شستی open existing GUI برای ادیت شدن باز می‌شود). سپس از منوی Tools بر انتخاب GUI Options کلیک کرده در پنجره دیالوگ باز شده مقابل عنوان Resize behavior: انتخاب Proportional را بیاورید. سپس پنجره را اجرا کنید. این‌بار اندازه پنجره خروجی ایجاد شده قابل تغییر است، با بردن ماوس به گوشه‌های آن و فشرده- کشیدن، ابعاد پنجره را تغییر دهید.

مثال:

شبیه سازی مدولاسیون دامنه

یک مدولاسیون دامنه را مطابق شکل، شبیه سازی کنید. فرکانس Signal (پیش‌فرض ۱۰۰) روی فرکانس Carrier (پیش‌فرض ۱۰۰۰) مدوله (AM) می‌شود. دامنه و گام زمان زیر متن ایستای Duration می‌آید. فشردن دکمه Modulate شبیه‌سازی را انجام می‌دهد. محور axes1 شکل‌موج مدوله را نمایش می‌دهد. سپس در داخل برنامه (که شرح آن در زیر آمده) سری فوریه این شکل موج تعیین و دوباره تبدیل فوریه معکوس انجام می‌شود که بایستی نتیجه آن، شکل موج اولیه باشد. نتیجه روی axes2 نمایش داده می‌شود.

LE را مطابق شکل ۱۴-٤ می‌سازیم:



شکل ۱۴-٤

جدول زیر شاخصه‌های اشیاء را به دست می‌دهد:

شیئ	عنوان String or Name (ادیت شده)	برچسب Tag (پیش‌فرض)
LE	TwoA	figure1
Static Text	۵ دکمه متن ایستا را مطابق شکل نام گذارید	text1...text5
Edit	100, 1000, 0:0.0001:0.03	edit1...edit3
Axes		axes1,axes2
Push Button	Modulate	pushbutton1

در پنجره LE انتخاب Run را از میله ابزار یا از منوی Tools اجرا می‌کنیم. یک ام- فایل برای نگه‌داری دستورات برنامه باز می‌شود. نام آن را TwoA.m می‌گذاریم. نام LE هم خودبه‌خود TwoA.fig می‌شود. هردو فایل در دیرکتوری جاری ذخیره می‌شوند. حال برای برنامه نویسی به داخل TwoA.m می‌رویم. در متن تابع فراخوان pushbutton1 برنامه زیر را وارد می‌کنیم. به دستورها و سطرهای راهنما comments که در داخل ام- فایل به صورت خودکار نوشته شده کاری نداریم.

```
function pushbutton1_Callback(hObject, eventdata, handles)
f1 = str2double(get(handles.edit1,'String'));
```
نوشته روی دکمه با برچسب edit1 را به عدد تبدیل و در f1 می‌ریزد. %
```
f2 = str2double(get(handles.edit2,'String'));
t = eval(get(handles.edit3,'String'));
```
نوشته روی دکمه edit3 را به یک ماتریس عددی تبدیل می‌کند %
```
omg1 = 2*pi*f1;
omg2 = 2*pi*f2;
yt = cos(omg1*t) .* cos(omg2*t);
axes(handles.axes1) % محور اولی فعال می‌شود
plot(omg1*t, yt)
set(handles.axes1,'XMinorTick','on') % درجه گذاری ریز محور افقی را آشکار می‌کند
grid on
yf = fft(yt);% سری فوریهِ شکل موج تعیین می‌شود، ضرائب این سری اعداد موهومی هستند
yt1=real(ifft(yf));% معکوس سری فوریه برای مقادیر حقیقی ضرائب فوریه تعیین می‌شوند
axes(handles.axes2) % محور دومی فعال می‌شود
plot(omg1*t, yt1)
set(handles.axes2,'XMinorTick','on') % درجه گذاری ریز محور افقی را آشکار می‌کند
grid on
```

محورین۱ (axes1) موج مدوله را نشان می‌دهد. برای نمایش روی محورین۲ (axes2) ابتدا از موج AM با تابع fft() تبدیل فوریه مُقَطَّع (سری فوریه) گرفته‌ایم. سپس قسمت حقیقی ضرائب فوریه را با تابع real() پیدا کرده و با تابع ifft() تبدیل فوریه معکوس انجام داده‌ایم. بایستی شکل موج اولیه دو باره به دست آید. نتیجه اجرا پس از یک بار فشردن دکمه Modulate شکل ١٤-۵ است.

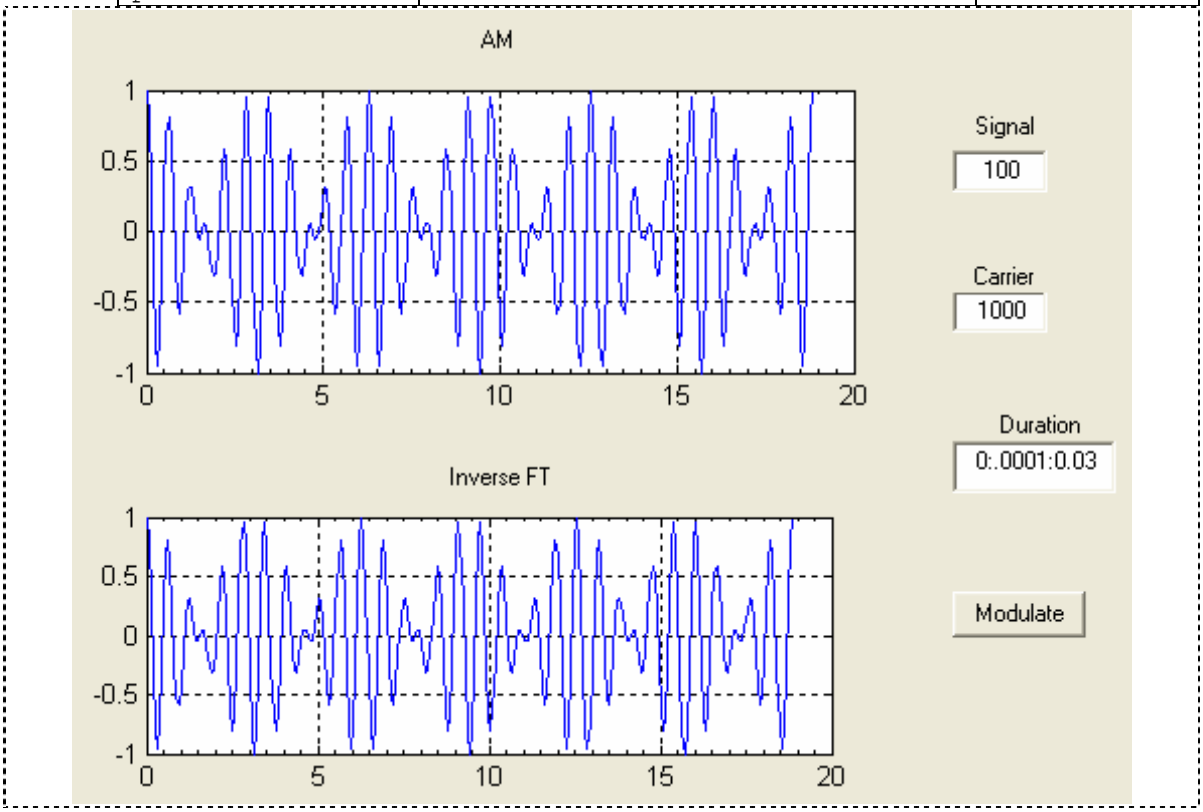
سؤال: مقادیر سیگنال، کریر، و فاصله زمانی را چند بار تغییر دهید و نتیجه را با فشردن دکمه Modulate ببینید.

۱۴-۲ تمرین

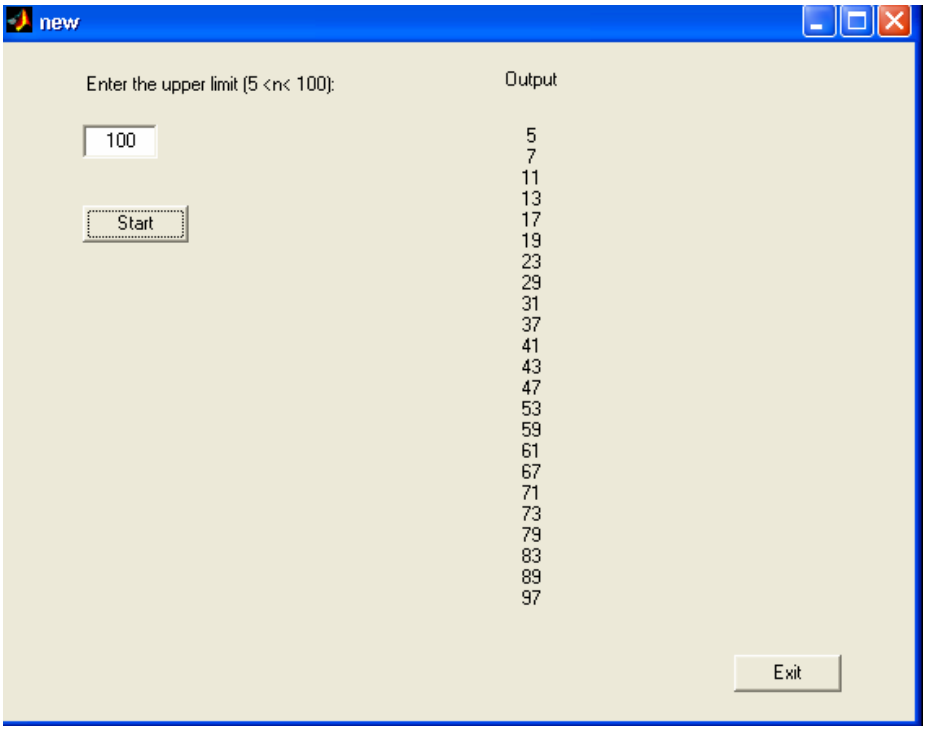
یک GUI به نام new (مطابق شکل ١٤-٦) طراحی کنید که با فشردن دکمه Start اعداد اول را نمایش دهد. و با فشردن دکمه Exit برنامه تمام و پنجره بسته شود. مجموعه اشیاء این GUI در جدول آمده است.

راهنما: برای بستن پنجره از دستور close(handles.new) استفاده شود.

برچسب Tag (پیش‌فرض)	عنوان String or Name (ادیت شده)	شیئ
tex1	Enter the upper limit(5<n<100)	Static Text
edit1	100	Edit
pushbutton1	Start	Push Button
text2	Output	Static Text
text3		Static Text
pushbutton2	Exit	Push Button



شکل ١٤-٥



شکل ١٤-٦

بخش دو EXCEL

فصل ۱۵ نکاتی پیرامون صفحه گسترده

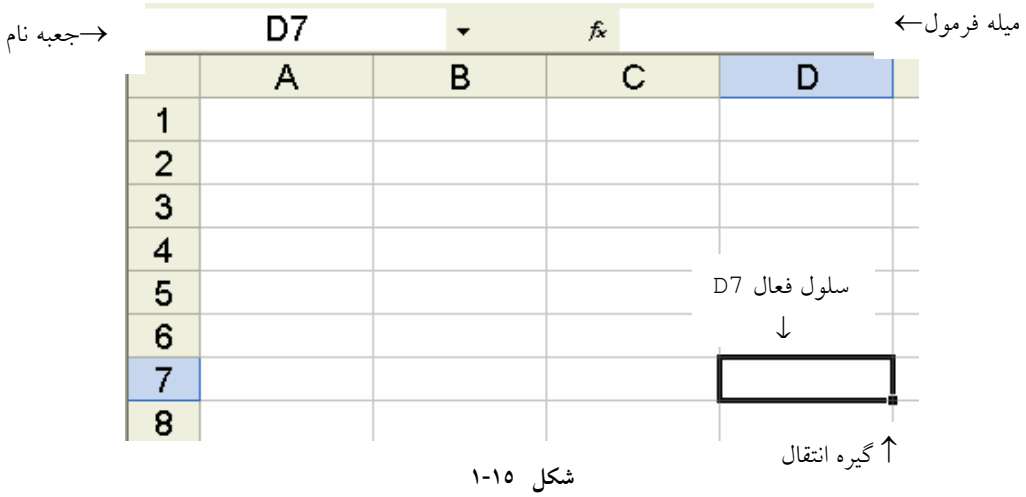
۱۵-۱ برچسب گذاری

برچسب پیش فرض یک سلول

پس از اجرای برنامه اکسل، صفحه گسترده به صورت یک جدول پیش روی ما ظاهر می‌شود. هر خانه این جدول یک سلول نامیده می‌شود. برچسب ستون‌ها با حروف بزرگ انگلیسی در بالای جدول و برچسب ردیف‌ها با عدد در منتهاالیه سمت چپ جدول آمده‌اند. برچسب پیش‌فرض هر سلول از تقاطع نام ستون و ردیف آن سلول حاصل‌شده و برچسب ستون- ردیف گفته می‌شود. برچسب ستون- ردیف هر سلول مشابه نام متغیر در زبان‌های برنامه نویسی است. مربع کوچک در گوشه یک سلول فعال (سلولی که روی آن کلیک شده) گیره انتقال `fill handle` نام دارد، و محتویات یک سلول را به طور نسبی به اطراف می‌کشد. برای اطلاع بیشتر به مطالعه مباحث ادامه دهید.

مثال:

برای فعال کردن سلول D7 ابتدا روی آن کلیک می‌کنیم. برچسب ستون- ردیفِ پیش‌فرض هر سلول در جعبه‌ی نام نشان داده می‌شود.



شکل ۱۵-۱

برچسب گذاری دل خواه، متغیر و آرایه در اکسل

می‌توانیم برچسب‌های دلخواهی را برای یک ستون یا یک ردیف در محلی از جدول وارد کنیم. و سپس در ارجاعات از برچسب‌های اختصاصی خودمان استفاده کنیم. سلول محل تقاطع برچسب‌های دلخواه نام ستون- ردیف جدید را خواهد گرفت. برای فعال کردن برچسب گذاری دلخواه بایستی ابتدا انتخاب زیر را تیک بزنیم:

`Tools_Options_Calculation_Accept labels in formulas`

اگر یک سلول را مشابه یک متغیر در نظر بگیریم، نام پیش‌فرض متغیر همان برچسب ستون- ردیف آن است. البته با برچسب گذاری و نام‌گذاری می‌توان نام متغیر (سلول) را به دلخواه تعیین کرد. اما نام پیش فرض همیشه معتبر می‌ماند و قابل مراجعه است. بر چسب یا نام سلول در جعبه نام نشان داده می‌شود.

می‌توان مقدار سلول را یک رشته قرار داد. از رشته‌ها برای توضیح مطالب یا برچسب‌گذاری یا نام‌گذاری (توضیح نام-گذاری بعداً می‌آید) استفاده می‌شود. رشته ای که مقدار سلول است در داخل آن نمایش داده می‌شود.

می‌توان مقدار سلول را یک عدد قرار داد. این عدد در داخل سلول نمایش داده می‌شود.

اگر مقدار سلول نتیجه یک محاسبه باشد، مقدار در محل سلول نشان داده می‌شود. فرمولِ محاسبه در میله فرمول نشان داده می‌شود.

وقتی به یک ستون مراجعه کنیم نام ستون مانند نام آرایه‌ای عمل می‌کند که اعضاء آن آرایه سلول‌های واقع در بین نام ستون و محل فرمول است

مثال ها:

مراجعه به نام متغیر (سلول)

یک برچسب ستون در محل C3 و یک برچسب ردیف در محل B6 وارد می‌کنیم. در سلول محل تقاطع آن دو یعنی C6 (که از این پس با برچسب Cnam Rnam نیز قابل ارجاع است) عدد 25 را وارد می‌کنیم. سپس به سلول A8 رفته با ارجاع به سلول Cnam Rnam دو برابر مقدار آن را در A8 قرار میدهیم. وضعیت میله فرمول و جعبه نام را بررسی می‌کنیم.

همان‌طور که در شکل ۱۵-۲ می‌شود، سلول A8 را مساوی برچسب جدیدِ C6 (یعنی Cnam Rnam) ضرب در دو قرار داده‌ایم. اگر در محل A8 فرمولِ 2*C6 را هم وارد کنیم همین نتیجه حاصل می‌شود.

جعبه نام ←

میله فرمول ←

A8 | = 2*Cnam Rnam

	A	B	C	D
1				
2				
3			Cnam	
4				
5				
6		Rnam	25	
7				
8	50			
9				

شکل ۱۵-۲

در زیرِ Cnam سه عدد وارد می‌کنیم، حاصل‌جمع آن‌ها را با استفاده از تابع کتاب‌خانه‌ای SUM() در C7 قرار می‌دهیم. پس از وارد کردن اعداد در زیر Cnam به C7 می‌رویم و عبارت =SUM(Cnam) را وارد می‌کنیم، مجموع اعداد ستون Cnam در این سلول نمایش داده می‌شود.

C7 | =SUM(Cnam)

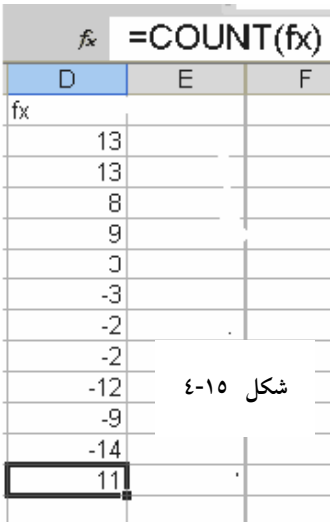
	A	B	C	D
1				
2				
3			Cnam	
4			6	
5			12	
6		Rnam	25	
7			43	

شکل ۱۵-۳

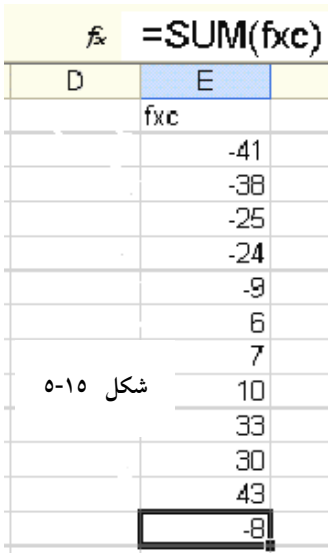
مراجعه به نام و عناصر یک ستون (آرایه)

تعداد عناصر (سلول‌های) متعلق به ستون fx شکل زیر را با استفاده از برچسب گذاری در سلول بعد از آخرین مقدار قرار می‌دهیم، از تابع COUNT() که مخصوص شمارش است استفاده می‌کنیم. شکل ۱۵-٤

مجموع عناصر (سلول‌های) متعلق به ستون fxc شکل ۱۵-۵ را با استفاده از برچسب گذاری در سلول بعد از آخرین مقدار قرار می‌دهیم.



شکل ۱۵-۴



شکل ۱۵-۵

می‌توان از عنوان fxc که در بالای ستون E آمده برای برچسب‌گذاری آن ستون استفاده و سپس در فرمول‌ها به آن رجوع کرد.

یادآوری می‌کنم برای این که برچسب‌های دلخواه *قابل ارجاع* شوند باید از منیوی اصلی انتخاب زیر تیک زده شود.

`Tools_Options_Calculation_Accept labels in formulas`

کتابخانه داخلی اکسل

همان‌طور که در تصویرها دیده می‌شود فرمول `COUNT(fx)` تعداد سلول‌های ستون و فرمول `SUM(fxc)` مجموع مقادیر ستون را تعیین می‌کنند. هر دو فرمول جزو کتابخانه داخلی اکسل هستند که شامل تعداد زیادی توابع در زمینه‌های مختلف مورد نیاز ازقبیل توابع ریاضی، آماری، تجاری، و ... می‌باشد.

۱۵-۲ فرمول دهی

میله فرمول Formula Bar

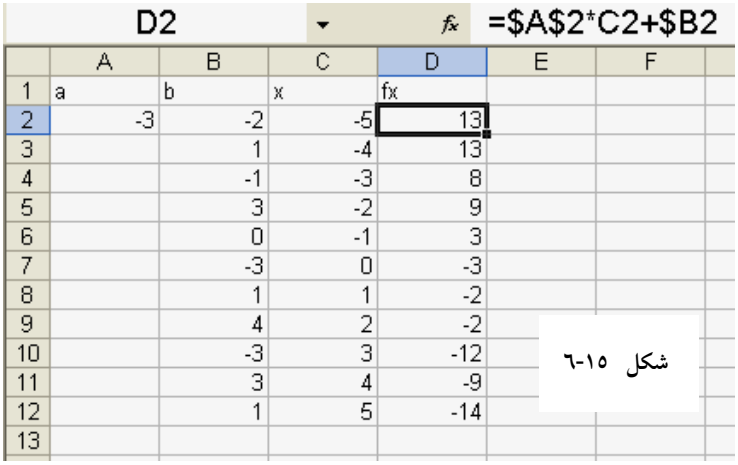
هر فرمول که در یک سلول نوشته می‌شود، شامل نام سلول‌های دیگری از صفحه گسترده یا نام‌های تعریف شده (که شرح آن بعداً می‌آید) است. فرمول نوشته شده در صورت انتخاب سلول مربوطه در میله فرمول نشان داده می‌شود.

برچسبِ ستون- ردیفِ سلول در جعبه نام نمایش داده می‌شود. هرگاه بخواهیم در سلولی فرمول قرار دهیم آن سلول را *انتخاب* کرده ودر میله فرمول پس از *نوشتن علامت مساوی* فرمول را وارد میکنیم.

مثال:

در شکل ۱۵-۶ با توجه به برچسب‌های تعیین شده فرمول `fx=ax+b` را در سلول D2 قرار می‌دهیم.

ابتدا برای هر ستون در بالاترین سطرش یک عنوان مانند آن‌چه در فرمول فوق آمده قرار می‌دهیم. سپس به D2 می‌رویم و فرمول فوق را برحسب برچسب ستون- ردیف به این شکل: `=$A$2*C2+$B2` در میله فرمول می‌نویسیم. هرگاه به D2 برویم این فرمول در میله فرمول ظاهر می‌شود. در مبحث ***نام‌گذاری*** خواهیم دید که چگونه این فرمول به شکل `fx=ax+b` نوشته می‌شود.



D2 =$A$2*C2+$B2

	A	B	C	D	E	F
1	a	b	x	fx		
2	-3	-2	-5	13		
3		1	-4	13		
4		-1	-3	8		
5		3	-2	9		
6		0	-1	3		
7		-3	0	-3		
8		1	1	-2		
9		4	2	-2		
10		-3	3	-12		
11		3	4	-9		
12		1	5	-14		
13						

شکل ۱۵-۶

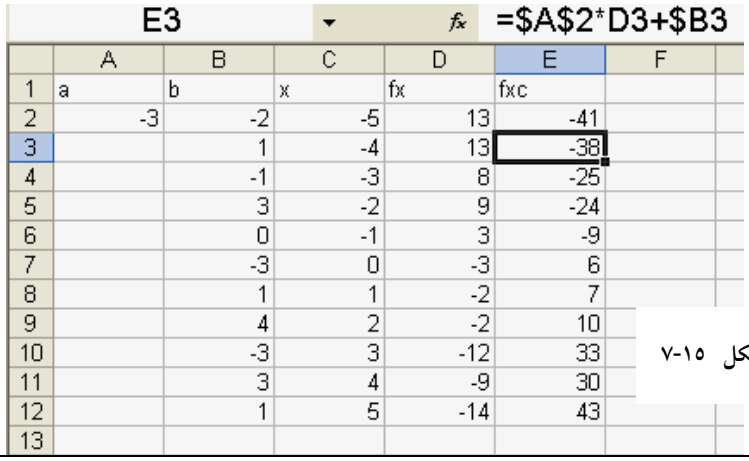
رجوع نسبی و رجوع مطلق به سلول

در رجوع نسبی (بدون علامت $) نسبت یا فاصله سلول‌های رجوع کننده و رجوع شونده حفظ می‌شود، لذا فرمول‌ها خود را با وضعیت جدید تطبیق می‌دهند. در مراجعه مطلق (با دو علامت $ در کنار برچسب‌های ستون و ردیف) مقدار مورد مراجعه دست نخورده می‌ماند. در ارجاع نیمه نسبی (با یک علامت $ در کنار نام ستون یا نام ردیف) مقدار آن‌که علامت $ دارد دست نخورده می‌ماند، اما دیگری به تناسب جابه‌جائی در ردیف یا ستون حالت نسبی می‌یابد.

مثال:

در شکل ۱۵-۷ یک کپی از ستون fx را در ستون fxc قرار داده، فرمول پدیدار شده در سلول E3 را بررسی می‌کنیم. با دقت در میله فرمول دیده می‌شود که رجوع به سلول‌های ردیف C نسبی است یعنی با کپی کردن D3→E3 ارجاع هم یک خانه به راست آمده C3→D3 . مراجعه به سلول A2 مطلق است زیرا در خانه E3 هم $A$2 دست نخورده.

مراجعه به ستون B نیمه نسبی است زیرا ستون آن ثابت مانده اما شماره ردیف به تناسب جابه‌جائی در ردیف (رو به پائین) تغییر کرده $B2→$B3. به این ترتیب در ستون D که فرمول جبری fx=ax+b نوشته شده، مقدار a ثابت، مقدار b ثابت- در- ستون (ثابت نسبی)، و مقدار x متغیر است.



E3 =$A$2*D3+$B3

	A	B	C	D	E	F
1	a	b	x	fx	fxc	
2	-3	-2	-5	13	-41	
3		1	-4	13	-38	
4		-1	-3	8	-25	
5		3	-2	9	-24	
6		0	-1	3	-9	
7		-3	0	-3	6	
8		1	1	-2	7	
9		4	2	-2	10	
10		-3	3	-12	33	
11		3	4	-9	30	
12		1	5	-14	43	
13						

شکل ۱۵-۷

- **برای مراجعه نسبی به یک ستون فقط برچسب آن را می‌آوریم (مثل C2).**
- **برای رجوع مطلق، قبل از برچسب ستون و قبل از شماره ردیف علامت $ قرار می‌دهیم (مثل $A$2).**
- **در ارجاع نیمه نسبی یا قبل از ردیف یا قبل از ستون علامت $ قرار می‌گیرد (مثل $B2).**

نام‌گذاری

برای استفاده از نام‌های ساده‌تر در فرمول‌ها، ابتدا نام و مقدار را در سلول‌های مجاور هم، عمودی یا افقی، وارد کرده سپس نام و مقدار را با هم انتخاب می‌کنیم. از منیوی اصلی `Insert_Name_Create` را می‌آوریم و نگاه می‌کنیم نام در چه سمتِ مقدار واقع شده، و برحسب نسبت مکانی نام عمل می‌کنیم. هم‌چنین می‌توان به مجموعه‌ای از سلول‌ها یک *نام* اختصاص داد، این نام مانند آرایه‌ای عمل می‌کند که مقادیر آن در سلول‌های مجاور نام چیده شده‌اند.

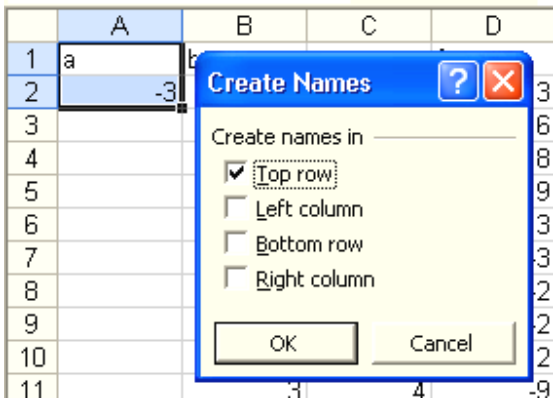
در نام‌گذاری دل‌خواه یک سلول یا یک ستون باید از اختصاص نام‌هائی نظیر `a2, R1, C23` که برچسب پیش‌فرض هستند خودداری کرد.

مثال ها:

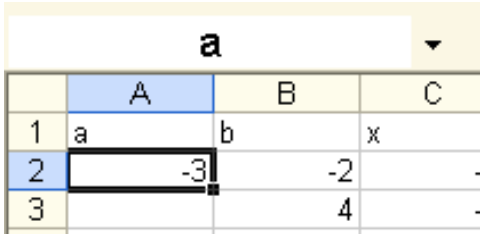
دادن نام به یک سلول

در شکل ۱۵-۸ سلول A2 را با حرف a نام‌گذاری کرده و مقدار 3- را به a نسبت می‌دهیم.

ابتدا نام و مقدار را در سلول‌های مجاور هم، عمودی یا افقی، وارد کرده سپس آن‌ها را انتخاب می‌کنیم. از منیوی اصلی `Insert_Name_Create` را می‌آوریم و نگاه می‌کنیم نام در چه سمتِ مقدار واقع شده. در مثال زیر چون نام بالای مقدار است `Top row` را تیک زده OK می‌کنیم. به این ترتیب متغیری عددی با نام a و با مقدار 3- به مجموعه نام‌های صفحه گسترده افزوده می‌شود. حالا اگر سلول A2 را انتخاب کنیم، در میله برچسب نام آن a نمایش داده می‌شود (شکل ۱۵-۹). از این به بعد به این سلول هم می‌توان با نام a و هم با برچسب A2 مراجعه کرد. نام a را بعداً می‌توان تغییر داد.



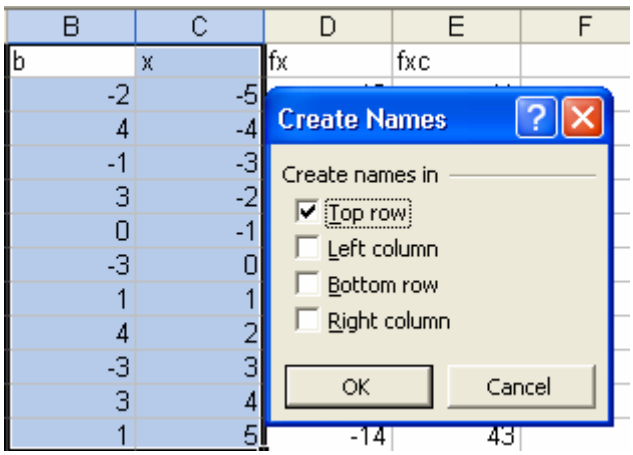
شکل ۱۵-۸



شکل ۱۵-۹

دادن نام به آرایه

در شکل ۱۵-۱۰ b و x را به صورت دو متغیر آرایه‌ای تعریف کرده و مقادیر زیر هر ستون را به آرایه مربوطه نسبت می‌دهیم. ابتدا نام‌ها و مقادیر را انتخاب کرده، به ترتیب ملحوظ در شکل عمل می‌کنیم، نام آرایه‌ها b و x می‌شوند.



شکل ۱۵-۱۰

وارد کردن فرمول با استفاده از نام‌های قراردادی

به D2 رفته فرمول a*x+b را وارد میکنیم، سپس با گیره انتقال فرمول را به خانه‌های زیرین می‌کشیم (شکل ۱۵-۱۱) چون علامت $ وجود ندارد انتقال نسبی خواهد بود.

D2 =a*x+b

	A	B	C	D	E
1	a	b	x	fx	
2	-3	-2	-5	13	
3		4	-4	16	
4		-1	-3	8	
5		3	-2	9	
6		0	-1	3	
7		-3	0	-3	
8		1	1	-2	
9		4	2	-2	
10		-3	3	-12	
11		3	4	-9	
12		1	5	-14	
13					

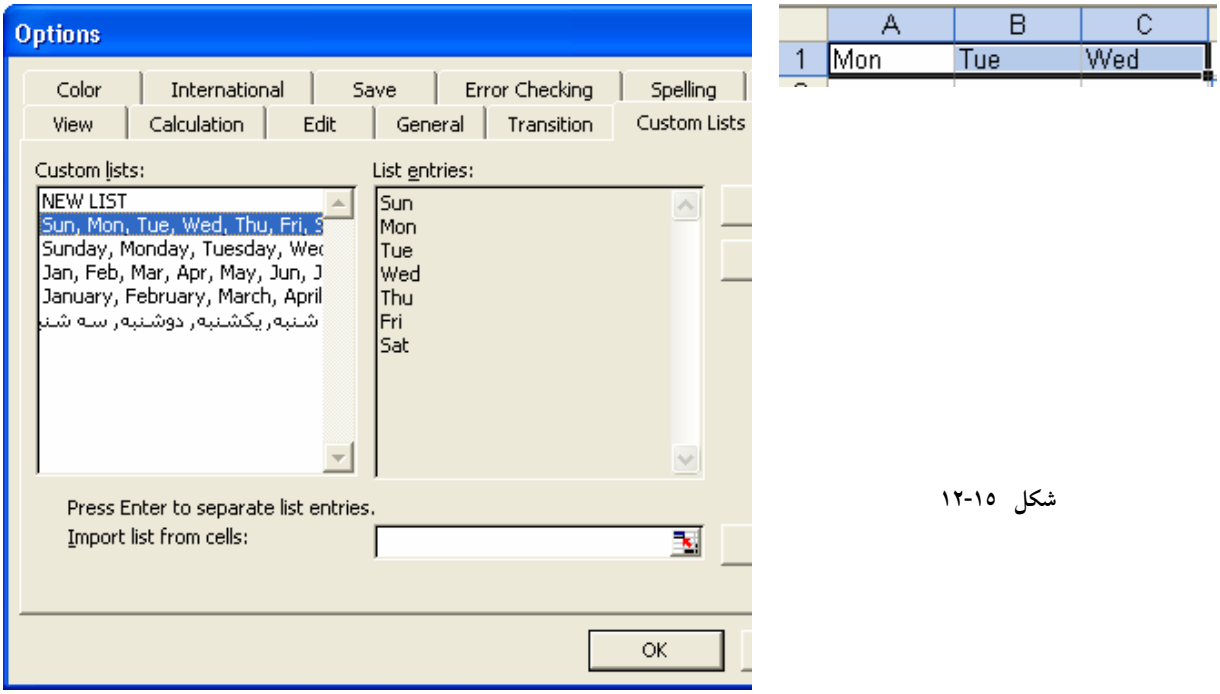
شکل ۱۵-۱۱

۱۵-۳ پر کردن سلول ها با لیست های قراردادی

مثال:

روزهای هفته را به صورت سری در خانه‌های افقی یک صفحه قرار می‌دهیم. در شکل ۱۵-۱۲ ابتدا کلمه Mon را در خانه A1 وارد کرده سپس آن را با گیره انتقال fill handle به راست می‌کشیم.

خانه‌های طرف راست به صورت سری با روزهای هفته پر می‌شود. چنان که گفته شد مربع کوچک پائین- راست یک سلول انتخاب شده گیره انتقال fill handel نام دارد. برای سری پُر شدن بایستی ابتدا با اجرای Tools_Options_Custom Lists مطابق شکل ۱۵-۱۳ لیست سری مورد نظر را وارد کنیم. بعضی از لیست‌ها به صورت پیش‌فرض از قبل وجود دارند.



	A	B	C
1	Mon	Tue	Wed

شکل ۱۵-۱۲

۱۵-۴ تمرین

۱- یک برچسب ستون و یک برچسب ردیف در محل D8 وارد کنید. در سلول آن یک عدد وارد کنید. سپس به سلول A10 رفته با ارجاع به سلول محل تقاطع مربع مقدار آن را در A10 قرار دهید. وضعیت میله فرمول و جعبه نام را بررسی کنید.

۲- در زیرِ برچسب ستون فوق پنج عدد وارد کنید، معدل آن‌ها را با استفاده از تابع کتاب‌خانه‌ای در سلول زیرین قرار دهید.

۳- یک ستون را برچسب قراردادی دهید، و تعدادی عدد زیر آن وارد کنید. حاصل‌جمع و تعداد عناصر (سلول-های) متعلق به ستون را با استفاده از برچسب گذاری و کتب‌خانه داخلی اکسل در سلول بعد از آخرین مقدار قرار دهید. **راهنما:** برای *قابل ارجاع* شدن برچسب‌های دل‌خواه باید انتخاب لازم در منیو تیک زده شود.

٤- یک سلول را نام‌گذاری کرده و مقدار دهید. سلول مربوطه را انتخاب کنید، در میله برچسب چه می‌بینید. در سلول دیگری به این نام با یک عمل ریاضی مراجعه کنید. دو نام در دو سرستون، و مقادیری زیر هر ستون بنویسید. نام‌ها و مقادیر را انتخاب کرده، آرایه‌ها را نام‌گذاری کنید. سپس به سلول سمت راست رفته فرمولی متشکل از دو نام را وارد کنید، سپس با گیره انتقال فرمول را به خانه‌های زیرین بکشید.

۵- نام فصل اول سال را در خانه A1 وارد کرده سپس آن را با گیره انتقال fill handle به پائین بکشید. باید خانه‌ها به صورت سری با نام فصل‌ها پر شودند. **راهنما:** برای سری پُر شدن باید انتخاب لازم در منیو تیک زده شود.

فصل ۱۶ عملیات محاسباتی، توابع و نمودارها

۱۶-۱ چند تابع کتاب خانه ای

مثال:

در شکل ۱۶-۱ بیشینه- کمینه دمای روزهای هفته را وارد کرده، در خانه‌های B4, B5, B6 به ترتیب توابع داخلی AVERAGE(B2:H3), Max(B2:H3), Min(B2:H3) را قرار داده، نتیجه را بررسی میکنیم.

B4 =AVERAGE(B2:H3)

	A	B	C	D	E	F	G	H
1		Mon	Tue	Wed	Thu	Fri	Sat	Sun
2	Min	13.5	18	14	14.5	12.6	14	12.5
3	Max	17	29	18	22	19	23.5	18.6
4	Av/Week	17.59						
5	Max/Wk	29						
6	Min/Wk	12.6						
7								

شکل ۱۶-۱

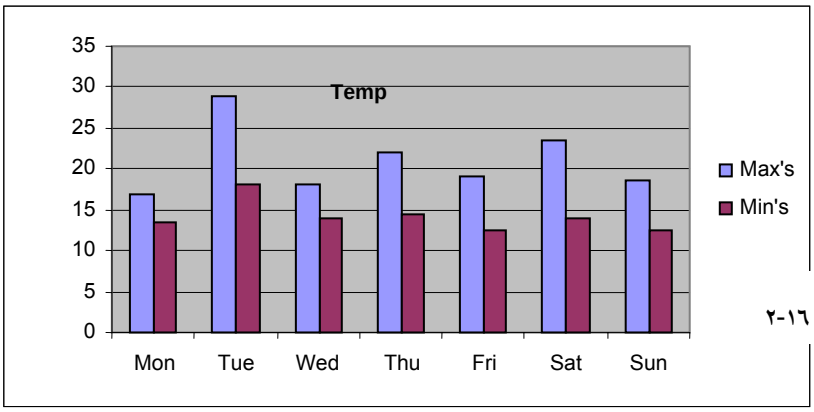
۱۶-۲ نمودارها

نمودار ستونی

مثال:

با استفاده از جدول فوق درجه حرارت بیشینه و کمینه هر روز را برحسب روز هفته با نمودار ستونی نمایش می‌دهیم.

دکمه چارت را از میله ابزار کلیک کرده، پس از انتخاب نمودار ستونی و زدن Next به لبه Series رفته برای سری یک B2:H2 و برای سری دو B3:H3 و برای Category (X) axis labels: ، B1:H1 را وارد می-کنیم. پس از انتخاب اسامی و برچسب‌های مناسب نمودار را به شکل ۱۶-۲ رسم می‌کنیم:



شکل ۱۶-۲

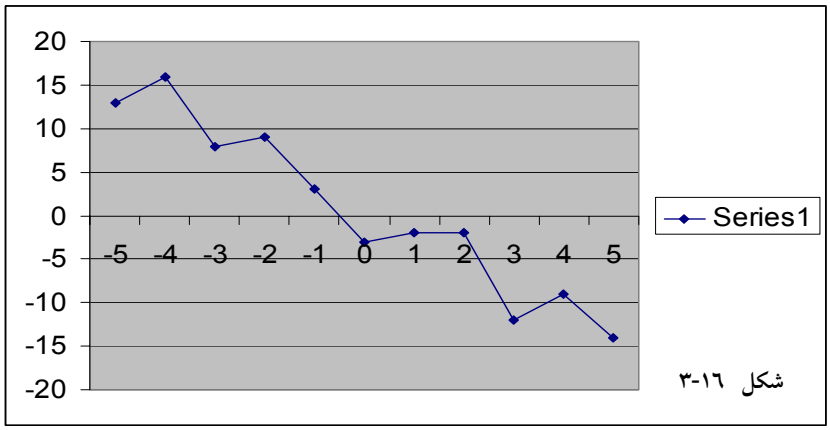
ترسیم منحنی

مثال:

برای رسم نمودار fx برحسب x (جدول شکل ۱۶-۳) دکمه چارت را از میله ابزار کلیک کرده، پس از انتخاب نمودار منحنی و زدن Next به لبه Series رفته برای سریِ یک، ستون fx

و برای Category (X) axis labels: ، ستون x را وارد می‌کنیم.

C	D
x	fx
-5	13
-4	16
-3	8
-2	9
-1	3
0	-3
1	-2
2	-2
3	-12
4	-9
5	-14

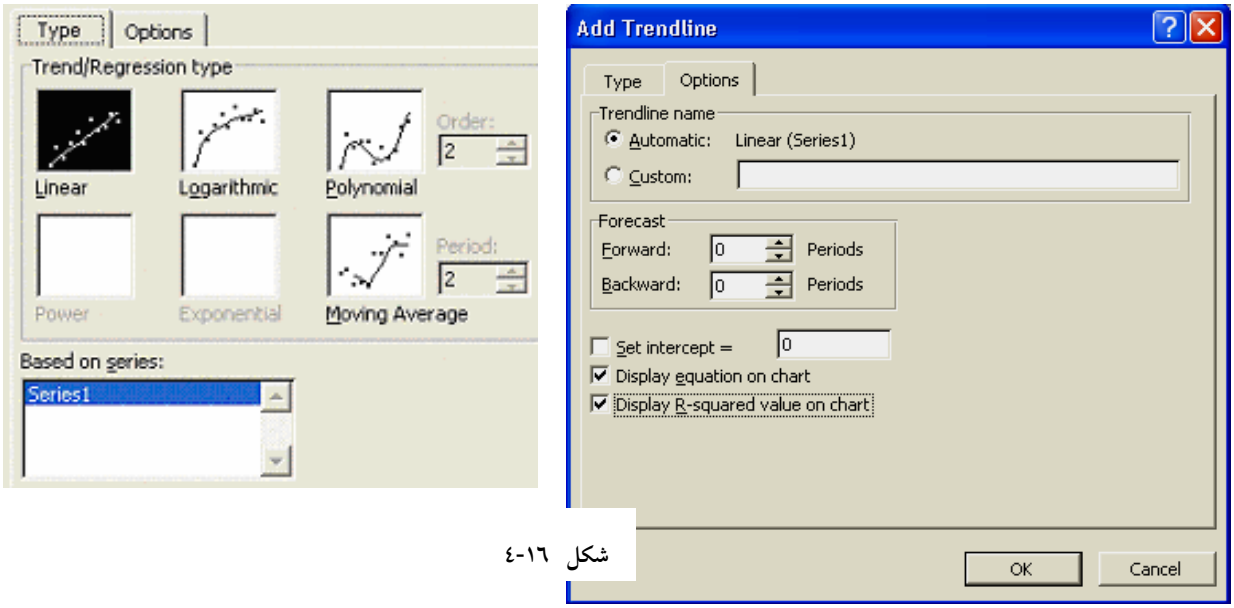


شکل ۱۶-۳

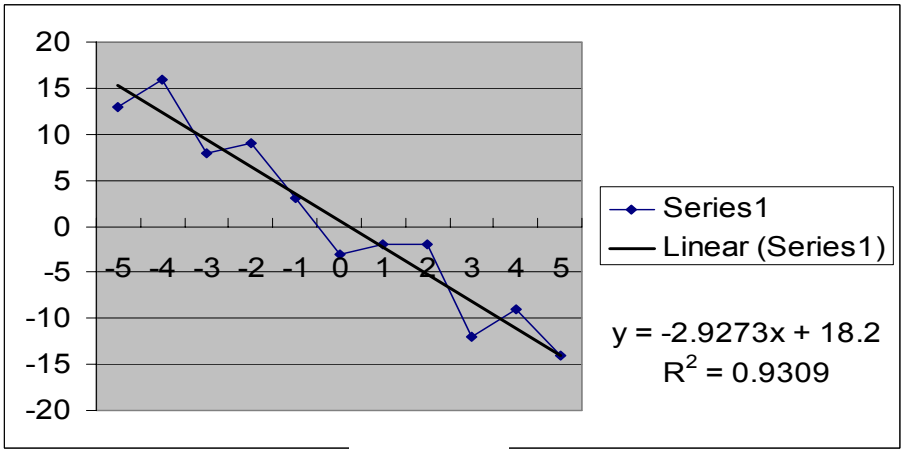
جبری سازی منحنی

مثال:

برای حصول نزدیک‌ترین منحنی جبری روی یکی از نقاط منحنی فوق راست کلیک کرده و Add Trendline را کلیک می‌کنیم. در جعبه محاوره شکل ۱۶-٤ در لبه Type یکی از منحنی‌های متناسب با منحنی خود را انتخاب میکنیم. سپس درلبه Options انتخاب‌های شکل را تیک زده OK می‌کنیم. منحنی ۱۶-۵ نتیجه می‌شود. پارامترR-squared یا ضریب تعیّن هر قدر به یک نزدیک‌تر باشد، تقریب منحنی بهتر است.



شکل ۱۶-٤



شکل ۱۶-۵

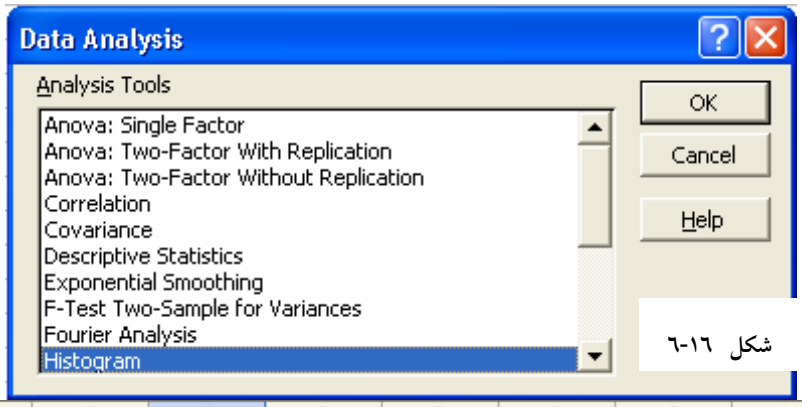
۳-۱۶ هیستوگرام

تعدادی عدد تصادفی نامرتب در نظر بگیرید، هیستوگرام عددهای مساوی را داخل یک ظرف می‌ریزد و موجودی هر ظرف را نمایش میدهد.

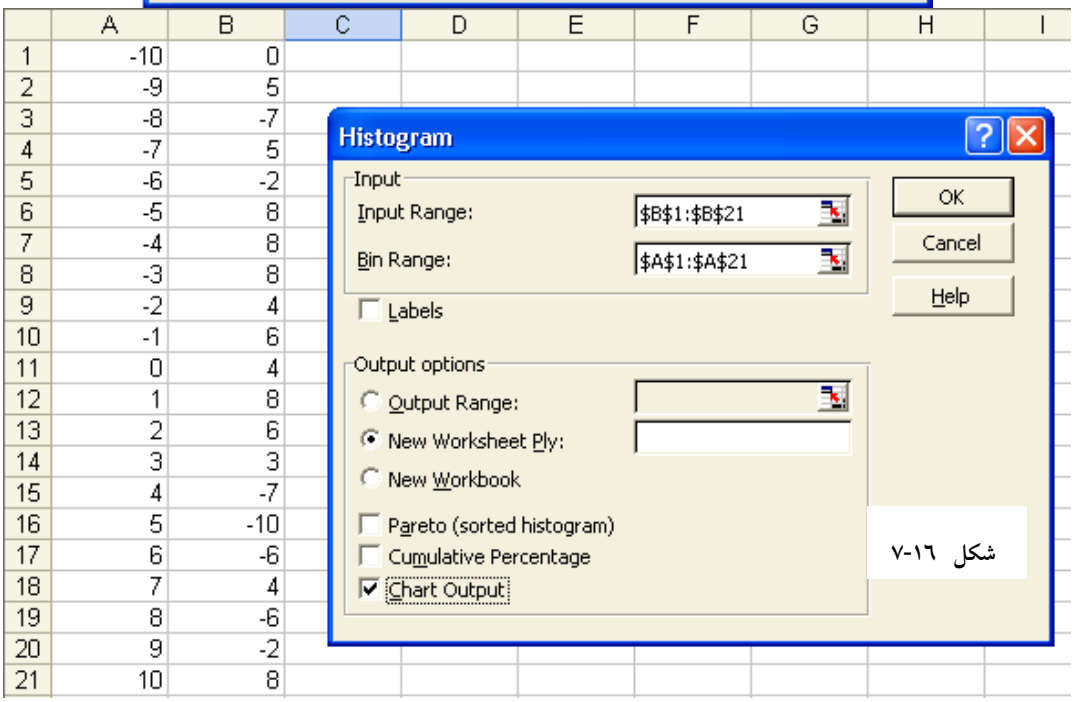
اگر انتخاب Data Analysis در زیرمنیوی Tools موجود نباشد، از منیوی اصلی انتخاب Tools_Add-Ins... را اجرا و Data Analysis را به زیر منیوی Tools اضافه می‌کنیم.

مثال:

این تعداد عدد صحیح تصادفی 0,5,-7,5,-2,8,8,8,4,6,4,8,6,3,-7,-10,-6,4,-6,-2,8 را که در بازه 10- تا 10 قرار دارند در نظر بگیرید، برای رسم هیستوگرام آن‌ها ابتدا مطابق شکل، بازه اعداد (10- تا 10) را به صورت مرتب در ستون سمت چپ و اعداد تصادفی فوق را درستون مقابل آن وارد می‌کنیم. از منیوی اصلی، Tools_ Data Analysis را کلیک می‌کنیم. سپس از جعبه محاوره ظاهر شده Histogram را انتخاب می‌کنیم (شکل ۱۶-۶). ستون B را به عنوان Input Range و ستون A را به عنوان Bin Range قرار داده New Worksheet Ply و Chart Output را تیک زده، OK می‌کنیم(شکل ۱۶-۷). نتیجه در یک کاربرگ دیگر ظاهر می‌شود (شکل ۱۶-۸).



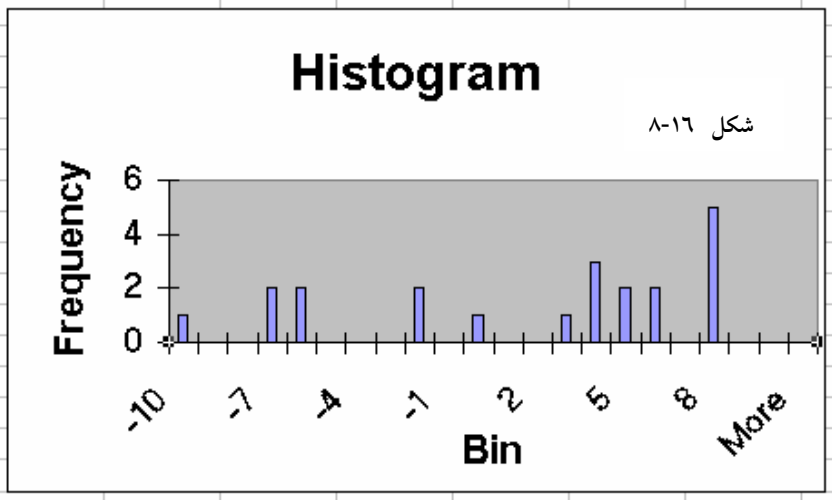
شکل ۱۶-۶



	A	B
1	-10	0
2	-9	5
3	-8	-7
4	-7	5
5	-6	-2
6	-5	8
7	-4	8
8	-3	8
9	-2	4
10	-1	6
11	0	4
12	1	8
13	2	6
14	3	3
15	4	-7
16	5	-10
17	6	-6
18	7	4
19	8	-6
20	9	-2
21	10	8

شکل ۱۶-۷

	A	B
1	*Bin*	*Frequency*
2	-10	1
3	-9	0
4	-8	0
5	-7	2
6	-6	2
7	-5	0
8	-4	0
9	-3	0
10	-2	2
11	-1	0
12	0	1
13	1	0
14	2	0
15	3	1
16	4	3
17	5	2
18	6	2
19	7	0
20	8	5
21	9	0
22	10	0
23	More	0
24		



شکل ۸-۱۶

۱۶-۴ تابع کاربر- تعریف در اکسل VBA in Excel

برنامه نویسی

با استفاده از VBA می‌توان فرمول‌هائی را تعریف کرد و آن‌ها را نظیر توابع کتاب خانه ای اکسل داخل سلول‌ها نوشت و به کار برد. حروف VBA تقطیع عبارت Visual Basic for Applications است که یک زبان برنامه نویسی است که در داخل نرم‌افزارهای Microsoft Office نظیر Word, Excel تعبیه شده است.

مثال:

سرعت یک چتر باز را از لحظه پرش تا زمانی که به سرعت یک‌نواخت برسد، در لحظات مختلف با آنالیز عددی به دست آورده و منحنی آن را رسم کنید. وزن چترباز را با m ، و ضریب مقاومت هوا را با cv نشان دهید.

حل: نیروی وارد به چترباز در لحظه t1 برابر نیروی ثقل زمین منهای نیروئی است که در اثر مقاومت هوا درجهت خلاف نیروی ثقل وارد می‌شود. نیروی مقاومت هوا با سرعت متناسب، و برابر است با:

$$F_{air}(t1) = cv * v(t1)$$

در نتیجه شتاب معکوس حاصل از مقاومت هوا برابر است با:

$$acc_{air} = (cv/m)*v(t1)$$

شتاب کل در لحظه t1 مساوی شتاب گرانش خواهد بود منهای شتاب معکوس:

$$acc(t1) = g - (cv/m)*v(t1)$$

شتاب تقریبی در لحظه t1 می‌شود (زیرنویس appr همان approximated یعنی تقریبی است):

$$acc_{appr} = [v(t1+DELt) - v(t1)]/DELt$$

عدد ثابت DELt رشد زمان را نشان می‌دهد.

حال مقدار واقعی شتاب را با مقدار تقریبی آن برابر قرار می‌دهیم:

$$acc_{appr} \approx acc(t1) \rightarrow [v(t1+DELt) - v(t1)]/DELt \approx g - (cv/m)*v(t1)$$
$$v(t1+DELt) = v(t1) + [g-(cv/m)v(t1)] * DELt$$

سرعت پس از رشد زمان `v(t1+DELt) = v(t1) + acc * DELt`

عبارت فوق را اگر بخواهیم به صورت کامپیوتری بنویسیم می‌شود:

```
v = v + acc * DELt
```

حال از منیوی اصلی `Tools_Macro_Visual Basic Editor` را اجرا و تابع زیر را در ادیتور VBA نوشته و ضبط می‌کنیم. زبان VBA قواعد خاصی دارد و تا حدی شبیه `Quick Basic` است.

```
Function Vnum(DELt, t1, t2, v1, m, cv)
Dim t As Single, dvBdt As Single, acc As Single, v As Single
Const g As Single = 9.8
t = t1
v = v1
Do
 acc = g - (cv / m) * v
 v = v + acc * DELt
 t = t + DELt
 If t >= t2 Then Exit Do
Loop
Vnum = v
End Function
```

در این تابع زمان را بین `t1` و `t2` به قطعات `DELt` ثانیه‌ای تقسیم کرده‌ایم. سپس در داخل حلقه `DO ... LOOP` زمان بعد از `DELt` را مساوی زمان قبل از آن به علاوه رشد زمان قرار داده‌ایم یعنی: `(t = t + DELt)` هم‌چنین سرعت بعد از `DELt` را مساوی سرعت قبل از آن به علاوه رشد سرعت قرار داده‌ایم یعنی: `(v = v + acc * DELt)`. در هر تکرار، سرعت در داخل حلقه حساب می‌شود تا به زمان `t2` برسیم و حلقه تمام شود . سرعت در `t2` به عنوان خروجی تابع برگشت داده می‌شود. سرعت و زمان اولیه، ضریب مقاومت هوا، و شتاب ثقل آرگومان‌های دیگر تابع `Vnum` هستند. این تابع جمعاً شش آرگومان دارد.

استفاده از تابع VBA در صفحه گسترده

همان‌طور که ذکر کردیم، می‌توان توابعی را با VBA نوشته‌ایم، نظیر توابع کتاب خانه ای اکسل به صورت فرمول در سلول‌ها نوشت و به کار برد. مثال زیر کاربرد تابع `Vnum` را نشان می‌دهد.

مثال:

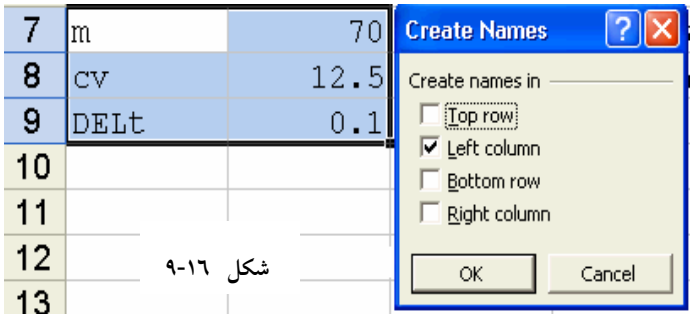
ابتدا ستونی را `t` و ستون دیگری را `Vnum` عنوان می‌دهیم. سپس ستون `t` را از زمان صفر شروع کرده با افزایش ۳ ثانیه-ای حدود ۲۰ سلول را پر کرده، در ستون `Vnum` فرمول منتج از تابع فوق را با آرگومان‌های

`m = 70kg, DELt = 0.1sec, cv = 12.5kg/sec`

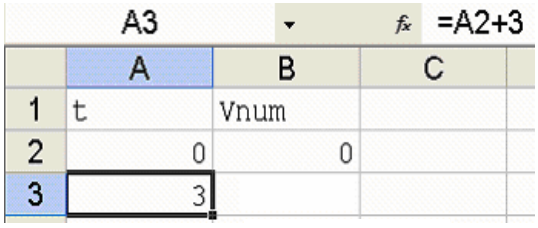
قرار می‌دهیم. منحنی سرعت بر حسب زمان را رسم می‌کنیم.

ابتدا آرگومان‌ها و مقادیر آن‌ها را به صورت نام‌گذاری وارد می‌کنیم (شکل ۱۶-۹)

سپس عناوین `t` و `Vnum` را در سلول‌های مربوطه قرار می‌دهیم. زیر `t` اول صفر را وارد کرده و در ردیف بعد ۳ واحد به آن اضافه می‌کنیم. سپس با گیره انتقال آن را به تعداد خانه‌های مورد نظر پائین می‌کشیم (شکل ۱۶-۱۰).



شکل ۱۶-۹



شکل ۱۶-۱۰

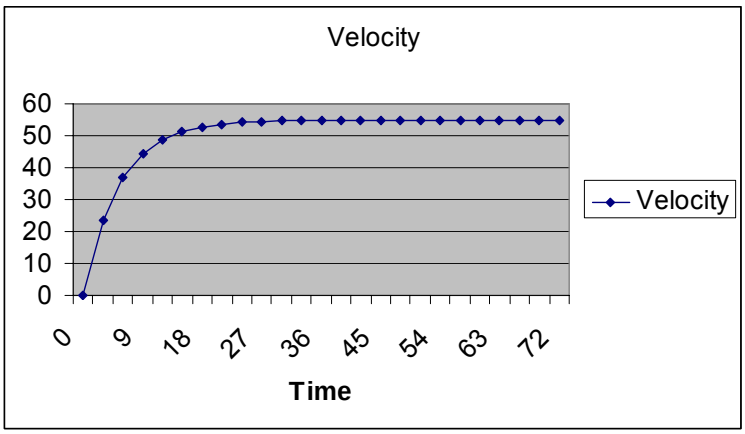
در ستون Vnum ابتدا صفر و در ردیف بعد تابع VBA را چنان‌چه در شکل ۱۶-۱۱ دیده می‌شود می‌نویسیم. در این فرمول آرگومان t1 را برابر A2، آرگومان t2 را برابر A3 و آرگومان v1 را برابر B2 قرار داده‌ایم. فرمول نوشته شده در سلول B3 را با گیره انتقال آن به تعداد خانه‌های مورد نظر پائین می‌کشیم. با توجه به ارجاع نسبی مقادیر A2, A3, B2 در سلول‌های زیرین به تناسب تغییر می‌کند.

مقدار سرعت از ردیف 19 به بعد تقریباً یک‌نواخت شده، این موضوع در نمودار رسم شده شکل ۱۶-۱۱ هم دیده می‌شود.

B3 =Vnum(DELt,A2,A3,B2,m,cv)

	A	B	C	D	E
1	t	Vnum			
2	0	0			
3	3	23.487			

t	Vnum
0	0.0000
3	23.4873
6	36.9226
9	44.4212
12	48.7885
15	51.3322
18	52.8137
21	53.6765
24	54.1791
27	54.4717
30	54.6422
33	54.7440
36	54.8022
39	54.8355
42	54.8545
45	54.8654
48	54.8717
51	54.8752
54	54.8773
57	54.8784
60	54.8791
63	54.8795
66	54.8797
69	54.8798
72	54.8799



شکل ۱۶-۱۱

۱۶-۵ ضبط ماکرو

از منیوی اصلی Tools_Macro_Record New Macro را اجرا می‌کنیم. میله ابزار ضبط ماکرو ظاهر شده و از آن لحظه اعمالی که روی صفحه گسترده انجام دهیم ضبط می‌شوند. با زدن دکمه Stop Recording در میله ابزار ضبط ماکرو متوقف می‌شود. اعمال ضبط شده به صورت یک فایل VBA در می‌آید. که از داخل اکسل قابل اجرا یا قابل ادیت است.

در اکسل یک برنامه که با VBA یا از طریق ضبط ماکرو تهیه شود، ماکرو نام دارد. برای فعال شدن ماکروها بایستی Tools_Options_Security_Macro Security_Medium را از منیوی اصلی اجرا کرد. به این ترتیب به هنگام شروع اکسل ماکروها به اختیار خودتان فعال یا غیر فعال می‌شوند.

سؤال: نام خود را در یک سلول وارد کنید. کپی کردن و چسباندن آن به سلول دیگر را به صورت ماکرو ضبط کنید.

۱۶-۶ ابزارهای محاسباتی

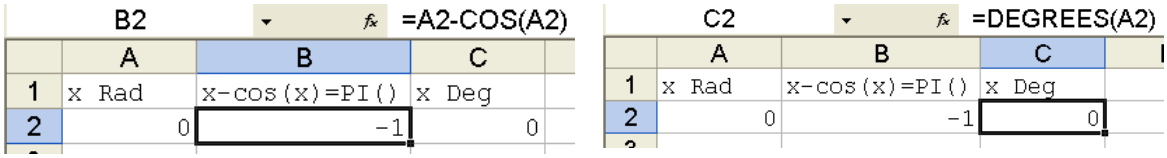
Goal Seek

ابزار Goal Seek می‌تواند پس از رساندن یک تابع به مقدار نهائی، مقدار متغیر مستقل را برای این مقدار نهائی تعیین کند.

مثال:

مقدار x را برای برقراری تساوی $x - \cos x = \pi$ به دست می‌آوریم.

مقدار x را در سلول A2 و تابع A2-COS(A2) را به شکل فرمول در B2 وارد می‌کنیم. برای این که x بر حسب درجه هم دیده شود، تابع DEGREES(A2) را هم در ستون بعد قرار می‌دهیم. توضیح مربوط به هر یک ازاین سلول‌ها در سلول بالای آن آمده، شکل ۱۶-۱۲ (ستون x Deg نقشی در Goal Seek ندارد).

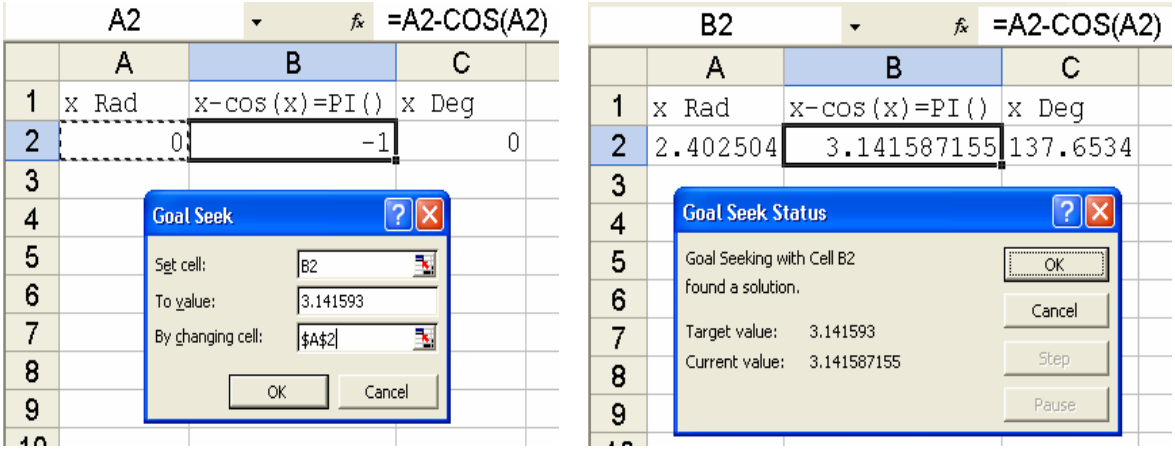


شکل ۱۶-۱۲

اکنون Tools_Goal Seek را اجرا می‌کنیم. در Set cell خانه‌ای را که معادله در آن آمده (B2)، در To value مقدار سمت راست معادله π و در By changing cell خانه‌ی متغیر x (A2) را می‌نویسیم و OK می‌کنیم شکل ۱۶-۱۳. در To value بایستی حتماً یک عدد صحیح یا اعشاری وارد کرد.

Goal Seek مقداری برای x را که در معادله صدق کند محاسبه می‌کند، در صورت پذیرش این مقدار OK می‌کنیم.

نوشته‌های ردیف 1 فقط توضیح است.



شکل ۱۶-۱۳

سؤال: برای مشاهده مقدار π تابعِ مقدار- ثابتِ PI() را در یکی از خانه‌ها وارد کنید.

Solver

این ابزار برای حل معادلات چند مجهولی و موارد مشابه آن به کار می‌رود.

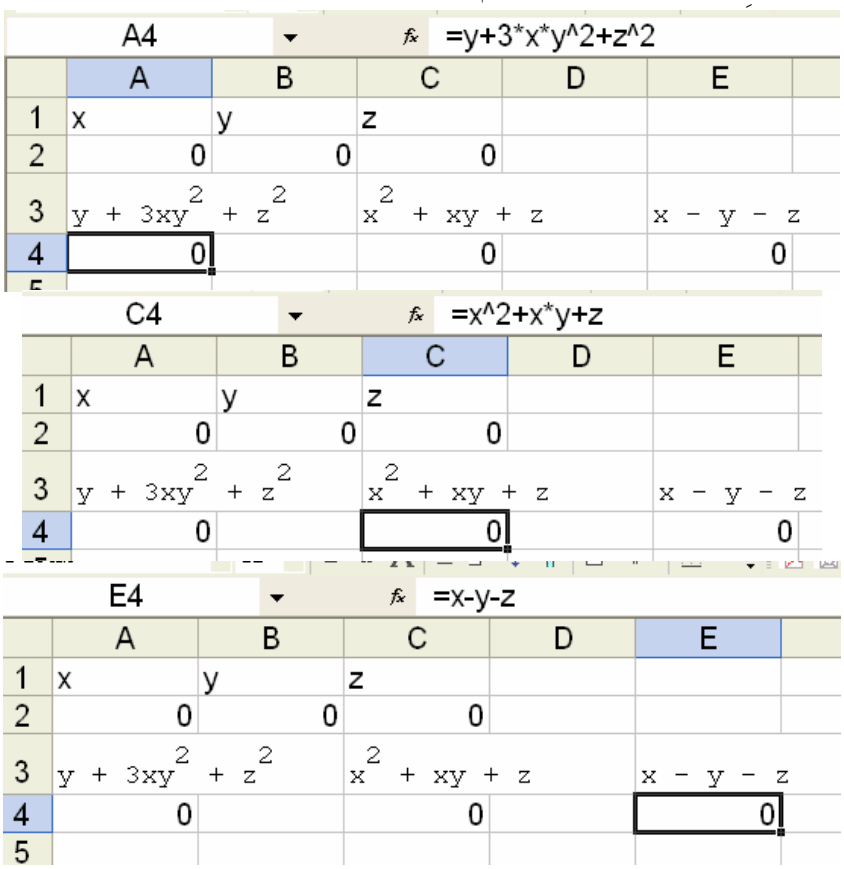
اگر انتخاب Solver در زیرمنوی Tools موجود نباشد، Tools_Add-Ins... را اجرا و Solver را به زیر منیوی Tools اضافه می‌کنیم.

مثال:

دستگاه معادلات سه مجهولی زیر را حل می‌کنیم:

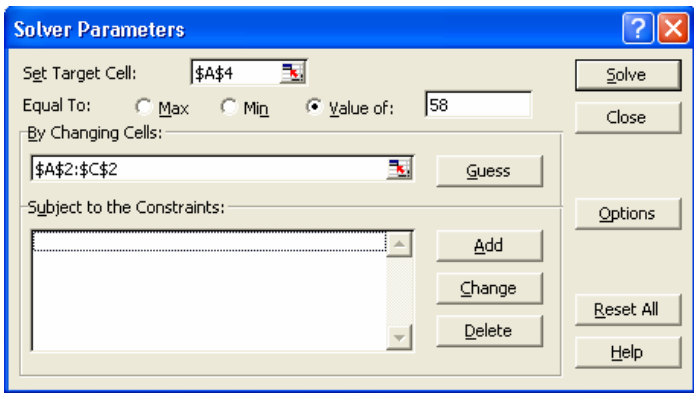
$$y + 3xy^2 + z^2 = 58$$
$$x^2 + xy + z = 9$$
$$x - y - z = 0$$

ابتدا x=0, y=0, z=0 را به عنوان سه نام از طریق Insert_Name_Create تولید می‌کنیم. در خانه‌هائی نزدیک به آن‌ها، طرف چپ معادلات فوق را می‌نویسیم در شکل ۱۶-۱۴ به میله فرمول توجه کنید.

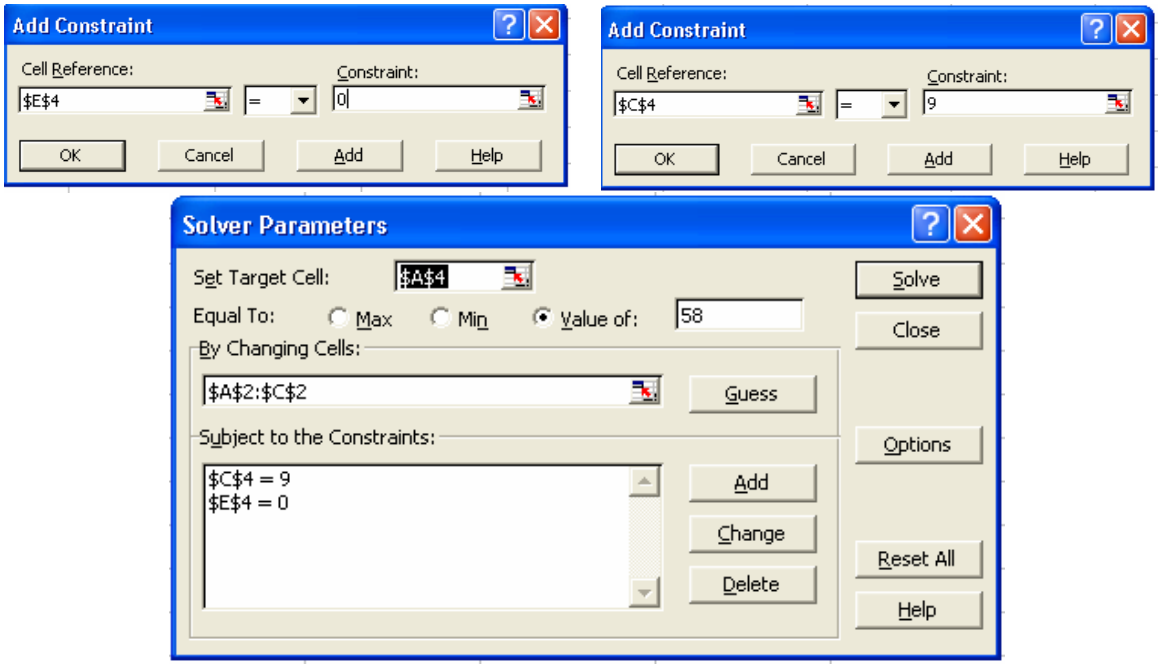


شکل ۱۶-۱۴

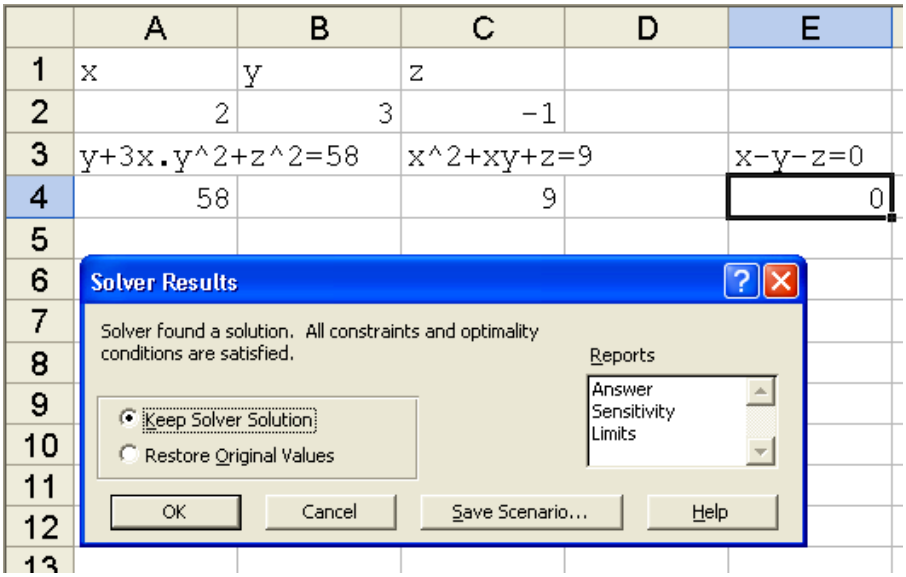
سپس از منیوی اصلی Tools_Solver را اجرا می‌کنیم. در مقابل Set Target Cell برچسب سلول معادله اول $A$4، در مقابل Value of مقدار سمت راست معادله اول 58، و در جعبه By Changing Cells دامنه سلول‌های شامل مقادیر x, y, z را قرار می‌دهیم. در قسمت Subject to the Constraints دکمه Add را اجرا کرده و خانه‌های شامل فرمول‌های دو معادله دیگر را به ترتیب در Cell Reference و مقادیر سمت راست آن‌ها را در Constraint وارد می‌کنیم. پس از OK جعبه محاوره Solver Parameters به شکل نشان داده شده در ۱۶-۱۵ در می‌آید. سپس دکمه Solve را می‌زنیم. در جعبه Solver Results دکمه Keep Solver Solution را OK می‌کنیم (شکل ۱۶-۱۶). مقادیر x, y, z در سلول‌های A2, B2, C2 نوشته می‌شوند(شکل ۱۶-۱۷). اگر در جعبه Solver Results دکمه Save Scenario را بزنیم. اعمال انجام شده به صورت Scenario ضبط می‌شوند. با اجرای Tools_Scenario می‌شود Scenario را مشاهده کرد یا گزارش گرفت.



شکل ۱۶-۱۵



شکل ۱۶-۱۶



شکل ۱۶-۱۷

جدول داده Data Table

مثال:

مقاومت معادل این شکل را با مقادیر R1 = 10, R2 = 15 و 18=> R3 >=0 به دست آورده بر حسب تغییرات R3 رسم می‌کنیم.

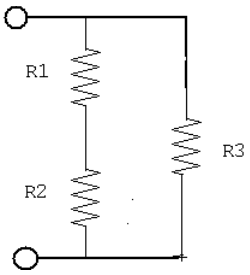
ابتدا سه نام با مقادیر مربوطه تعریف می‌کنیم. چون R1 برچسب ستون- ردیف پیش‌فرض است برای نام‌گذاری از R_1 استفاده کرده‌ایم (شکل ۱۶-۱۸).

فرمول را در یکی از خانه‌ها نوشته یک ستون به چپ، یک ردیف پائین رفته و مقادیر R3 را رو به پائین وارد می‌کنیم (شکل ۱۶-۱۹)

سلول‌هائی را که شامل فرمول و مقادیر R3 هستند مارک کرده، از منیوی اصلی Data_Table را اجرا می‌کنیم. در Column input cell محل متغیر را که در این جا E3 است وارد و OK می‌کنیم (شکل ۱۶-۲۰)

مقادیر مقاومت معادل برحسب مقادیر R3 در مقابل آن‌ها نوشته می‌شود.

سپس منحنی مربوطه را رسم می‌کنیم. (شکل ۱۶-۲۱)



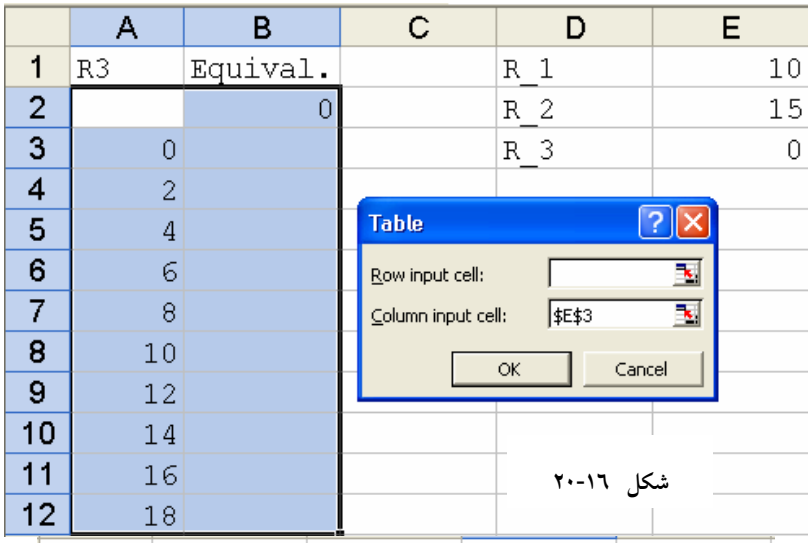
R_1	10
R_2	15
R_3	0

شکل ۱۶-۱۸

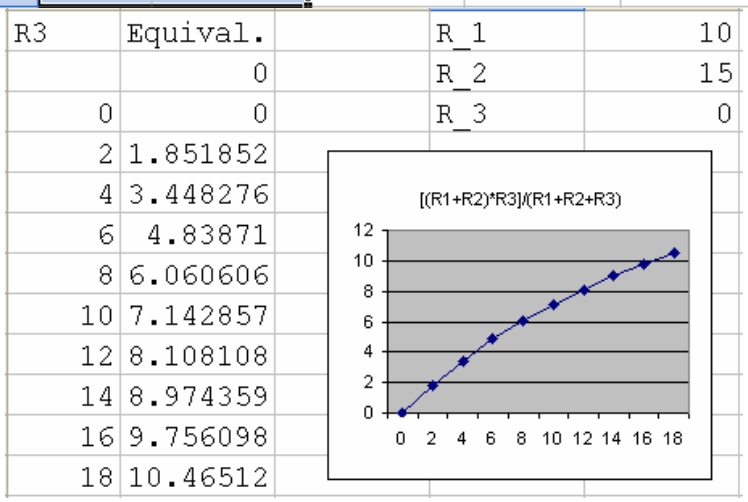
B2 =(R_1+R_2)*R_3/(R_1+R_2+R_3)

	A	B	C	D	E	F
1	R3	Equival.		R_1	10	
2		0		R_2	15	
3	0			R_3	0	
4	2					
5	4					
6	6					
7	8					
8	10					
9	12					
10	14					
11	16					
12	18					

شکل ۱۶-۱۹



شکل ۱۶-۲۰



شکل ۱۶-۲۱

۱۶-۷ تمرین

۱- نمرات بیست دانشجو را به چهار دسته پنج‌تائی تقسیم و با نمودار ستونی نمایش دهید.

۲- میزان باران ماه‌های یک سال را در جدولی قرار داده، پس‌از رسم منحنی دو نوع جبری سازی انجام دهید.

۳- برای پانصد نفر ده مجموعه سنی تعریف کنید، و هیستوگرام آن را نمایش دهید.

۴- فرمول `B = A * (1+r)^n` را که در آن `A = 300000` سرمایه اولیه، B سرمایه نهائی، `r = 0.1` نرخ سود و n تعداد سال‌ها است، به صورت فرمول کاربر تعریف بنویسید. در یک ستون سرمایه اولیه و در ستون دیگر با استفاده از تابعی که نوشته‌اید سرمایه نهائی را بر حسب سال نمایش دهید.

۵- چند عمل مانند محاسبه حاصل‌جمع یک ستون را به صورت ماکرو ضبط و سپس اجرا کنید.

۶- معادله $\frac{2}{x^5}+e^{-x}=0$ را با `Goal Seek` حل کنید.

۷- این دستگاه را با `solver` حل کنید:

$$uw^2 + t^2 = -5,\ u^2 + w^2 + t^2 = 14,\ u - 2w - 2t = -10$$

۸- فرمول قسط ماهانه پرداخت وام خانه: $P = [rL(1+r/12)^{12N}]/[12[(1+r/12)^{12N} -1]]$

مبلغ `L = 10e6` وام با سود `r = 0.15` سالانه در مدت `N = 3` سال بازپرداخت می‌شود، قسط ماهانه را برای زمان بازپرداخت (۳ سال) به صورت جدول داده نشان دهید،و رسم کنید است.