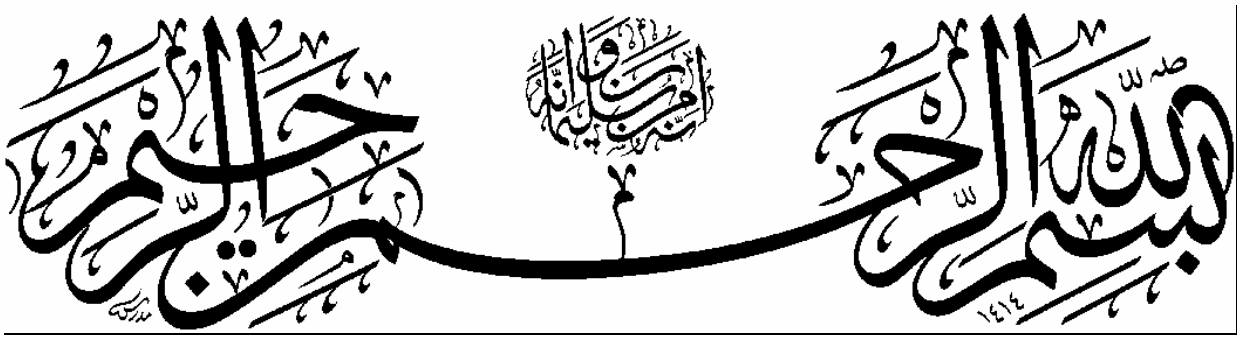


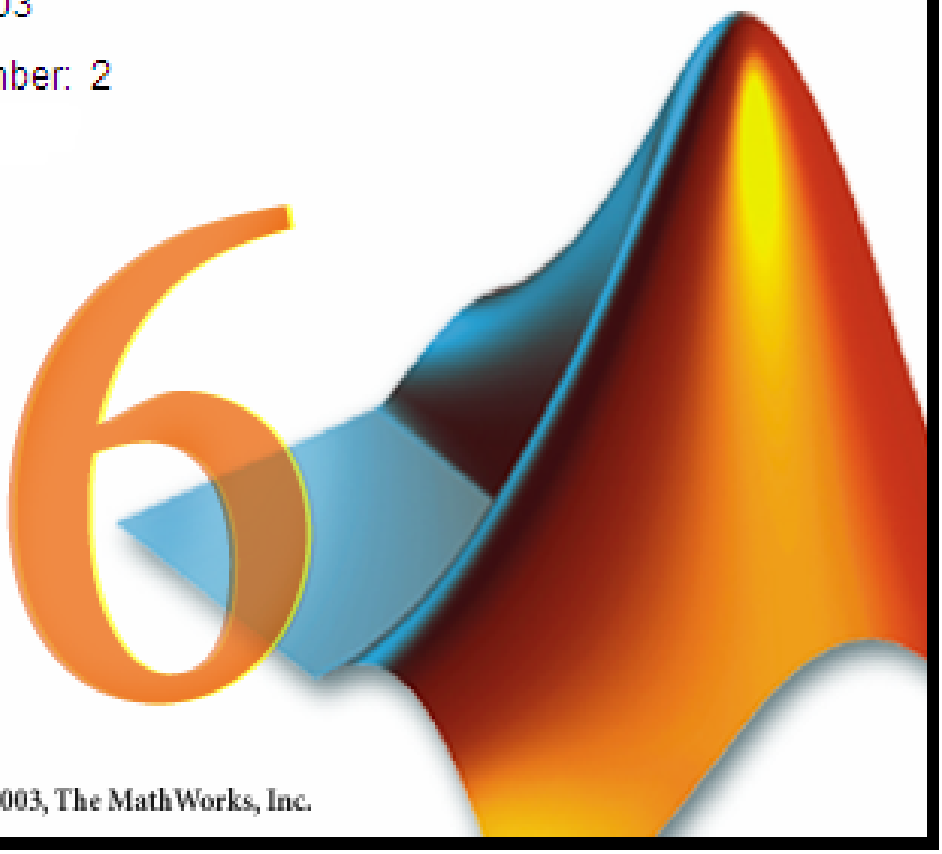
راهنمای استفاده از MATLAB

MATLAB®

The Language of Technical Computing

Version 6.5.1.199709 Release 13 (Service Pack 1)
August 4, 2003

License Number: 2



۱- مقدمه

۱-۱ MATLAB چیست؟

نرم افزار MATLAB برنامه کامپیوتری است که برای کسانی که با محاسبات عددی، و بویـژه جـبر خطی سر و کار دارند، تهیه شده است. نام این نرم افزار از عبارت انگلیسی MATrix LABoratory اقتباس شده و هدف اولیه آن قادر ساختن مهندسین و دانشمندان به حـل مسـائل شـامل عملیـات ماتریسی بدون نیاز به نوشتن برنامه در زبانهای برنامه نویسـی متـداول همچـون C و FORTRAN بود. با گذشت زمان قابلیتهای بسیار بیشتری به این نرم افزار افزوده شده اند بطـوری کـه در حـال حاضر MATLAB به ابزار پر قدرتی برای ترسیم داده ها، برنامه نویسی و انجام محاسبات مهندسـی و پژوهشی تبدیل شده است.

در طول این جزوه علامت « علامتی است که در محیـط کـار موجـود اسـت و نشـان دهنـده محـل نوشتن دستورات می باشد. شما نیازی ندارید که آن را بنویسید.

۱-۲ استفاده از help

در صورتی که بخواهید در مورد دستور و یا تابع خاصی اطلاعاتی بـه دسـت بیـاورید مـی توانیـد در پنجره MATLAB کلمه help و پس از آن نام دستور یا تابع مورد نظر را نوشته و کلید بازگشـت را فشار دهید:

» help magic

```
MAGIC  Magic square.
   MAGIC(N) is an N-by-N matrix constructed from the integers
   1 through N^2 with equal row, column, and diagonal sums.
   Produces valid magic squares for N = 1,3,4,5,...
```

روش دیگر استفاده از help بکار بردن دستور helpwin است. این دستور پنجره کمک MATLAB را باز می کند و اجازه می دهد تا توضیحـات مـورد نیـاز را در پنجـره جـداگانـه ای بدسـت آوریـد. توضیحات داده شده در این پنجره همانهایی هستند که دستور help ارائه می نماید.

لازم به توضیح است که نام دستورات و توابع در help با حروف بزرگ آورده مـی شـوند در حالیکـه MATLAB نسبت به بزرگ و کوچک بودن حروف حساس است و هنگام استفاده از این دستورات و توابع باید آنها را با حروف کوچک بکار برد.

۱-۳ استفاده از demo

دستور demo پنجره جدیدی باز می کند که شما در آن مـی توانیـد مثالـهای متعـددی از امکانـات MATLAB را بیابید. بسیاری از این مثالها نمودارهای جالب و همراه با جزئیات تولید می نمـایند و همچنین توضیحات مفیدی در باره نحوه استفاده از MATLAB ارائه می دهند. توصیه می شود که حتماً تعدادی از این مثالــها را مشـاهده کنیـد تـا متوجـه شـوید کـه چـه کارهـایی مـی تـوان بـا MATLAB انجام داد. بویژه دقت کنید که چگونه برنامه های ساده مـی تواننـد نتـایج پیچیـده ای تولید نمایند.

۲- عملیات ابتدایی

۲-۱ تعریف کردن آرایه ها و عملیات جبری روی آنها

در MATLAB چهار نوع آرایه می توان تعریف کرد:

۱. اعداد اسکالر که تک عضوی هستند.
۲. بردارها که شامل یک سطر یا ستون می باشند (یک بعدی).
۳. ماتریسها که از اعضای چیده شده در یک آرایش مربعی تشکیل می گردند (دو بعدی).
۴. آرایه های با ابعاد بیش از دو.

اعضای یک آرایه می توانند عدد و یا حرف باشند و تفاوتی بین اعداد صحیح و اعشاری وجود نـدارد. در صورت جایگزینی یک عـدد و یـا حـرف در یـک متغـیر، MATLAB مقـدار جـایگزین شـده را بلافاصله نشان می دهد مگر آنکه عبارت تعریف متغیر با semicolon خاتمه یابد.

```
» a=2.5

a =

   2.5000

» a=3.2;
» a


a =

   3.2000

» p='hello'

p =

hello
```

MATLAB بین حروف کوچک و بزرگ فرق قائل است:

```
» A
??? Undefined function or variable 'A'.
```

از آنجا که نشان دادن مقادیر به شکل فوق قدری طولانی است معمولاً بهتر است که در انتهای دستور معرفی متغیر از semicolon استفاده کرد. در صورتی که این عمل را فراموش کنید و برنامه شروع به نشان دادن مقاذیر یک آرایه طولانی نماید کافی است که CONTROL C را فشار دهید تا نشان دادن مقادیر متوقف گردد. همانطور که در بالا دیدید همیشه می توان با نوشتن نام متغیر مقدار آن را مشاهده نمود. همچنین مشاهده می کنید MATLAB یک خط فاصله بین دستورها می گذارد. برای حذف این خطوط اضافی می توانید از دستور زیر استفاده کنید:

```
» format compact
```

اکنون چند بردار تعریف می کنیم:

```
» v=[1 2 3]
v =
    1    2    3
» w=['abcd' '1234']
w =
abcd1234
```

برای تعریف بردارهای عددی حتماً باید از کروشه استفاده کرد ولی استفاده از آنها برای متغیرهای حرفی الزامی نیست. حالت خاصی از بردار (که در توابع MATLAB به عنوان جای خالی استفاده بسیاری دارد) عبارتست از بردار تهی که به صورت [ ] تعریف می گردد.

نحوه تعریف ماتریسها به صورت زیر است:

```
» m=[1 2 3
4 5 6]
m =
    1    2    3
    4    5    6
» n=['abcd'
'1234']
n =
abcd
1234
```

اعضای یک ماتریس را می شود بطور جداگانه مشاهده کرد و یا تغییر داد:

```
» m(2,3)
ans =
    6
```

```
» m(2,3)=7
m =
    1    2    3
    4    5    7
```

عملیات ساده جبری روی بردارها و ماتریسها به صورت زیر انجام می شود:

```
» 2*m
ans =
    2    4    6
    8   10   14

» m+1
ans =
    2    3    4
    5    6    8

» n1=[2 5 4
-1 -2 0];
» m+n1
ans =
    3    7    7
    3    3    7
```

لازم به ذکر است که اعضای یک سطر ماتریس را می توان هم با فاصله و هم با ویرگول از هــم جـدا کرد. بکار بردن semicolon در تعریف یک ماتریس به معنای انتقال به سطر بعدی می باشد:

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

عملگر دو نقطه (:) کاربرد زیادی در رجوع به سطرها، ستونها و یا بخشی از آرایه دارد:

```
» q(1,:)
ans =
    1    2    3
» q(:,2)
ans =
    2
    5
    8
» q(1:2,2:end)
ans =
    2    3
    5    6
```

اگر بخواهید نام متغیرهای ایجاد شده را ببینید می توانید از دستور who استفاده نمایید:

```
» who

Your variables are:

a       n       q       w
m       p       v
```

برای مشاهده نام متغیرهای موجود به همراه اطلاعات اضافه تر در مورد آنها دستور whos را بکار ببرید:

```
» whos
  Name      Size         Bytes  Class

  a         1x1              8  double array
  m         2x3             48  double array
  n         2x4             16  char array
  p         1x5             10  char array
  q         3x3             72  double array
  v         1x3             24  double array
  w         1x8             16  char array

Grand total is 31 elements using 122 bytes
```

برای تولید بردارهای عددی که اعضای آن به فاصله مساوی از هم قرار دارند روش ساده ای در MATLAB وجود دارد. فرض کنید که t برداری باشد که عضو اول آن ۰، عضو آخر آن ۲ و اعضای آن به فاصله مساوی ۰/۵ از یکدیگر باشند

```
» t=0:.5:2
t =
         0    0.5000    1.0000    1.5000    2.0000
```

آرایه های چند بعدی (آرایه هایی که بیش از دو بعد دارند) از امکانات جدید پیش بینی شده در MATLAB 5 هستند. به عنوان مثال می توان بعد سوم را به شکل زیر به ماتریس m که قبلاً تعریف شده افزود:

```
» m(:,:,2)=ones(2,3)
m(:,:,1) =
    1    2    3
    4    5    7
m(:,:,2) =
    1    1    1
    1    1    1
```

افزودن بعدهای چهارم و بیشتر نیز به طریق مشابه امکان پذیر است. اصطلاحاً به بعد سوم صفحه گفته می شود ولی نام خاصی برای ابعاد چهارم به بعد وجود ندارد.

برای بدست آوردن طول یک بردار می توانید از دستور length استفاده کنید:

```
» length(t)
ans =
    5
```

دستور size تعداد سطرها و ستونهای یک ماتریس را نمایش می دهد:

```
» size(n)
ans =
    2    4
```

استفاده از size در مورد آرایه های چند بعدی برداری را می دهد که مولفه های آن طـول آرایـه در هر یک از ابعاد آن است.

برخی از توابعی که در ساختن آرایه ها بکار می روند عبارتند از:

یک ماتریس ۲×۲ با مولفه های ۱ ایجاد می کند	ones(2)
یک ماتریس ۳×۲ با مولفه های ۱ ایجاد می کند	ones(2,3)
یک ماتریس ۳×۲ با مولفه های صفر ایجاد می کند	zeros(2)
یک ماتریس یکه ۳×۳ ایجاد می کند	eye(3)
برداری با ۷ مولفه با فواصل مساوی بین ۱- و ۵ ایجاد می کند	linspace(-1,5,7)
برداری با ۸ مولفه با فواصل لگاریتمی مساوی بین $۱۰^{-۱}$ و $۱۰^{۲}$ ایجاد می کند	linspace(-1,2,8)

تعدادی از توابعی که روی آرایه ها عمل می کنند عبارتند از:

حاصل جمع مولفه های x	sum(x)
حاصل جمع مولفه های x از اول تا هر مولفه	cumsum(x)
حاصلضرب مولفه های x	prod(x)
حاصلضرب مولفه های x از اول تا هر مولفه	cumprod(x)
بزرگترین مولفه x را پیدا می کند	max(x)
کوچکترین مولفه x را پیدا می کند	max(x)
مولفه های x را مرتب می کند	sort(x)
میانگین حسابی مولفه های x	mean(x)
انحراف معیار مولفه های x	std(x)

۲-۲ ذخیره کردن و بازیابی داده ها

در صورتی که بخواهید کلیه متغیرهـــای موجـود در محیـط کـار (workspace) را ذخـیره کنیـد از دستور save استفاده کنید:

» save

Saving to: matlab.mat

این دستور، داده ها را در پرونده matlab.mat ذخیره می نماید. داده های موجود در این پرونـــده را می توان به طریق زیر بازیابی نمود:

» load

Loading from: matlab.mat

درصورتی که لازم باشد می توانید نام پرونده ذخیره را خودتان تعیین کنید:

» save myfile

و آن را با دستور زیر بازیابی نمایید:

» load myfile

اگر می خواهید که فقط بعضی از متغیرها را ذخیره کنید، نام آنها را بعد از نام پرونده بیاورید:

» save myfile t f

در صورتی که بخواهید تعدادی از متغیرها را از حافظه پاک کنیـــد کـافی اسـت نـام آنـها را پـس از دستور clear بیاورید:

» who

Your variables are:

a f n t w
ans m p v

» clear a f
» who

Your variables are:

ans n t w
m p v

در صورت استفاده از دستور clear بدون ذکر نام متغیری پس از آن، کلیه متغیرهـا از حافظـه پـاک می شوند.

توجه کنید که دستور save به صورتی که در بالا نشان داده شد داده ها را به شکل binary ذخیره می نماید و فقط در محیط MATLAB می توانید این داده ها را بازیابی کنید. در این صورت متغیرها با همان نامی که ذخیره شده اند، بازیابی می گردند. در مواردی که نیاز داشته باشید که داده ها را در محیطهای دیگری بازیابی نمایید باید متغیرها را به صورت ascii ذخیره کنید:

```
» save name t -ascii
» clear
» load name
» who

Your variables are:

name
```

همانطور که در بالا مشاهده می کنید هنگام بازیابی یک پرونده ascii نام متغیر، همان نام پرونده خواهد بود. ضمناً پرونده ascii ایجاد شده فاقد دنباله (extension) است مگر آنکه دنباله را در نام پرونده ذکر کنید.

۲-۳ عملیات ماتریسی روی آرایه ها

در MATLAB می توان دو نوع عملیات روی آرایه ها انجام داد که به آنها عملیات ماتریسی و عملیات عضو به عضو می گویند. عملیات ماتریسی شامل محاسبه ترانهاده، ضرب ماتریسی، جمع و تفریق آرایه های هم اندازه و غیره می شود. ترانهاده یک ماتریس با کمک علامت پریم بدست می آید:

```
» r=rand(2,4)
r =
   0.9501   0.6068   0.8913   0.4565
   0.2311   0.4860   0.7621   0.0185
» r'
ans =
   0.9501   0.2311
   0.6068   0.4860
   0.8913   0.7621
   0.4565   0.0185
```

ضرب ماتریسی با استفاده از علامت * و جمع و تفریق ماتریسها با استفاده از علامتهای مربوطه انجام می گیرند:

```
» v=[1:4];
» r*v'
ans =
   6.6636
   3.5634
```

```
» s=[0:3; 2:-.5:.5];
» s+r
ans =
   0.9501   1.6068   2.8913   3.4565
   2.2311   1.9860   1.7621   0.5185
```

تعدادی از توابع ماتریسی در زیر آورده شده اند:

دترمینان ماتریس مربعی	det(a)
ماتریس وارون	inv(a)
مقادیر و بردارهای ویژه ماتریس مربعی	eig(a)
چند جمله ای مشخصه ماتریس	poly(a)

۲-۴ عملیات عضو به عضو روی آرایه ها

انجام عملیات جبری روی آرایه ها در MATLAB نیازمند دقت است. بطور کلی دو نوع عملیات می توان روی آرایه ها انجام داد: ۱-عملیات عضو به عضو، ۲-عملیات برداری-ماتریسی. اشتباه گرفتن این دو نوع عملیات باعث بروز مشکل در محاسبات می گردد. دو بردار زیر را در نظر بگیرید:

```
» a=[1 2 3];
» b=[2 -1 0];
```

فرض کنید که می خواهید این دو را در هم ضرب کنید:

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

دلیل گرفتن پیام خطا از عمل فوق این است که در MATLAB استفاده از علامت ضرب به تنهایی به معنای ضرب ماتریسی است. بنابراین عمل بالا را می توان با ترانهاده بردار دوم به انجام رسانید:

```
» a*b'
ans =
    0
```

این عمل در حقیقت ضرب اسکالر دو ماتریس است، یعنی: ۱×۲+۲×(-۱)+۳×۰=۰

حال اگر بخواهید ضرب عضو به عضو این دو بردار را به دست آورید باید یک نقطه قبل از علامت ضرب بگذارید:

```
» a.*b
ans =
    2   -2    0
```

همین دستورالعمل را می توان برای تقسیم و به توان رساندن آرایه ها بکار بست:

```
» a.^2
ans =
    1    4    9
```

در صورت فراموش کردن نقطه قبل از علامت توان:

```
» a^2
??? Error using ==> ^
Matrix must be square.
```

۵-۲ تنظیم خروجیها روی صفحه نمایش با دستورات disp و format

اگر مقدار یک متغیر را بخواهید بدانید می توانید آن را با نوشتن نام متغیر مشاهده کنید. در این صورت MATLAB نام متغیر و به دنبال آن علامت تساوی را نشان داده و سپس مقدار را در سطر یا سطور بعد می نویسد. برای دیدن مقدار متغیر بدون آنکه لازم باشد دوباره نام آن و علامت تساوی را مشاهده کنید می توانید دستور disp را بکار ببرید.

```
» x=[2 4 5];
» disp(x)
    2    4    5
» y='That is better';
» disp(y)
That is better
```

پنجره MATLAB را می توانید با دستور clc پاک کنید:

```
» clc
```

همانطور که قبلاً دیدید دستور format compact باعث می شود که خطوط اضافی هنگام ارائه نتایج حذف گردند. دستور format دارای کاربردهای فراوان دیگری نیز هست. فرض کنید که می خواهید مولفه های بردار زیر را روی صفحه نمایش ببینید:

```
» v=exp(-10*(1:5))
v =
  1.0e-004 *
   0.4540    0.0000    0.0000    0.0000    0.0000
```

واضح است که در حالت فعلی نمی توانید مقادیر مولفه ها را بخوانید. در این وضعیت می توانید با کمک دستور format نحوه نمایش اعداد را تغییر دهید:

```
» format long
» v
v =
  1.0e-004 *
  Columns 1 through 4
   0.45399929762485   0.00002061153622   0.00000000093576   0.00000000000004
  Column 5
   0.00000000000000
```

مشاهده می کنید با وجود اینکه این دستور تعداد اعداد نشان داده شده بعد از ممیز را افزایش می دهد ولی هنوز قادر نیست که همه مولفه های بردار مورد نظر را بطور مناسبی نمایش دهد. در چنین حالتی بهتر است اعداد را با استفاده از نماد علمی به نمایش بگذارید:

```
» format short e
» v
v =
  4.5400e-005  2.0612e-009  9.3576e-014  4.2484e-018  1.9287e-022
```

برای اطلاع بیشتر از امکانات دستور format توصیه می شود که توضیحات مربوط به این دستور را در help مطالعه کنید.

۳- چند جمله ایها

یک چند جمله ای در MATLAB به صورت یک بردار سطری که مولفه های آن ضرایب چند جمله ای به ترتیب نزولی هستند معرفی می شود. برای مثال چند جمله ای $p(x) = x^3 - 2x + 5$ در MATLAB به شکل زیر معرفی می گردد:

```
» p=[1  0  -2  5];
```

۳-۱ ریشه های یک چند جمله ای

ریشه های یک چند جمله ای را می توانید به صورت زیر بدست آورد:

```
» r=roots(p)
r =
  -2.0946
   1.0473 + 1.1359i
   1.0473 - 1.1359i
```

با دانستن ریشه های معادله می توانید ضرایب چند جمله ای مربوطه را محاسبه نمائید:

```
» p2=poly(r)
p2 =
   1.0000    0.0000   -2.0000    5.0000
```

۳-۲ محاسبه مقدار یک چند جمله ای

تابع polyval مقدار چند جمله ای را در هر نقطه محاسبه می نماید. برای مثال مقدار p(۵) به طریق زیر محاسبه می گردد:

```
» polyval(p,5)
ans =
  120
```

۳-۳ ضرب و تقسیم چند جمله ایها

برای ضرب و تقسیم چند جمله ایها می توانید توابع conv و deconv را بکار ببرید. چند جمله ایهای $a(x)=x^2+x+1$ و $b(x)=x-1$ را در نظر بگیرید. حاصلضرب این دو چند جمله ای به طریق زیر بدست می آید:

```
» a=[1  1  1]; b=[1  -1];
» c=conv(a,b)
c =
   1    0    0   -1
```

و تقسیم a/b نیز به صورت زیر قابل محاسبه است:

```
» [q,r]=deconv(a,b)
q =
   1    2
r =
   0    0    3
```

۳-۴ مشتق چند جمله ای

مشتق چند جمله ای را می توانید با بکار بردن تابع polyder محاسبه کنید.

```
» c=polyder(a)
c =
   2    1
```

مشتق حاصلضرب دو چند جمله ای (a×b) را می توانید به صورت زیر بدست آورید:

```
» d=polyder(a,b)
d =
   3    0    0
```

در صورتی که تعداد آرگومانهای خروجی تابع polyder برابر ۲ باشد، تابع مشتق تقسیم دو چند جمله ای (a/b) را تعیین می نماید:

```
» [q,d]=polyder(a,b)
q =
    1    -2    -2
d =
    1    -2     1
```

۴-۳ برازش منحنی چند جمله ای

تابع polyfit ضرایب بهترین چند جمله ای را پیدا می کند که از میان مجموعه نقاط داده شده عبور می نماید. به عنوان مثال مجموعه نقاط زیر را در نظر بگیرید:

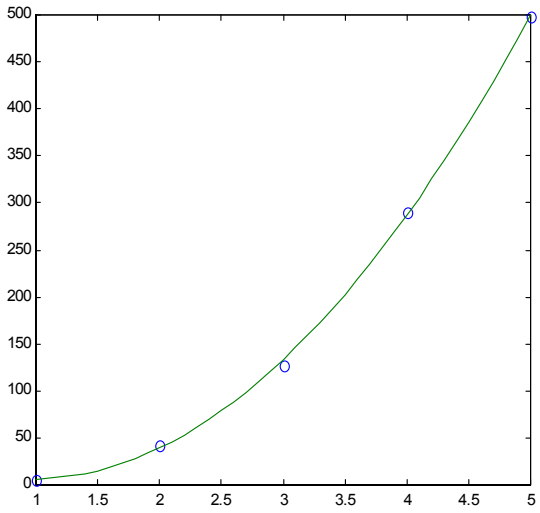
```
» x=[1  2  3  4  5];
» y=[5.5  43.1  128  290.7  498.4];
```

دستور زیر ضرایب بهترین چند جمله ای درجه سوم را محاسبه می کند که از بین نقاط فوق می گذرد:

```
» p=polyfit(x,y,3)
p =
  -0.1917   31.5821  -60.3262   35.3400
```

حال می توانید برای مقایسه منحنی محاسبه شده و داده های اولیه را در یک نمودار رسم کنید:

```
» x2=1:.1:5;
» y2=polyval(p,x2);
» plot(x,y,'o',x2,y2)
```



۴- عملیات و توابع منطقی

۴-۱ مقایسه منطقی

در MATLAB علامتهای زیر برای مقایسه مقادیر عددی و حرفی بکار می روند.

کوچکتر از	<
کوچکتر از یا مساوی با	=<
بزرگتر از	>
بزرگتر از یا مساوی با	>=
مساوی با	==
مخالف با	~=

چنین مقایسه ای را می توان بین دو اسکالر، دو آرایه یا اسکالر و اعضای آرایه انجام داد. مثالهایی برای نحوه عمل این عملگرها در زیر آورده می شوند. توجه کنید که حاصل همه عملیات منطقی می تواند ۰ به معنی نادرست یا ۱ به معنی درست باشد.

```
» 3<5
ans =
    1

» [1 2]>=[0 3]
ans =
    1    0

» a=[1 2 3
2 3 4];
» b=[-1 2 1
0 2 4];
» a~=b
ans =
    1    0    1
    1    1    0
```

بردار زیر را در نظر بگیرید:

```
» x=[1  2  -1  0  -5  4  -1.5  3  2.5  -.5];
```

عبارت زیر مولفه های مثبت این بردار را نمایش می دهد:

```
» x(x>0)
ans =
   1.0000    2.0000    4.0000    3.0000    2.5000
```

و این عبارت تعداد مولفه هایی را که بین صفر و ۳ هستند تعیین می کند:

```
» length(x((x>=0)&(x<=3)))
ans =
    5
```

۲-۴ عملگرهای منطقی

روابط منطقی را می توان با استفاده از عملگرهای منطقی با هم ترکیب کرد. این عملگرها عبـــارتند از:

و (ترکیب عطفی)	&
یا (ترکیب فصلی)	|
یا (مانع جمع)	xor
نقیض	~

مثالهایی از طرز عمل این عملگرها در زیر آورده شده اند:

```
» m=[1  2  4;  -2  3  -1];
» ~(m>0)
ans =
    0    0    0
    1    0    1
» (m>0)|(m<=2)
ans =
    1    1    1
    1    1    1

» (m>0)&(m<=2)
ans =
    1    1    0
    0    0    0

» xor([0 0 1 1],[0 1 0 1])
ans =
    0    1    1    0
```

توجه کنید که xor یک تابع است و دو بردار ورودی به آن باید هم اندازه باشند.

۳-۴ توابع منطقی any، all و find

تابع any معین می کند که آیا مولفه غیر صفر در یک بردار وجود دارد یا خیر.

```
» v=[-2  1  3  5];
» any(v<1)
ans =
    1
```

```
» any(v>6)
ans =
    0
```

تابع all معین می کند که آیا همه مقایسه ها درست هستند یا خیر.

```
» all(v<1)
ans =
    0

» all(v<6)
ans =
    1
```

توابع فوق را می توانید برای ماتریسها نیز بکار ببرید. در این صورت خروجی این توابع عبارت است از حاصل مقایسه های گفته شده برای هر ستون ماتریس.

تابع find مکان مولفه های غیر صفر را در یک آرایه نشان می دهد.

```
» find(v>3)
ans =
    4
```

۴-۴ اعداد ویژه

علاوه بر اعداد حقیقی، MATLAB قادر است حاصل عباراتی را که از نظر جبری انجام پذیر نیستند را نیز پوشش دهد. تقسیم یک عدد بر صفر بی نهایت (Inf) و تقسیم صفر بر صفر غیر قابل محاسبه است (NaN یا Not a Number).

```
» x=[1  2  0]./[2  0  0]
Warning: Divide by zero.
x =
   0.5000       Inf       NaN
```

همچنین اعداد موهومی را به سادگی اعداد حقیقی می توانید در محاسبات استفاده کنید.

```
» y=sqrt(-1)
y =
      0 + 1.0000i
```

توابع finite، isinf، isnan و isreal به شما امکان می دهد که این اعداد را شناسایی کنید:

```
» finite(x)
ans =
    1     0     0
```

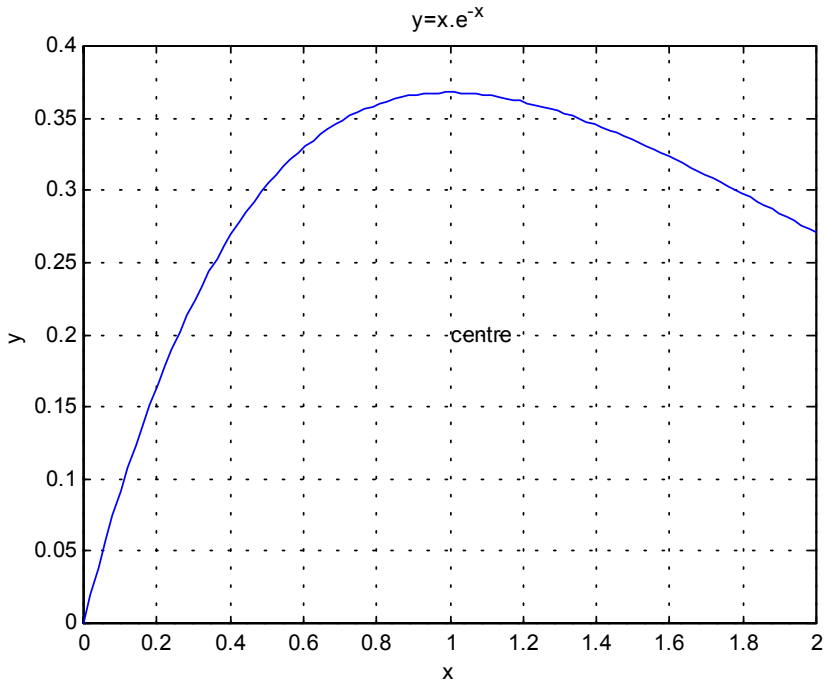
```
» isinf(x)
ans =
    0     1     0

» isnan(x)
ans =
    0     0     1

» isreal(x)
ans =
    1

» isreal(y)
ans =
    0
```

۵- ترسیم داده ها

۵-۱ نمودارهای ۲ بعدی

مجموعه دستورات زیر نحوه ترسیم یک تابع بر حسب یک متغیر مستقل را نشان می دهد:

```
» x=linspace(0,2); y=x.*exp(-x);
» plot(x,y)
» grid
» xlabel('x')
» ylabel('y')
» title('y=x.e^{-x}')
» text(1,.2,'centre')
```



هفت خط فوق به ترتیب اعمال زیر را انجام می دهند:

۱- بردار متغیرهای مستقل (x) و تابع (y) را ایجاد می کند.

۲- مقادیر y را بر حسب x رسم می نماید.

۳- شبکه را به نمودار می افزاید.

۴- توضیح محور افقی را می نویسد.

۵- توضیح محور عمودی را می نویسد.

۶- تیتر نمودار را در بالای آن می نویسد.

۷- در نقطه مورد نظر (در این مثال نقطه (۰/۲ و ۱)) متغیر حرفی مشخص شده (در این مثال centre) را می نویسد.

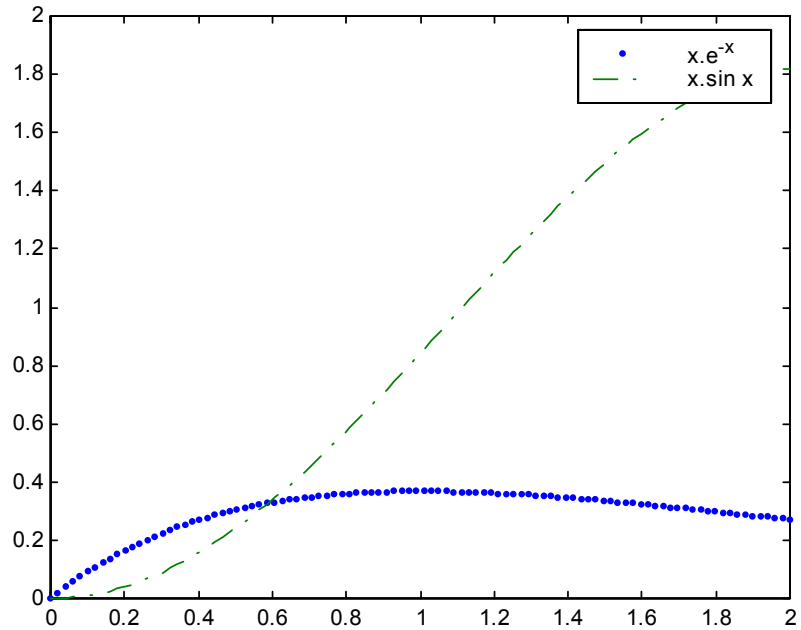
می توانید نمودار ایجاد شده را به کمک دستور Save As در منوی File پنجره نمودار، ذخیره نمایید. این دستور نمودار را در یک پرونده که نام آن را خودتان وارد خواهید کرد و دنباله آن ffig. می باشد ذخیره می کند. شما می توانید این نمودار را در دفعات بعدی کار با MATLAB با استفاده از دستور open بازیابی نمایید.

در هنگام رسم نمودارها می توانید از علامتهای مختلف (بجای خط) برای رسم توابع استفاده کنید. همچنین می توانید بیش از یک تابع را در یک نمودار نمایش دهید.

```
» plot(x,y,'.',x,x.*sin(x),'-.')
```

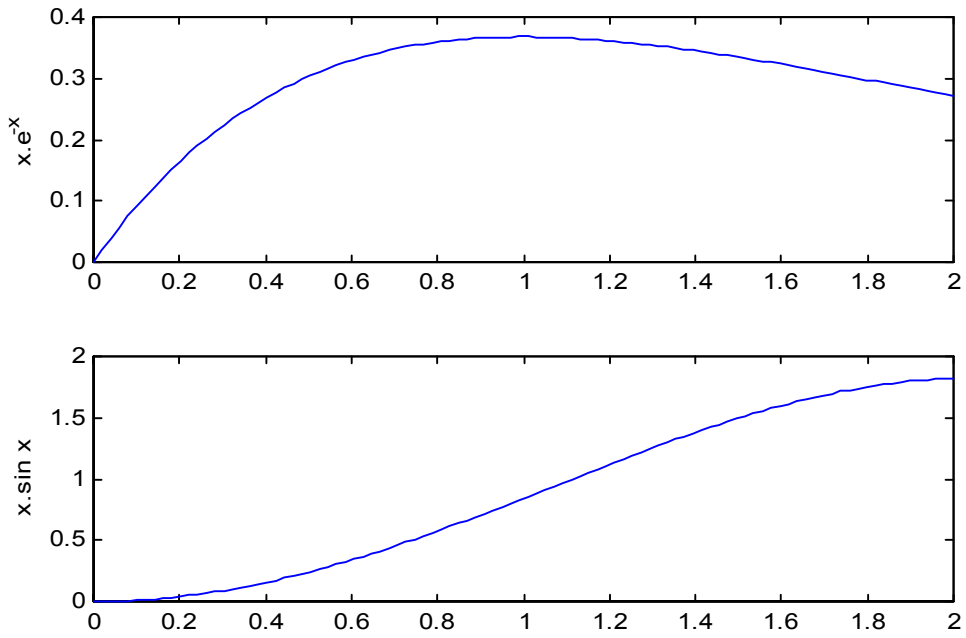
و در صورت لزوم نام توابع را نیز در همان نمودار نشان دهید.

```
» legend('x.e^{-x}','x.sin x')
```



می توانید بیش از یک نمودار را در یک پنجره نشان دهید:

```
» subplot(2,1,1), plot(x,y)
» ylabel('x.e^{-x}')
» subplot(2,1,2), plot(x,x.*sin(x))
» ylabel('x.sin x')
```



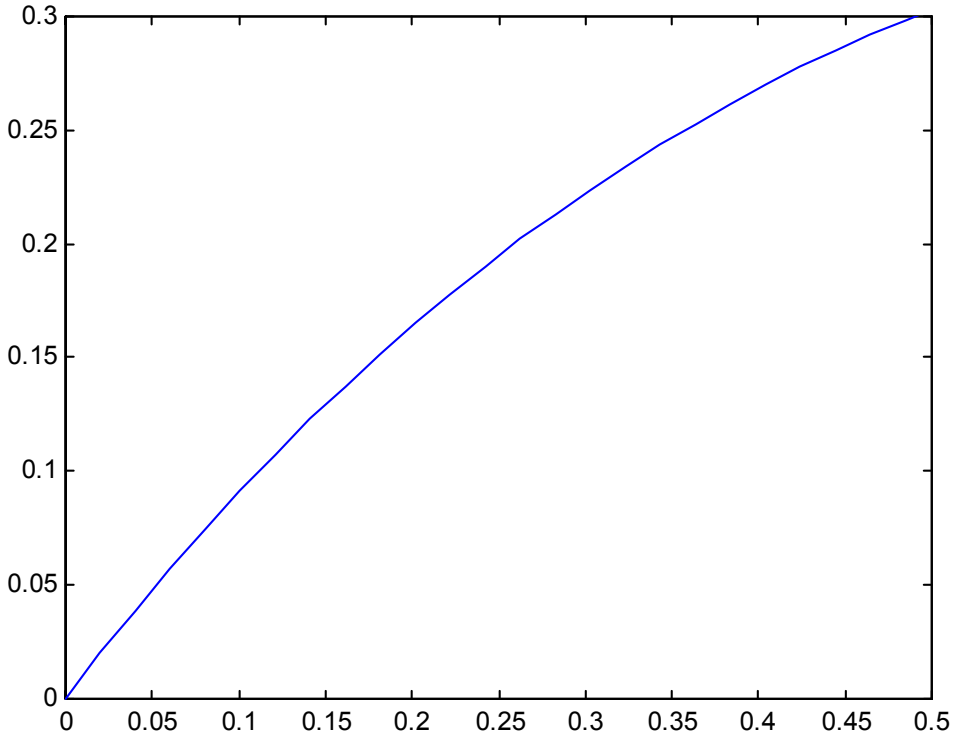
دو عدد اول در دستور subplot تعداد تقسیمات صفحه را معین می کنند (سطری و ستونی) و عـدد سوم مکان رسم نمودار (یا تغییر روی نمودار موجود) را مشخص می نماید.

نمودار را می توانید با استفاده از دستور clf پاک کنید.

```
» clf
```

با استفاده از دستور figure می توانید پنجره جدیدی برای رسـم نمـودار بـاز نمـائید. دسـتور axis حدود بالا وپایین محورهای مختصات را به صورت یک بردار ارائه می نماید.

```
» figure(2)
» plot(x,y)
» axis
ans =
      0    2.0000         0    0.4000
```



در تمامی مثالهای بالا مقادیر متغیر مستقل و متغیر وابسته به صورت دو بردار بر حسب هم رسم شده اند. در صورتی که تابعیت متغیر وابسته بر حسب متغیر مستقل مشخص باشد می توانید از دستور fplot برای رسم آن استفاده کنید:

```
» fplot('x*exp(-x)',[0  2])
```

آرگومان اول این دستور یک بردار حرفی است که مشخص کننده رابطه تابع (در صورت ساده بودن رابطه تحلیلی تابع، همانند مثال فوق) یا نام m-file حاوی تابع (که جداگانه باید ایجاد شده باشد) است. آرگومان دوم fplot یک بردار دو عضوی است که حد پائین و بالای متغیر مستقل را مشخص می کند.

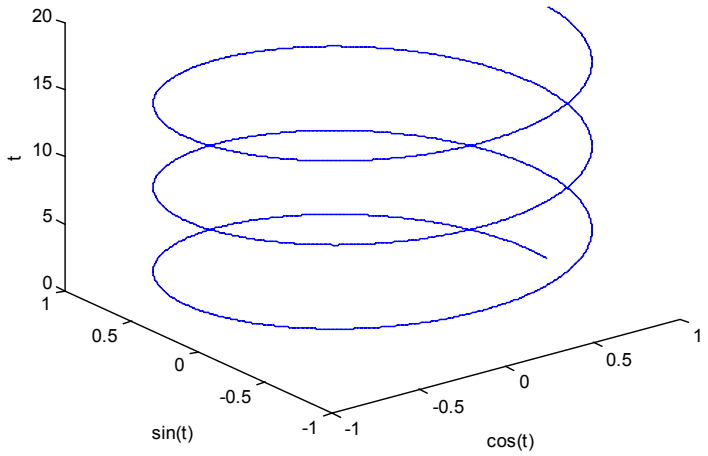
تعدادی از دستورهای ترسیم دو بعدی در زیر آورده شده اند:

نمودار نیمه لگاریتمی (محور x لگاریتمی)	semilogx(x,y)
نمودار نیمه لگاریتمی (محور y لگاریتمی)	semilogy(x,y)
نمودار تمام لگاریتمی	loglog(x,y)
رسم در دستگاه مختصات قطبی	polar(r,theta)
نمودار میله ای	bar(x,y)
نمودار مساحت	area(x,y)

۲-۵ نمودارهای ۳ بعدی

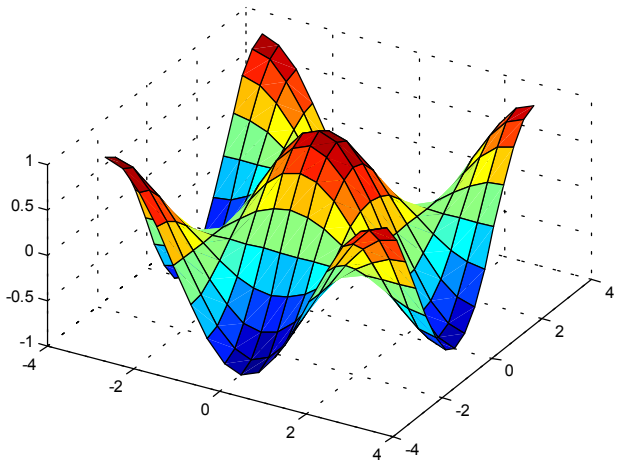
دستورهای زیادی در MATLAB برای ترسیم نمودارهای سه بعدی وجود دارند. یک منحنی سه بعدی را می توانید به کمک دستور plot3 ببینید:

```
» t=0:.01:6*pi;
» plot3(cos(t),sin(t),t)
» xlabel('cos(t)')
» ylabel('sin(t)')
» zlabel('t')
```



سطوح سه بعدی را می توانید با استفاده از دستور surf ترسیم کنید:

```
» [x,y]=meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
» z=cos(x).*cos(y);
» surf(x,y,z)
» view(30,45)
```



دستور meshgrid شبکه دو بعدی روی صفحه xy را ایجاد می کند. بردارهای ورودی به این دستور مشخص کننده تقسیمات در جهات x و y هستند. سطح ایجاد شده را می توانید با کمک دستور shading هموار کنید. همچنین برای تطابق رنگها با اعداد محور z می توانید از دستور colorbar استفاده کنید.

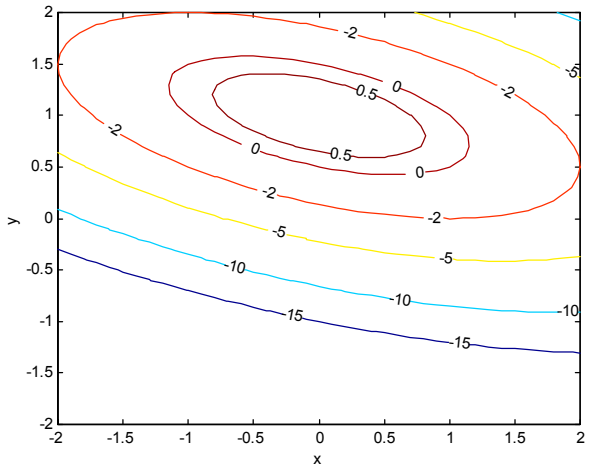
```
» shading interp
» colorbar
```

برای رسم سطوح سه بعدی از دستورات دیگری مانند mesh، meshc، meshz و waterfall نیز می توانید کمک بگیرید.

۳-۵ نمودارهای ۲/۵ بعدی

نمودارهای به اصطلاح ۲/۵ بعدی برای دیدن سطوح ۳ بعدی روی صفحه مختصات ۲ بعدی بکار می روند. یکی از این روشها رسم خطوط همتراز یک سطح است.

```
» [x,y]=meshgrid(-2:.1:2,-2:.1:2);
» z=2-((x-1).^2+4*(y-1).^2+2*x.*y);
» [c,h]=contour(x,y,z,[-15  -10  -5  -2  0  0.5]);
» clabel(c,h), xlabel('x'), ylabel('y')
```



آرگومان چهارم در دستور contour برداری است که بر اساس آن منحنیهای همترازی که مقادیر عددی آنها برابر با مولفه های آن بردار است روی نمودار نشان داده خواهند شد. دستور clabel مقادیر خطوط همتراز را روی نمودار نشان می دهد.

روش دیگر آن است که سطح را از زاویه ای عمود بر صفحه xy نگریست و رنگهای متفاوتی به مقادیر مختلف z نسبت داد:

```
» pcolor(x,y,z)
» shading interp
» colorbar
```

۶- برنامه نویسی (m-files)

مجموعه ای از دستورات MATLAB را می توانید در یک پرونده ذخیره کنید و سپس آنها را یکجا اجرا نمائید. چنین پرونده ای برای آنکه در محیط MATLAB قابل اجرا باشد باید حتماً دارای

دنباله ".m" باشد. در صورتی که از ویرایشگر MATLAB (MATLAB Editor) استفاده کنید، دنباله ".m" بطور خودکار در هنگام ذخیره پرونده به نام آن افزوده می گردد. در صورت استفاده از ویرایشگر دیگری بغیر از ویرایشگر MATLAB (نظیر Notepad) اطمینان حاصل کنید که پرونده حتماً به روش ascii و با دنباله ".m" ذخیره گردد.

در این بخش از یادداشت فقط بر نحوه برنامه نویسی و اجرای برنامه ها تاکید شده است و نتایج اجرای برنامه های مورد بحث نشان داده نشده اند. به خواننده توصیه می گردد که خود برنامه ها را اجرا کرده و نتایج آنها را مشاهده نماید.

۶-۱ برنامه اصلی

m-file ها می توانند به دو شکل برنامه اصلی و تابع باشند. برنامه اصلی عبارتست از مجموعه ای از دستورها که می توان آنها را بطور جداگانه در محیط کار MATLAB اجرا نمود. هنگامی که نام برنامه اصلی را در محیط کار MATLAB بنویسید این دستورها به ترتیب اجرا می گردند. به عنوان مثال برای محاسبه حجم گاز کامل، در دماهای مختلف و فشار معلوم، دستورات زیر را در ویرایشگر MATLAB بنویسید و سپس تحت عنوان pvt.m ذخیره کنید:

```
% A sample scritp file: pvt.m
disp(' Calculating the volume of an ideal gas.')
R = 8314;    % Gas constant (J/kmol.K)
t = ...
   input(' Vector of temperature (K) = ');
p = input(' Pressure (bar) = ')*1e5;
v = R*t/p;      % Ideal gas law
% Plotting the results
plot(t,v)
xlabel('T (K)')
ylabel('V (m^3/kmol)')
title('Ideal gas volume vs temperature')
```

علامت ٪ نشانگر وجود توضیحات در برنامه است. علامت ٪ و آنچه بدنبال آن در همان سطر می آید به هنگام اجرای برنامه نادیده گرفته می شود. همچنین علامت . . . بیانگر آن است که دستور مورد نظر در این سطر تمام نشده و در سطر بعدی ادامه می یابد. مورد استفاده این علامت بیشتر در مورد دستورهای محاسباتی طولانی است که برای مطالعه راحت تر این قسمت از برنامه بهتر است در دو یا سه خط نوشته شود.

پس از ایجاد پرونده pvt.m، برای اجرای آن کافی است که نام آن را در محیط کار MATLAB بنویسید و نتایج را مشاهده کنید (نمودار در زیر نشان داده نشده است).

```
» pvt
 Calculating the volume of an ideal gas.
 Vector of temperature (K) = 100:25:300
 Pressure (bar) = 10
```

۶-۲ استفاده از diary برای ایجاد برنامه

یک روش ایجاد برنامه برای مبتدیان بکار بردن دستور diary است. در صورت استفاده از دستور زیر

```
» diary xyz
```

تمامی نوشته های محیط کار MATLAB پس از آن در پرونده xyz حک می گردند. پرونده xyz بدون دنباله خواهد بود مگر آنکه خودتان برای آن دنباله مشخص کنید. در این حالت می توانید شروع به نوشتن دستورات مورد نظر در محیط کار MATLAB کنید، نتایج را همان جا ببینید و در صورت لزوم تصحیحات لازم را انجام دهید. هنگامی که به پایان محاسبات و نتیجه دلخواه رسیدید، پرونده xyz را به کمک دستور زیر ببندید:

```
» diary off
```

اکنون می توانید پرونده xyz را باز کرده، خطوط و دستورهای اضافی را از آن پاک کنید و سپس با دنباله m. آن را ذخیره نمائید. به این ترتیب یک m-file ایجاد کرده اید که به نتایج اجرای آن اطمینان دارید.

۶-۳ تابع

علاوه بر توابعی که همراه MATLAB هستند، شما می توانید توابعی را که محاسبات مورد نیازتان را انجام بدهد نیز ایجاد کنید. یک تابع یک یا چند داده را در ورودی دریافت می کند و پس از انجام محاسبات لازم نتایج را در قالب یک یا چند متغیر خروجی به شما برمی گرداند. خط اول یک تابع که خط تعریف تابع نیز نامیده می شود باید از ترتیب زیر پیروی نماید:

- کلمه function
- نام متغیر یا متغیرهای خروجی. در صورت وجود بیش از یک متغیر خروجی باید آنها را در کروشه گذاشته و با ویرگول از هم جدا کنید.
- علامت =
- نام تابع. پرونده ای که تابع در آن ذخیره می گردد باید دارای همین نام با دنباله m. باشد.
- آرگومان یا آرگومانهای ورودی (که با ویرگول از هم جدا شده باشند) در داخل پرانتز.

برای مثال تابع زیر، که باید در پرونده ideal.m ذخیره گردد، حجم گاز کامل را در فشارها و دماهای مختلف محاسبه می نماید:

```
function v = ideal(t,p)
% ideal: Calculation of ideal gas specific volume
% v=ideal(t,p) takes the vector of temperature (t) in K
% and the vector of pressure (p) in Pa and returns the
% matrix of specific volume (v) in m3/kmol.

% Start of calculations
R = 8314;       % Gas constant (J/kmol.K)
for k = 1:length(p)
   v(k,:) = R*t/p(k);     % Ideal gas law
end
```

حال این تابع را می توانید در محیط کار MATLAB، در یک برنامه اصلی و یا در تابع دیگری بکـار ببرید. مثلاً (نتایج در اینجا نشان داده نشده اند):

```
» p=1:10; t=300:10:400;
» vol=ideal(t,p);
» surf(t,p,vol)
» view(135,45), colorbar
```

توصیه می شود در توابعی که می نویسید، پس از خط تعریف تابع، کار تابع و نحـوه بکـاربردن آن را در چند خط توضیح دهید. خطوط توضیح پیوسته ای که در ابتدای تـابع مـی آینـد را مـی توانیـد همانند دیگر توابع و دستورهای موجود در MATLAB با استفاده از دستور help مرور کنید.

```
» help ideal

  ideal: Calculation of ideal gas specific volume
  v=ideal(t,p) takes the vector of temperature (t) in K
  and the vector of pressure (p) in Pa and returns the
  matrix of specific volume (v) in m3/kmol.
```

۴-۶ کنترل جریان محاسبات

MATLAB دارای چندین ترکیب کنترل جریان محاسبات است که به برنامه امکان می دهد که در حین اجرا تصمیمات لازم را اتخاذ کرده و ترتیب اجرای دستورات را کنترل کند. ایـن دسـتورها در زیر شرح داده می شوند.

if . . . (else . . .) end – دستور if برنامه را قادر می سازد که تصمیم بگیرد که چه دستورهایی باید اجرا گردند. مثال:

```
x = input(' x = ');
if x >= 0
   y=x^2
end
```

عبارتی که به دنبال کلمه if می آید باید یک عبارت منطقی باشد. در صورت درست بودن این عبارت منطقی، دستورهایی که در سطرهای بین if و end قرار دارند بترتیب اجرا می گردند و در صورت نادرست بودن این عبارت منطقی، دستورهای گفته شده نادیده گرفته می شوند.

شما همچنین می توانید از دستور else استفاده کنید. مثال:

```
x = input(' x = ');
if x >= 0
   y=x^2
else
   y=-x^2
end
```

در این حالت اگر عبارت منطقی مورد نظر درست باشد، مجموعه دستورهای بین if و else اجرا می گردند و در غیر این صورت دستورهای بین else و end قابل اجرا می باشند.

for . . . end – دستور for به برنامه اجازه می دهد که دستورهای درج شده بین for و end را به دفعات معینی تکرار نماید. مثال:

```
k = 0;
for x = 0:0.2:1
   k = k + 1
   y = exp(-x)
end
```

while . . . end – در مواردی که لازم باشد که در حین اجرای برنامه مجموعه ای از دستورات تکرار گردند ولی تعداد دفعات تکرار معلوم نباشد بلکه این عملیات تا ارضا شدن شرط یا شروط معینی ادامه یابند، می توان از دستور while استفاده نمود. مثال:

```
x = 0;
while x < 1
   y = sin(x)
   x = x + 0.1;
end
```

همانند آنچه در مورد دستور if گفته شد، عبارتی که به دنبال کلمه while می آید باید یک عبارت منطقی باشد که در واقع همان شرط مورد نظر است. در صورت صادق بودن این عبارت منطقی، دستورهایی که در سطرهای بین while و end قرار دارند بترتیب اجرا می گردند تا آنجایی که شرط مورد نظر دیگر برقرار نباشد.

switch . . . case . . . (otherwise . . .) end – وقتی که لازم باشد که برنامه بر حسب مقادیر مختلف یک متغیر، متناظراً دستورهای متفاوتی را اجرا کند، بکار بردن ترکیب switch-case راحت تر از بکار بردن چندین دستور if متداخل است. مثال:

```
a = input('a =');
switch a
case 1
   disp('One')
case 2
   disp('Two')
case 3
   disp('Three')
end
```

break و pause - دو دستور مفید دیگر که در برنامه نویسی می توانند مورد استفاده قرار گیرند عبارتند از break و pause. شما می توانید در صورت لزوم قبل از کامل شدن حلقه به کمک دستور break از آن خارج شوید. هنگامی که برنامه در حین اجرا به دستور pause برسد متوقف می ماند تا اینکه شما کلیدی را روی صفحه کلید فشار دهید و سپس اجرای برنامه از دستور بعد از pause ادامه می یابد. مثال:

```
k = 0;
for x = 0:0.2:1
   if k > 5
      disp('k > 5')
      break
   end
   k = k + 1;
   y = exp(-x);
   disp([' k = ',num2str(k),'  y = ',num2str(y)])
   pause
end
```

در مثال فوق، برنامه هر بار پس از نشان دادن مقادیر k و y متوقف می ماند تا اینکه کلیدی روی صفحه کلید فشرده شود. سپس حلقه for بار دیگر تکرار می گردد و این عمل آنقدر ادامه می یابد تا اینکه مقدار k از ۵ بیشتر شود. در این موقع دستور break باعث خروج برنامه از حلقه for (و در این مثال پایان اجرای برنامه) می شود.

۷- خطایابی برنامه ها

شما می توانید از راههای زیر، برنامه هایتان را خطایابی (debugging) نمائید:

- برنامه را به چند بخش کوتاهتر تقسیم کنید و هر بخش را جداگانه امتحان کنید.
- نتایج محاسبات را در مراحل میانی جریان برنامه بنویسید. این کار را می توانید به آسانی با برداشتن (;) semicolon از انتهای دستور محاسباتی ویا نوشتن نام متغیر مورد نظر انجام دهید.

همچنین می توانید با قرار دادن disp در مکانهای مشخصی از برنامه دریابید که برنامه تـا کجـا به پیش رفته است.

- تا حد امکان سعی کنید که از عملیات ماتریسی استفاده کنید و در برنامه از تعداد حلقـه هـایی که همان کار را انجام می دهند بکاهید.
- خطوط مورد شک برنامه را بطور جداگانه در محیط کار MATLAB اجـرا کنیـد (ترجیحـا" بـه کمک copy-paste) تا درستی و یا نادرستی محاسبه را دریابید.
- دقت کنید که پیغام خطا روی چه سطری از برنامه داده شده است و بویژه دقت کنید که پیغـام خطا چه می باشد و چه معنایی دارد.
- امکانات خطایابی موجود در نرم افزار را به کمک بگیرید.

۱-۷ پیغامهای خطا

بیشترین حجم پیغامهای خطایی که شما در ابتدای کار با MATLAB دریافت می کنید مربوط بـه عملیات و جایگزینی های برداری/ماتریسی است. در این بخش نحوه تصحیح برنامه را با اسـتفاده از پیغامهای خطای دریافتی با ذکر یک مثال نشان داده می شود.

در نظر بگیرید که می خواهید سطح PVT را بر اسـاس قـانون گـاز کـامل رسـم کنـید. داده هـای ورودی به برنامه محدوده های فشار و دما به صورت برداری هستند و برنامه باید حجـم ویـژه گـاز را محاسبه نماید و سپس سطح را رسم کند. بهتر است که محاسبه حجم در یک تابع جداگانـه انجـام گیرد تا اگر بخواهید محاسبه را با معادله حالت دیگری نیز تکـرار کنیـد، نیـازی بـه نوشـتن مجـدد برنامه اصلی نداشته باشید و فقط تابع محاسبه حجم را تغییر دهید. فرض کنید که برنامـه اصلـی و تابع مورد نیاز را در وهله اول به صورت زیر ایجاد کرده اید:

برنامه اصلی (main.m)

```
% Input
p = input(' Pressure (bar) = ');
t = input(' Temperature (K) = ');

% Calculation
v = ideal(t,p*1e5);

% Plotting results
surf(p,vol,t)
```

تابع (ideal.m)

```
function v = ideal(t,p)

R = 8314;       % Gas constant (J/kmol.K)
v = R*t/p;        % Ideal gas law
```

حال در صورتی که این برنامه را اجرا کنید، پیغام خطای زیر را دریافت می کنید:

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
??? Error using ==> /
Matrix dimensions must agree.

Error in ==> C:\MATLABR11\work\ideal.m
On line 4  ==> v = R*t/p;                    % Ideal gas law

Error in ==> C:\MATLABR11\work\main.m
On line 6  ==> v = ideal(t,p*1e5);
```

همانطور که ملاحظه می کنید اشکال از سطر ۶ برنامه اصلی که مربوط بـه مراجعـه بـه تـابع اسـت گرفته شده و در حقیقت خطا در سطر ۴ تابع و مشخصاً در نحوه تقسیم دو بردار t و p وجـود دارد. به یاد بیاورید که در عملیات ماتریسی، ابعاد ماتریسها باید اجازه انجـام چنیـن عملـی را بدهـد. در اینجا با دو بردار t و p نمی توان عمل تقسیم را انجام داد و اصولاً در این مسـئله مقصـود از عبـارت بکار برده شده برای محاسبه حجم گاز کامل انجام محاسبه ماتریسی نمی باشـد. بنـابراین سـطر ۴ تابع ideal.m به شکل زیر تغییر داده می شود (بکار بردن تقسیم عضو به عضو بجـای ماتریسـی) تـا محاسبه حجم به صورت ماتریسی صورت نگیرد:

```
function v = ideal(t,p)

R = 8314;       % Gas constant (J/kmol.K)
v = R*t./p;       % Ideal gas law
```

اما با اجرای مجدد برنامه می بینید که مشکل حل نشده است:

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
??? Error using ==> ./
Matrix dimensions must agree.

Error in ==> C:\MATLABR11\work\ideal.m
On line 4  ==> v = R*t./p;                   % Ideal gas law

Error in ==> C:\MATLABR11\work\main.m
On line 6  ==> v = ideal(t,p*1e5);
```

اگر تعداد مولفه های بردارهای t و p را در محیط کار MATLAB بخواهیم:

```
» length(p)
ans =
   10
» length(t)
ans =
   21
```

دیده می شود که این دو بردار هم اندازه نیستند و بنابراین عملیات عضو به عضو نیز نمی توان بر روی آن دو انجام داد. در اینجا چاره ای نیست جز آنکه از یک حلقه در محاسبات استفاده نمائید و مقادیر حجم ویژه را بر حسب دما، هر بار در یک فشار معین، محاسبه نمائید:

```
function v = ideal(t,p)

R = 8314;       % Gas constant (J/kmol.K)
for k = 1:length(p)
   v(:,k) = R*t/p(k);       % Ideal gas law
end
```

اما این بار نیز با پیغام خطا مواجه می شوید:

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
??? In an assignment  A(:,matrix) = B, the number of elements in the subscript of A
and the number of columns in B must be the same.

Error in ==> C:\MATLABR11\work\ideal.m
On line 5  ==>    v(:,k) = R*t/p(k);                    % Ideal gas law

Error in ==> C:\MATLABR11\work\main.m
On line 6  ==> v = ideal(t,p*1e5);
```

توجه کنید که بردار دما یک بردار سطری است و در نتیجه سمت راست عبارت محاسبه حجم یک بردار سطری خواهد بود. این در حالی است که در سمت چپ همان عبارت یک بردار ستونی قرار دارد و پیغام خطا نیز از همینجا ناشی می شود. بنابراین تابع ideal.m باید به شکل زیر تصحیح گردد:

```
function v = ideal(t,p)

R = 8314;       % Gas constant (J/kmol.K)
for k = 1:length(p)
   v(k,:) = R*t/p(k);       % Ideal gas law
end
```

این بار با اجرا کردن برنامه اصلی پیغام زیر را مشاهده می کنید:

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
??? Undefined function or variable 'vol'.

Error in ==> C:\MATLABR11\work\main.m
On line 9  ==> surf(p,vol,t)
```

باز هم پیغام خطا! اما اگر دقت کنید می بینید که این بار پیغام خطا مربوط به تابع ideal.m نیست بلکه خطا از دستور مربوط به رسم داده ها گرفته شده است. در حقیقت تابع کار خود را بـه خوبـی انجام داده و رفع اشکال شده است. خطای این دفعه مربوط به اشتباه در نام متغیر اسـت. متغـیر v که قبلاً تعریف شده است اشتباهاً در دستور surf با نام vol بکار برده شده اسـت. ولـی vol قبـلاً تعریف نشده است و در نتیجه MATLAB آن را نمی شناسد. پـس از تصحیـح ایـن سـطر، برنامـه اصلی به صورت زیر خواهد بود:

```
% Input
p = input(' Pressure (bar) = ');
t = input(' Temperature (K) = ');

% Calculation
v = ideal(t,p*1e5);

% Plotting results
surf(p,v,t)
```

اجرای این برنامه پیغام زیر را به دنبال خواهد داشت:

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
??? Error using ==> surface
Matrix dimensions must agree.

Error in ==> C:\MATLABR11\toolbox\matlab\graph3d\surf.m
On line 59  ==>     hh = surface(varargin{:});

Error in ==> C:\MATLABR11\work\main.m
On line 9  ==> surf(p,v,t)
```

خطای این دفعه باز هم مربوط به دستور surf و این بار در باره نحوه معرفی آرایه ها به آن است. بـا مراجعه به توضیحات (help) این دسـتور مشخص می گردد که آرگومانـهای اول و دوم ایـن دسـتور می توانند بردار باشند ولی آرگومان سـوم باید ماتریس باشد. در این حالت طول آرگومانـهای اول و

دوم باید به ترتیب برابر با تعداد ستونها و سطرهای آرگومان سوم باشد. لـذا طبـق ایـن توضیحـات متغیر v باید آرگومان سوم دستور surf باشد و ضمناً با مشاهده ابعاد این متغیر:

```
» size(v)
ans =
   10   21
```

می توانید بگوئید که آرگومان اول باید بردار t و آرگومان دوم بـاید بـردار p باشـد. بنـابراین برنامـه اصلی باید به شکل زیر اصلاح گردد:

```
% Input
p = input(' Pressure (bar) = ');
t = input(' Temperature (K) = ');

% Calculation
v = ideal(t,p*1e5);

% Plotting results
surf(t,p,v)
xlabel('T (K)')
ylabel('P (bar)')
zlabel('V (m^3/kmol)')
view(135,30)
```

در صورت اجرای برنامه نتیجه نهایی را خواهید دید.

```
» main
 Pressure (bar) = [1:10]
 Temperature (K) = 300:5:400
```

۲-۷ دستورهای echo و keyboard

بکار بردن دستور echo باعث می گردد که هر سطر از برنامه اصلی قبل از آنکه اجرا گردد روی صفحه نمایش نشان داده شود. بنابراین ترتیب اجرای دستورات مشخص می شود. این دستور بویژه هنگامی که در برنامه حلقه ها و دستورات شرطی متعدد وجود دارد می تواند مفید واقع شود. در صورتی که بخواهید این دستور در هنگام اجرای تابع خاصی بکار بیفتد باید نام تابع مورد نظر را بعد از echo بیاورید. به هر حال، این دستور در بسیاری از موارد کمک چندانی به پیدا کردن خطای برنامه نمی کند زیرا در بیشتر موارد MATLAB سطری که برنامه در آن متوقف شده است را مشخص می نماید.

در صورت استفاده از دستور keyboard در میان برنامه، اجرای برنامه هنگامی که به آن دستور می رسد موقتاً متوقف می گردد و به شما اجازه می دهد که عملیات مورد نظرتان را انجام دهید. در چنین حالتی علامت « K را روی صفحه نمایشگر مشاهده خواهید نمود. برنامه پس از آنکه دستور return را وارد نمودید از جایی که متوقف شده بود، ادامه می یابد. این دستور بویژه در مواقعی بکار می رود که برنامه بواسطه اندازه و یا مقدار یک متغیر پیغام خطا می دهد. شما با استفاده از دستور keyboard امکان می یابید که اندازه و یا مقدار متغیر مورد سئوال را دیده و یا آن را تغییر دهید و پس از استفاده از دستور return اثر این تغییر را در اجرای ادامه برنامه مشاهده نمائید

مشتق، حد، انتگرال و حل معادلات دیفرانسیل

محاسبه ی مشتق

فرم کلی دستور بدین صورت است:

```
>> syms x
>> f = تابع مورد نظر
>> diff(f, متغیر مشتق, مرتبه ی مشتق)
```

syms از واژه ی Symbol به معنای نمادین گرفته شده است و در خط اول، این دستور یک متغیر نمادین تعریف می کند. (و نه متغیر عددی)

مثال 1:

```
>> syms x

>> f = sin(x)

>> diff(f,x,1)


ans =
     cos(x)
```

محاسبه ی حد

```
>> syms x

>> f = تابع مورد نظر

>> limit(f,x,a)  ≡  lim f(x)   (x→a)
```

مثال 1

```
>> syms x
>> f = (1+1/x)^x
>> limit(f,x,inf)


ans =
     exp(1)
```

محاسبه ی انتگرال

```
>> syms x

>> f = تابع مورد نظر

>> int(f,x)  ≡  ∫f(x)dx
```

✓ برای محاسبه ی انتگرال معین از دستور زیر استفاده می کنیم:

```
>> int(f,x,a,b)  ≡  ∫_a^b f(x)dx
```

مثال 1:

```
>> syms x
>> f=sin(x)*x
>> int(f,x)

ans =
     sin(x)-x*cos(x)
```

✓ برای محاسبه ی مجموع یک دنباله از دستور روبرو استفاده می کنیم:

$\sum_{n=a}^{b} f(n)$ ≡ symsum(f,n,a,b)

✓ برای محاسبه ی سری تیلور از دستور روبرو استفاده می کنیم: taylor(f,b,a)

یاد آوری: سری تیلور بسط هر تابع را حول نقطه ی a بدست می دهد:

$$\sum_{n=0}^{b}(x-a)^n \frac{f^n(a)}{n!}$$

در دستور بالا اگر a را بکار نبریم سری مک لوران محاسبه می شود.

با اجرای دستور taylortool << یک محیط گرافیکی برای محاسبه ی سری تیلور فراخوانی می شود.

حل معادلات دیفرانسیل

```
>> dsolve('Dnامy = تابع بر حسب متغیر ها' , 'شرایط اولیه' )
```

مثال 1: حل معادله دیفرانسیل $\frac{dy}{dx}=1+y^2$ با شرایط اولیه ی $y(0)=1$:

```
>> dsolve('Dy=1+y^2','y(0)=1')
ans =
     tan(t+1/4*pi)
```

مثال 2: حل معادله دیفرانسیل $\frac{d^2y}{d^2x} = \cos(2x) - y$ با شرایط اولیه ی $y(0) = 1$ و $\frac{dy(0)}{dx} = 0$:

```
>> dsolve('D2y = cos(2*x)-y','y(0)=1','Dy(0)=0','x')
   ans =
       (1/2*sin(x)+1/6*sin(3*x))*sin(x)+(1/6*cos(3*x)-
       1/2*cos(x))*cos(x)+4/3*cos(x)

   >> simplify(ans)

   ans =
       -2/3*cos(x)^2+4/3*cos(x)+1/3
```

- چون مشتق بر حسب x است آن را به عنوان آخرین آرگومان در نظر می گیریم.
- دستور Simplify برای ساده کردن عبارت حاصل به کار می رود.

تبدیل لاپلاس و معکوس آن

برای محاسبه ی لاپلاس یک عبارت از دستور laplace استفاده می کنیم.

مثال 1.

$f(t) = \sin(t)$

```
>> syms t
>> f = sin(t)
>> laplace(f)

ans =
     1/(s^2+1)
```

- برای ساده کردن پاسخ نهایی از دستور simplify استفاده می کنیم.
- توابع پله واحد و ضربه واحد به ترتیب با Heaviside و Dirac معرفی می شوند.

بسط به کسرهای جزئی

برای پارامترهای r، p و y در رابطه ی زیر از دستور residue استفاده می شود:

در صورت عدم وجود ریشه ی تکراری در مخرج: $$\frac{N(s)}{D(s)} = \frac{r_1}{x-p_1} + \frac{r_2}{x-p_2} + \ldots + \frac{r_n}{x-p_n} + y(s)$$

در صورت وجود ریشه ی تکراری در مخرج: $$\frac{N(s)}{D(s)} = \frac{r_1}{(x-p)} + \frac{r_2}{(x-p)^2} + \ldots + \frac{r_n}{(x-p)^n} + y(s)$$

مثال 1.

عبارت $V(s) = \frac{2}{s^3 + 12s + 36s}$ را به کسرهای جزئی بسط دهید.

$$\frac{2}{s^3 + 12s + 36s} = \frac{-0.0556}{(s+6)} + \frac{-0.3333}{(s+6)^2} + \frac{0.0556}{s}$$

```
>> syms s
>> N = [2]
>> D = [1 12 36 0]
>> [r p y] = residue(N,D)

r =              p =              y =

   -0.0556          -6               []
   -0.3333          -6
    0.0556           0
```

حل دستگاه های جبری با متغیر نمادین

دستگاه زیر را که از حل یک مدار در حوزه ی لاپلاس بدست آمده است در نظر بگیرید:

$$\begin{cases} (3s+10)I_1 - 10I_2 = \dfrac{4}{s+2} \\ -10I_1 + (4s+10)I_2 = \dfrac{-2}{s+1} \end{cases}$$

بدلیل اینکه متغیر ها پارامتری می باشند نمی توانیم از روش ارائه شده در بخش سوم برای حل این دستگاه استفاده کنیم و باید دو متغیر رشته ای تعریف کنیم:

```
>> eq1 = '(3*s+10)*I1-10*I2=4/(s+2)'
>> eq2 = '-10*I1+(4*s+10)*I2=-2/(s+1)'
```

برای حل دستگاه تعریف شده از دستور solve استفاده می کنیم:

```
>> solution = solve(eq1,eq2,'I1','I2')
```

مطلب حاصل را در یک structure ذخیره می کند که برای دسترسی به این ساختار از روشی شبیه به گرامر زبان C استفاده می کنیم:

```
>> I1 = solution.I1
>> I2 = solution.I2
```

و در نهایت لاپلاس معکوس I1 و I2 را می یابیم:

```
>> i1 = ilaplace(I1)
   i1 =

    10/29*exp(-t)-172/667*exp(-35/6*t)-2/23*exp(-2*t)

    >> i2 = ilaplace(I2)
       i2 =

    129/667*exp(-35/6*t)-10/23*exp(-2*t)+7/29*exp(-t)
```

تبدیل فوریه و تبدیل Z

تبدیل فوریه یکی از تبدیلات مهم در ریاضیات و پردازش سیگنال می باشد و حالت گسسته ی آن تبدیل Z می باشد. با استفاده از دستورات Matlab به راحتی می توان این تبدیلات را انجام داد.

تبدیل فوریه ی یک تابع با استفاده از دستور Fourier محاسبه می شود.

مثال 1.

```
>> syms x
>> f = sin(x)
>> fourier(f)

ans =
    -i*pi*Dirac(w-1)+i*pi*Dirac(w+1)
```

برای محاسبه ی عکس فوریه از ifourier استفاده می کنیم.

تبدیل z یک دنباله که با سری $z[f(n)]=\sum_{n=0}^{\infty} f(n)z^{-n}$ تعریف می شود با دستور ztrans قابل محاسبه است.

```
>> syms n
>> f = 3^(-n)
>> ztrans(f)

ans =

  3*z/(3*z-1)
```

simulink

بخش اول

شبیه‌سازی یک مدار ساده

قطعات سیستم قدرت به شما اجازه ساخت و شبیه‌سازی مدارات الکتریکی که شامل المان‌هـای خطی و غیرخطی هستند را می‌دهد. از بخش ۱ تا بخش ۳ مدار زیر ساخته و شبیه‌سازی می‌شود

.

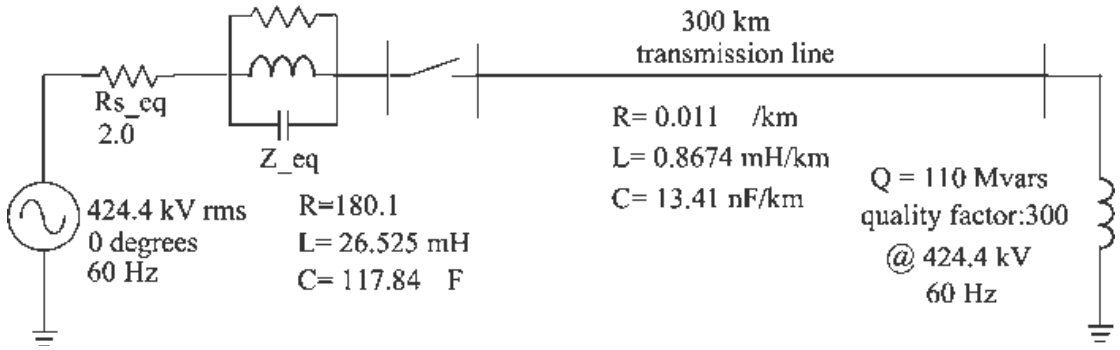
در این بخش با :

ـ کتابخانه قطعات سیستم قدرت، **powerlib** را بررسی خواهیدکرد.

ـ یاد خواهید گرفت که چگونه یک مدار ساده از کتابخانه **powerlib** بسازید.

مثال :

مدار زیر یک سیستم قدرت هم توان را که یک خط انتقال ۳۰۰ کیلومتری را تغذیه می‌کند نشان می‌دهد. خط در طرف تغذیه با القاگر شنت جبران شده‌است. یک **Breaker** اجازه باردار و بی‌بار کردن خط را می‌دهد. برای ساده کردن بحث فقط یکی از سه فاز نمایش داده شده است. پارامترهای نشان داده شده در شکل مقادیر مربوط به سیستم قدرت **powerlib** می‌باشند.



شکل ۱ـ۱: مداری که با قطعات سیستم قدرت باید مدل شود.

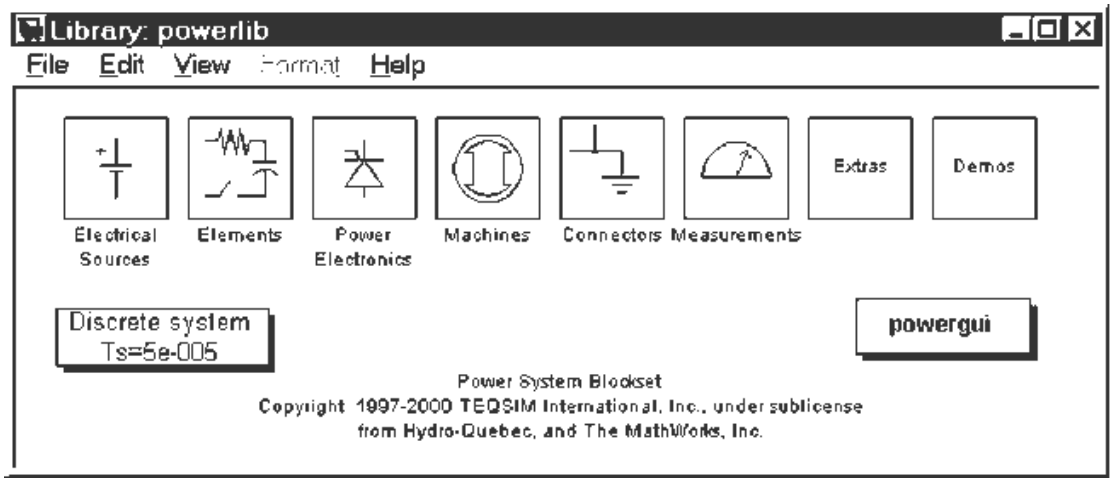
ساختن مدار با کتابخانه powerlib

استفاده از رابط گرافیکی، اتصال قطعات **Simulink** به قطعات قدرت را امکان‌پذیر می‌نماید. قطعات در کتابخانه ویژه‌ای به اسم **powerlib** گردآوری شده‌اند.

با وارد کردن دستور زیر در محیط **MATLAB** ، کتابخانه قطعات سیستم قدرت را باز نمایید:

powerlib

این فرمان یک پنجره **Simulink** را نمایش می‌دهد که آیکون کتابخانه‌های مختلف را نشان می‌دهد.



شما می‌توانید این کتابخانه‌ها را بازنمایید تا قطعاتی که به مدارتان کپی خواهد شد را ببینید. هر قطعه با آیکونی که دارای ورودی و خروجی‌های منطبق بر ترمینالهای قطعه می‌باشد، نمایش داده می‌شود.

۱ـ در پنجره **powerlib** از منوی **File**، یک پنجره جدید که اولین مدار شما خواهد بود باز نمایید. آنرا با نام **circuit1** ذخیره نمایید.

۲ـ کتابخانه منابع تغذیه (**Electerical Sources**) را باز نموده و یک منبع ولتاژ **AC** (**AC_Voltage_Source**) به پنجره **circuit1** کپی کنید.

۳ـ با دوبار کلیک کردن روی منبع ولتاژ **AC**، پنجره تنظیمات آنرا باز نموده و پارامترهای دامنه (**Peak amplitude (V)**:)، فاز (**Phase (deg)**:) و فرکانس (**Frequency (Hz)**:) را مانند مقادیر نشان داده شده در شکل ۱ـ۱ وارد نمایید.

۴ـ اسم منبع ولتاژ را به **Vs** تغییر دهید.

۵ـ قطعه شاخه موازی **RLC** (**Parallel RLC Branch**) را که می‌توانید آنرا در کتابخانه عناصر (**Elements**) بیابید به مدار خود کپی نموده و پارامترهای آنرا هماگونه که در شکل ۱ـ۱ نشان داده شده است تغییر دهید و اسم آنرا **Z eq** بگذارید.

۶ـ مقاومت **Rs eq** می‌تواند از شاخه موازی **RLC** درست شود. قطعه شاخه موازی **RLC** که در حال حاضر از مدارتان وجود دارد را دوباره ایجاد نمایید و پارامتر مقاومت (**Resistan R (ohms)**:) را طبق شکل ۱ـ۱ تغییر دهید و پارامترهای اندوکتانس (**Inductance L (H)**:) و کاپاسیتانس (**Capasitance C(F):**) را به ترتیب به بینهایت (**inf**) و صفر (۰) تغییر دهید.

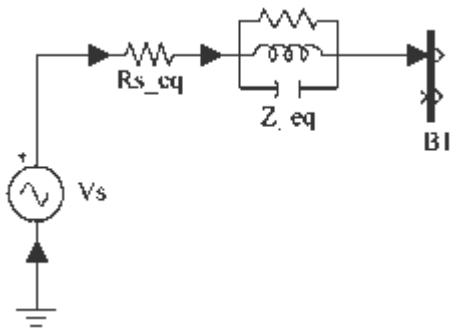
وقتی پنجره تنظیمات را می‌بندید، متوجه می‌شوید که قطعات **L** و **C** نشان داده نمی‌شوند و آیکون در حال حاضر یک مقاومت ساده را نمایش می‌دهد. نتیجه مشابه‌ای می‌تواند با شاخه سری **RLC** **(Series RLC Branch)** رخ می‌دهد. با تغییر **L** و **C** به ترتیب به صفر و بینهایت.

۷ـ این قطعه را **Rs eq** بنامید.

۸ـ کتابخانه اتصالات (**Connectors**) از **powerlib** را باز نمایید و یک باس‌بار (**Bus Bar**) کپی کنید.

۹ـ پنجره تنظیمات باس‌بار را باز نمایید و پارامترهای آنرا به دو ورودی (**Number of input**) و دو خروجی (**Number of output**) تغییر دهید و نام آنرا **B1** قراردهید. همچنین یک قطعه زمین (**Ground**) کپی کنید (قطعه زمین را با یک خروجی را انتخاب نمایید)

اجزا مختلف را جا به جا نمایید و قطعات را با کشیدن خطوط از خروجی‌ها به ورودی‌های قطعات به هم وصل نمایید.



برای تکمیل مدار مطابق شکل ۱ـ۱ شما نیاز به اضافه کردن یک خط انتقال و یک راکتور شنت دارید. **Breaker** را بعداً در بخش ۳ اضافه خواهید کرد. مدل یک خط با توزیع یکنواخت پارامترهای **R**، **L** و **C** در طول خط شامل یک تاخیر زمانی است که مساوی با زمان انتشار موج در طول خط است. این مدل نمی‌تواند به عنوان یک سیستم خطی شبیه‌سازی گردد. زیرا یک تاخیر، مشابه یک سری حالت سری بی‌کران می‌باشد. ولیکن یک تقریب خوب از یک خط با یک سری از حالات پایدار می‌تواند با سری‌کردن چندین مدار **PI** که هرکدام به عنوان بخش کوچکی از خط ظاهر می‌شود قابل دست‌یابی است.

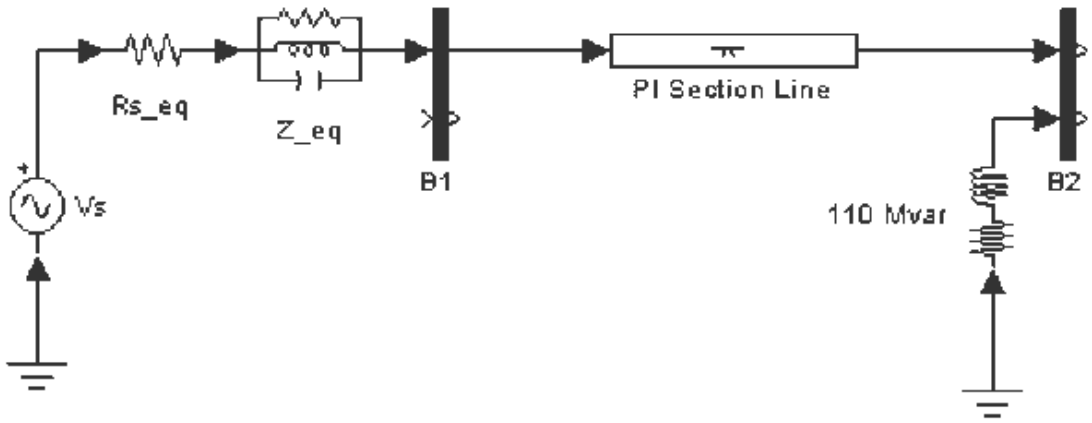
یک بخش **PI** شامل یک شاخه سری **RL** و دو شاخه شنت **PI** می‌باشد. دقت مدل بستگی به تعداد بخش‌های **PI** دارد که برای مدل استفاده شده‌است. قطعه خط **PI (PI Section Line)** را از کتابخانه عناصر (**Elements**) به پنجره **PI** کپی نمایید و پارامترهای آنرا مانند آنچه در شکل ۱ـ۱ نشان داده شده‌است تغییر دهید و تعداد بخش‌های **PI** خط (**Number of pi Section**) را برابر ۱ قراردهید.

راکتور شنت با یک مقاومت و القاگر سری مدل شده است. می‌توانید از شاخه سری **RLC (Series_RLC_Branch)** برای مدل کردن راکتور شنت استفاده نمایید. مقدار **R** و **L** را بر طبق توان اکتیو و راکتیو که در شکل ۱ـ۱ نشان داده شده‌است تغییر دهید. (**P=110/300=0.37MW** و **Q=110Mvars** در **V=4242.4kV(rms)** و **f=60Hz**)

شما شاید راحت‌تر باشید که از قطعه بار سری **RLC (Series RLC Load)** استفاده نمایید که به شما اجازه می‌دهد مستقیماً توان اکتیو و راکتیو جذب شده توسط راکتور شنت را تعیین نمایید. قطعه بار سری **RLC** را که می‌توانید آنرا در کتابخانه عناصر (**Elements**) از **powerlib** بیابید را کپی کنید. اسم آنرا **Mvars۱۱۰** قرار دهید و پارامترهای آنرا مانند زیر قرار دهید:

Nominal votage Vn (Vrms) : 424.4e3
Nominal frequency fn (Hz) : 60
Avtive power P(W) : 110e6/300
Inductive reactive power QL (positive var) : 110e6
Capasitive reactive power QC (negative var) : 0

توجه کنید که وقتی توان راکتیو خازنی تعیین‌نشده، با بسته شدن پنجره تنظیمات، خازن در آیکون قطعه نمایش داده نمی‌شود. یک باس پایانی گیرنده **PI** با کپی کردن **PI** اضافه نمایید و تمام قطعات جدید را همانطور که نشان داده شده به هم وصل نمایید.

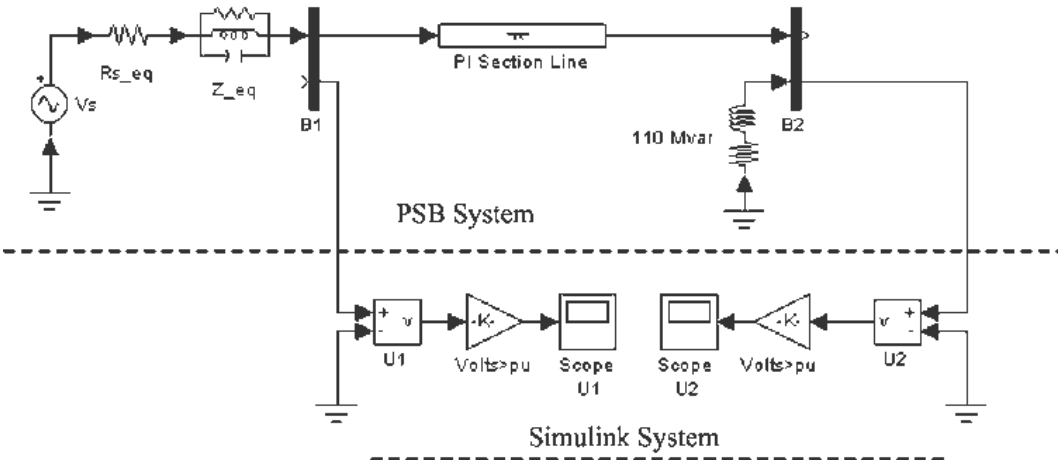


شما یک قطعه اندازه‌گیر ولتاژ (**Voltage Measurement**) برای اندازه‌گرفتن ولتاژ در باس **B1** نیاز دارید. این قطعه در کتابخانه اندازه‌گیرها (**Measurement**) از **powerlib** یافت می‌شود. آنرا کپی کنید و آنرا **U1** بنامید. ورودی مثبت آنرا به خروجی دوم باس‌بار **B1** و ورودی منفی را به یک زمین جدید وصل نمایید.

برای مشاهده ولتاژ اندازه گرفته شده توسط قطعه اندازه‌گیر ولتاژ که **U1** نامیده شده؛ یک سیستم نمایش مورد نیاز است. این سیستم می‌تواند هر وسیله‌ای که در کتابخانه **Sinks** از **Simulink** یافت می‌شود باشد.

کتابخانه **Sinks** از **Simulink** را باز نمایید و قطعه اسکوپ **Scope** را به پنجره مدارتان **circuit1** کپی نمایید. اگر اسکوپ؛ مستقیماً به خروجی اندازه‌گیر ولتاژ وصل شود، ولتاژ را به ولت نمایش می‌دهد. ولیکن مهندسان قدرت در سیستم‌های قدرت با مقادیر نرمال‌شده (سیستم پریونیت) کار می‌کنند. ولتاژ با تقسیم به مقدار پیک ولتاژ نامی سیستم به عنوان مقدار مبنا نرمالیزه می‌شود.

قطعه گین (**Gain**) را از کتابخانه **Simulink** انتخاب کنید و گین (**Gain**) آنرا مطابق بالا واردنمایید. خروجی آنرا به قطعه اسکوپ و خروجی قطعه اندازه‌گیر ولتاژ را به قطعه گین اتصال دهید این سیستم اندازه‌گیر ولتاژ را کپی کنید و در باس‌بار **B2** قراردهید. همانند شکل زیر:

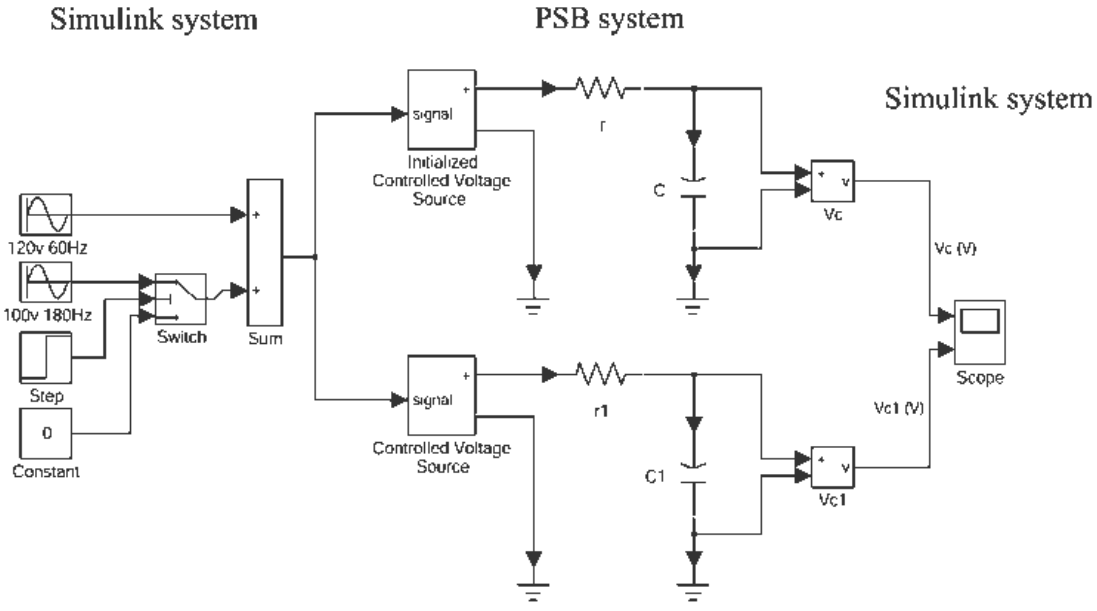


هماهنگ کردن مدار با Simulink

قطعه اندازه‌گیر ولتاژ به عنوان یک هماهنگ کننده بین قطعات **Simulink** عمل می‌نماید. برای سیستم نشان داده شده در بالا؛ ارتباط بین سیستم قدرت و سیستم **Simulink** انجام شده‌است. قطعات اندازه‌گیر ولتاژ، ولتاژهای اندازه‌گیری شده را به سیگنال **Simulink** تبدیل می‌نماید.

توجه کنید که قطعه اندازه‌گیر جریان (**Current Measurement**) از کتابخانه اندازه‌گیرهای **powerlib** می‌تواند برای تبدیل جیران‌های اندازه‌گیری شده به سیگنال **Simulink** مورد استفاده قرارگیرد.

اتصال قطعات **Simulink** به سیستم قدرت نیز امکان‌پذیر می‌باشد. به عنوان مثال، می‌توانید یک قطعه منبع ولتاژ کنترل‌شده (**Controlled Voltage Source**) برای تزریق ولتاژ در یک مدار الکتریکی استفاده نمایید. آنگاه ولتاژ با سیگنال **Simulink** کنترل می‌شود.



شبیه‌سازی مدار :

حال شما می‌توانید شبیه‌سازی را از منوی **Simulation** شروع کنید. طبق انتظار ولتاژ سینوسی و با مقدار پیک ۱**p.u**. می‌باشد.

وقتی شبیه‌سازی در حال اجرا می‌باشد؛ پنجره تنظیمات قطعه **Vs** را باز نمایید و دامنه را تغییر دهید. اثر آنرا روی دو اسکوپ مشاهده نمایید. همچنین می‌توانید فرکانس و فاز آنرا تغییر دهید. به یاد داشته‌باشید که می‌توانید بر روی قسمت مورد علاقه‌تان با کشیدن مسیر بسته روی آن (از کلید چپ موس و کشیدن آن روی ناحیه مورد نظر استفاده کنید) بزرگنمایی داشته باشید.

تذکر: برای شبیه‌سازی این مدار الگوریتم پیش‌فرض (**ode45**) استفاده شده‌است. ولی به سبب نیاز بیشتر قطعات سیستم قدرت مدارات شما شامل کلیدها و سایر مدل‌های غیرخطی خواهدبود. در چنین مواردی شما باید الگوریتم کامل سازی متفاوتی تعیین نمایید. این موضوع در بخش ۳ قسمت شبیه‌سازی حالت گذرا، هنگامیکه یک **Breaker** به مدارتان وصل می‌شود مورد بحث قرار می‌گیرد

بخش دوم

تحلیل یک مدار ساده

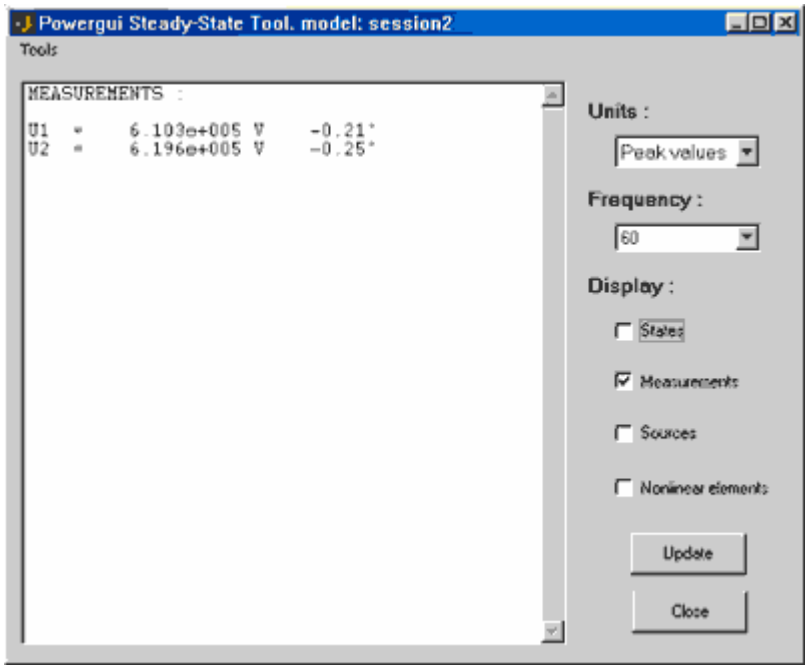
در این بخش با :

- از قطعه **powergui** (نمایشگر گرافیکی) استفاده خواهید نمود.
- خروجی‌های حالت پایدار سیستم را بدست می‌آورید.
- مدارتان را با تابع **power2sys** تحلیل می‌نمایید.
- یک مدار الکتریکی را در ناحیه فرکانسی تحلیل می‌نمایید.

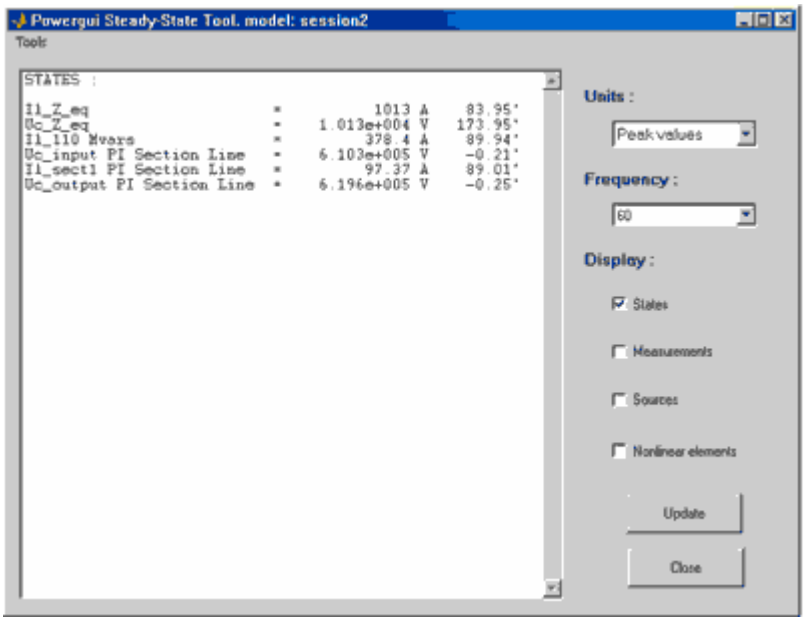
تحلیل حالت پایدار

برای آسان کردن تحلیل حالت پایدار، یک نمایشگر گرافیکی **(powergui)** در کتابخانه **powerlib** در نظر گرفته شده است. قطعه نمایشگر گرافیکی **(powergui)** را به پنجره **circuit1** کپی نمایید و روی آیکون آن دو بار کلیک کنید تا صفحه آن باز شود.

پنجره حالت پایدار را با کلیک روی دکمه ولتاژها و جریان‌های حالت پایدار **(Steady State Voltages and Currents)** باز نمایید که در آن فازورهای اندازه گرفته شده توسط دو قطعه اندازه‌گیری به فرم قطبی نشان داده شده‌است.



خروجی هر اندازه‌گیر با تطبیق آن با اسم قطعه اندازه‌گیر مشخص می‌شود. اندازه فازورهای **U1** و **U2** با مقدار پیک ولتاژ سینوسی تطبیق یافته‌اند. در پنجره حالت پایدار، همچنین می‌توانید مقدار حالت پایدار منبع ولتاژ یا مقدار حالت پایدار شرایط را با علامت زدن در جعبه علامت کنار هر یک از منابع **(Sources)** یا شرایط **(States)** انتخاب نمایید تا نمایش داده شوند.



اسم متغییرهای حالت؛ شامل اسم قطعه‌ای می‌باشد که در آن القاگر یا خازن وجود دارد. معرفی شده‌ها با پیشوند **(Il_)** برای جریان القاگر و یا معرفی شده‌ها با پیشوند **(Uc_)** برای ولتاژ خازن می‌باشند.

قرارداد علامت منابع ولتاژ و جریان و متغییرهای حالت بر مبنای وضعیت آنها در مدار می‌باشد.

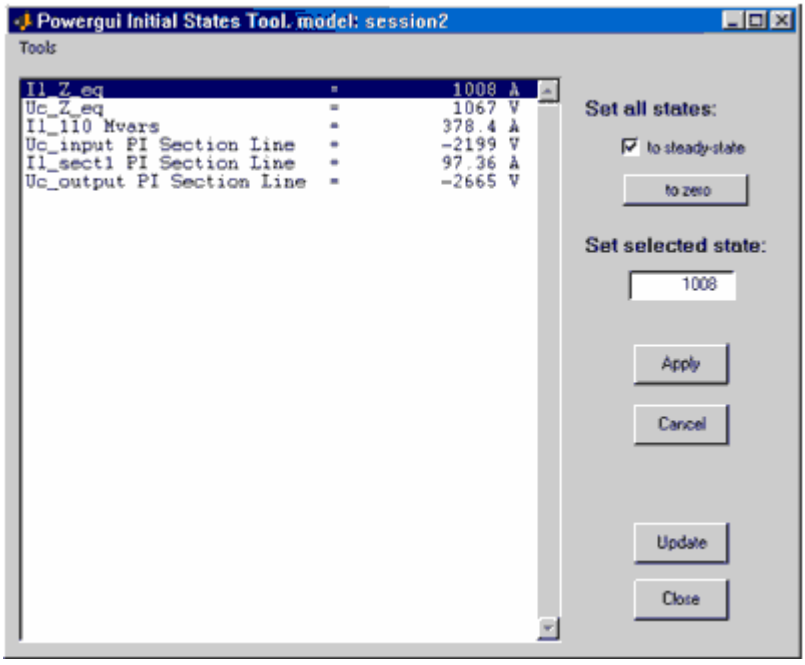
ـ جریان القاگرها اگر در جهت فلش جاری باشد مثبت است.

ـ ولتاژ خازن‌ها عبارت است از ولتاژ خروجی قطعه منهای ولتاژ ورودی قطعه

تذکر :

بر حسب موقعیت دقیق چندین قطعه در دیاگرام **circuit1** ممکن است مقادیر حالت پایدار قطعات همانند آنچه که در مدار دیده می‌شود نمایش داده نشود.

حالا از جعبه ابزار نمایشگر گرافیکی **(powergui)** روی دکمه تنظیم مقادیر اولیه **(Initial States Setting)** کلیک نمایید. مقادیر اولیه شش شرط مدار (جریان سه القاگر و ولتاژ سه خازن) نشان داده می‌شوند. این مقادیر اولیه برای شروع شبیه‌سازی حالت پایدار تنظیم شده‌اند.



تحلیل فرکانسی

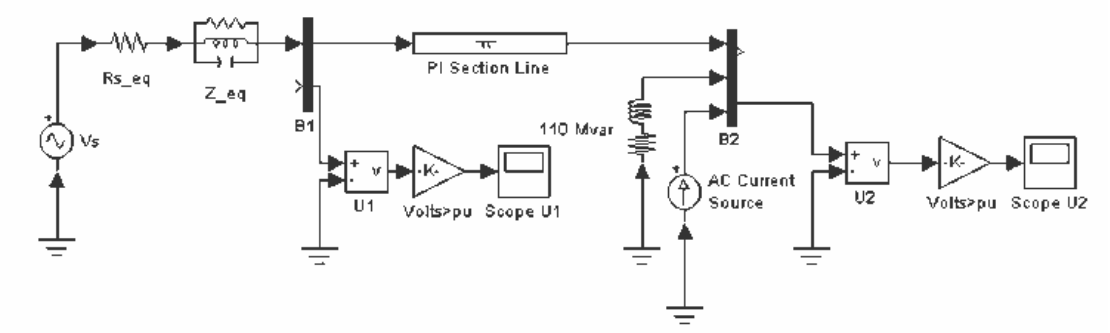
کتابخانه اندازه‌گیرها **(Measurments)** از **powerlib** دارای قطعه اندازه‌گیر امپدانس **(Impedance Measurment)** می‌باشد که امپدانس بین دو گره از یک مدار را اندازه می‌گیرد. در دو بخش آتی شما امپدانس بین **B2** و زمین مدارتان را به دو روش اندازه می‌گیرید.

ـ محاسبه از مدل فضای حالت

ـ اندازه‌گیری اتوماتیک یه کمک اندازه‌گیر امپدانس دو قطعه نمایشگر گرافیکی **(powergui)**

بدست آوردن امپدانس متقابل فرکانسی از مدل فضای حالت

برای اندازه‌گیری امپدانس متقابل در باس‌بار **B2**، به یک منبع جریان در باس‌بار **B2** نیازمندید تا یک ورودی دوم برای مدل فضای حالت ایجاد نمایید. کتابخانه منابع الکتریکی **(Electrical Sources)** را باز نمایید و قطعه منبع جریان **AC** را به مدار خود کپی نمایید. این منبع را به باس‌بار **B2** همانند شکل زیر اتصال دهید و اندازه ماکزیمم منبع جریان **(Peak amplitude (A):)** را به صفر و فرکانس **(Frequency (Hz):)** آنرا به **60Hz** تغییر دهید. قطعات را مانند زیر مرتب نمایید:



شکل ۱ـ۲: منبع جریان **AC** در باس‌بار **B2** (مدار **cicuit1-1**)

حالا مقادیر فضای حالت مدار **circuit1-1** را با تابع **power2sys** محاسبه نمایید. دستور زیر را در خط فرمان **MATLAB** وارد نمایید:

[A,B,C,D,x0,states,inputs,outputs]=power2sys('circuit1-1');

تابع **power2sys** مدل فضای حالت مدارتان را در چهار ماتریس **A**، **B**، **C** و **D** ارایه می‌کند. **x0** بردار شرایط اولیه‌ای است که شما در قطعه نمایشگر گرافیکی **(powergui)** نیز دیده‌اید. اسم متغییرهای حالت ، ورودی‌ها و خروجی‌ها در سه ماتریس ستونی داده می‌شود.

States=	**شرایط**
Il_110 Mvars	
Uc_input PI Section Line	
Il_sect1 PI Section Line	
Uc_output PI Section Line	
Il_Z_eq	
Uc_Z_eq	
Inputs=	**ورودی‌ها**
U_Vs	
I_AC Curent Source	
Outputs=	**خروجی‌ها**
U_U1	
U_U2	

توجه کنید که باید اسم و ترتیب متغییرها، ورودی‌ها و خروجی‌ها را از قطعه **Powergui** استخراج نمایید.

همینکه مدل فضای حالت سیستم شناخته شده باشد، می‌تواند در حوزه فرکانسی تحلیل گردد. به عنوان مثال مدهای این مدار می‌تواند از مقادیر آیگن ماتریس **(A)** به دست بیاید (فرمان **eig** از **MATLAB** را استفاده نمایید.)

eig(A)
ans
1.0e+05*
-2.4972
-0.0001+0.0144i ←229Hz
-0.0001-0.0144i
-0.0002+0.0056i ←89Hz
-0.0002-0.0056i
-0.0000

این سیستم دو مد نوسانی **89Hz** و **299Hz** دارد. فرکانس **89Hz** مربوط به منبع معادل می‌باشد که با تک قطبی معادل، مدل شده است. **229Hz** اولین مد خط است که با یک بخش **PI** مدل شده است.

اگر شما جعبه ابزار کنترل سیستم را داشته باشید، می‌توانید امپدانس شبکه را با تابع **bode** به عنوان یک تابع فرکانسی محاسبه نمایید. در فضای لاپلاس، امپدانس ***Z2***، در باس **B2** به صورت تابعی بین جریان تزریق‌شده به باس **B2** (ورودی دوم سیستم) و ولتاژ اندازه‌گرفته شده در باس **B2** (خروجی دوم سیستم) [انتقال یافته به محیط لاپلاس] تعریف می‌شود.

اگر شما جعبه ابزار کنترل سیستم را داشته باشید، می‌توانید امپدانس شبکه را با تابع **bode** به عنوان یک تابع فرکانسی محاسبه نمایید. در فضای لاپلاس، امپدانس ***Z2***، در باس **B2** به صورت تابعی بین جریان تزریق‌شده به باس **B2** (ورودی دوم سیستم) و ولتاژ اندازه‌گرفته شده در باس **B2** (خروجی دوم سیستم) [انتقال یافته به محیط لاپلاس] تعریف می‌شود.

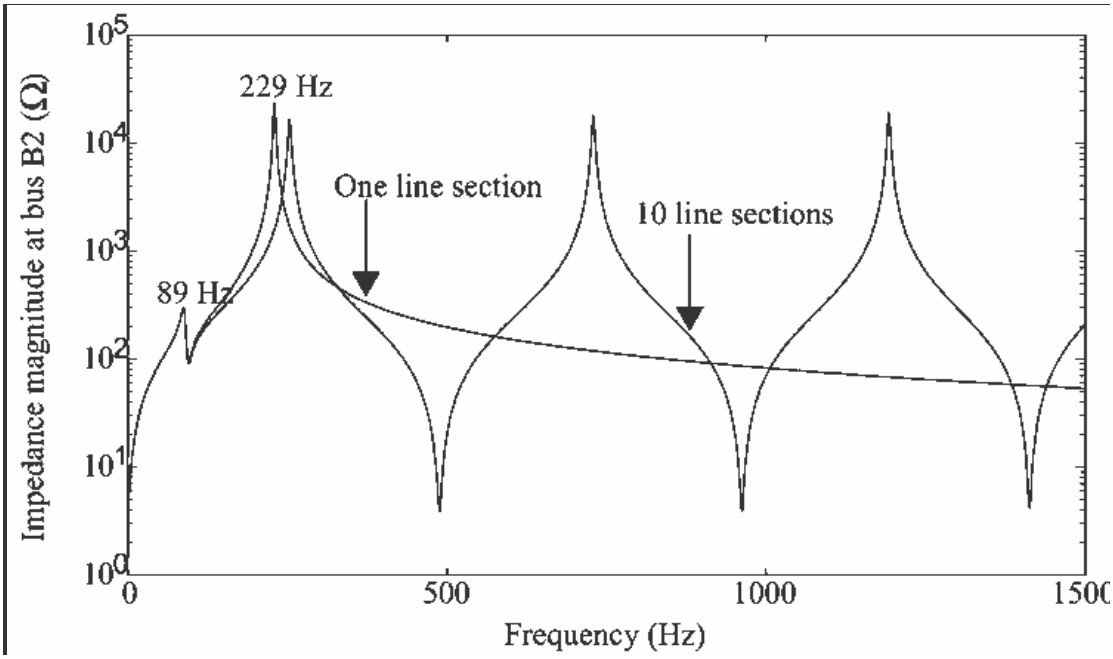
امپدانس در باس **B2** در بازه **0** تا **1500** هرتز می‌تواند به صورت زیر محاسبه شود:

```
Freq=0:1500;
W=2*pi*freq;
[mag1,phase1]=bode(A,B,C,D,2,w);
semilogy (freq,mag1 (:,2));
```

عملیات مشابهی را برای بدست‌آوردن پاسخ مدل خط با ۱۰ بخش تکرار کنید. جعبه محاوره‌ای خط مدل **PI** را باز کنید و تعداد بخشها را از ۱ به ۱۰ تغییر دهید. برای محاسبه پاسخ فرکانسی جدید و انداختن آن روی پاسخ بدست‌آمده از مدل تک بخشی، دستورات زیر را بنویسید:

```
  [A,B,C,D]=power2sys('circuit1-1');
[mag10,phase10]=bode(A,B,C,D,2,w);
semiloy(freq,mag1(:,2),freq,mag10(:,2));
```

این نتیجه رسم می‌باشد:



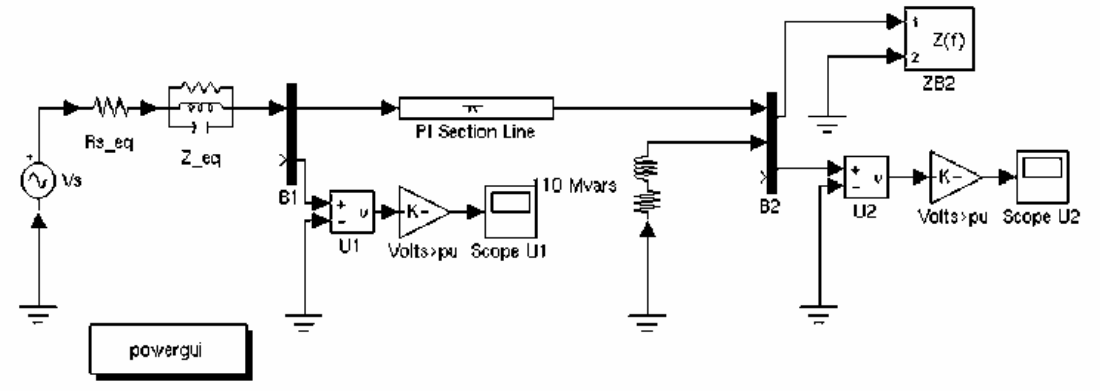
شکل ۱ـ۳: امپدانس باس **B2** تابع فرکانسی

این شکل نشان می‌دهد، ناحیه فرکانسی که توسط یک خط با یک بخش ظاهر می‌شود تقریباً به ۱۵۰ هرتز محدود شده است. برای فرکانس‌های بالاتر یک خط با ۱۰ بخش تقریب بهتری می‌باشد.

بنابراین زمان انتشار برای **300km** برابر است با: **T=300/293.208=1.023 s** و فرکانس اولین مدل خط برابر است با **f1=1/4T=244 Hz**. یک خط تکه تکه شده باید تعدادی نامتناهی از مدها را هر **224+n*488 Hz (n=1,2,3,...)** داشته باشد. خط ۱۰ بخشی؛ ۱۰ مد اول را شبیه‌سازی می‌کند. سه مد اول خط در شکل ۱ـ۳ دیده می‌شود.

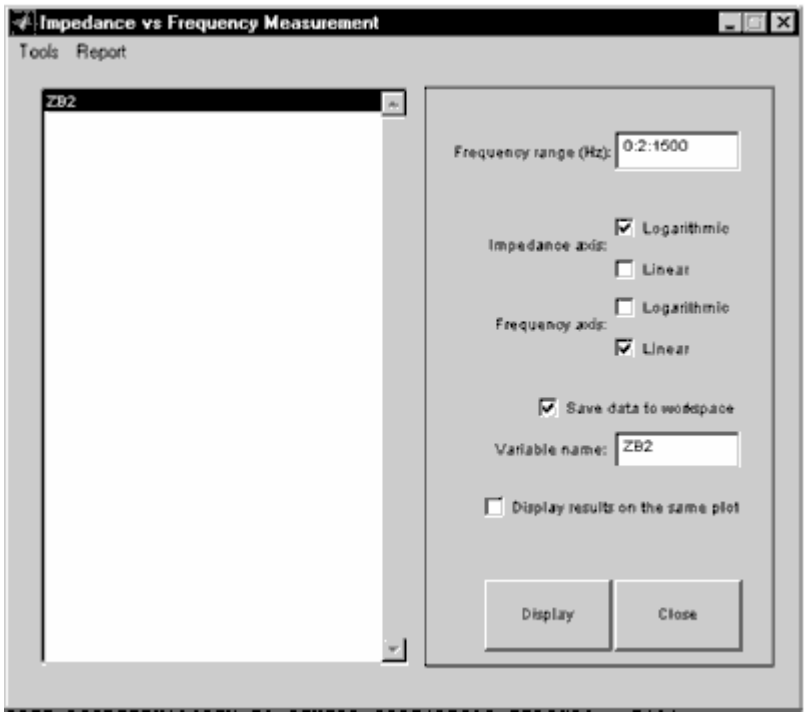
بدست‌آوردن امپدانس فرکانسی از قطعه اندازه‌گیر امپدانس (Measurement_Impedance) و قطعه نمایشگر گرافیکی (Powergui)

عملیاتی که در بالا برای اندازه‌گیری امپدانس یک مدار توضیح داده شد در قطعات سیستم قدرت اتوماتیک شده است. کتابخانه اندازه‌گیرها **(Measurements)** از **powerlib** را باز کرده و قطعه اندازه‌گیر امپدانس **(Impedance_Measurement)** را به مدل خود کپی نمایید و اسم آنرا به **ZB2** تغییر دهید. قطعه ؛ یک منبع جریان و یک اندازه‌گیر ولتاژ را برای اندازه‌گیری امپدانس به کار می‌برد. دو ورودی این قطعه را بین باس‌بار **B2** و زمین همانگونه که نشان داده شده‌است، وصل نمایید.



شکل ۱ـ۴: اندازه‌گیری امپدانس فرکانسی با قطعه اندازه‌گیر امپدانس

حالا پنجره **powergui** را باز کنید و در منوی آن **Impedance_vs_Frequency_Measurement** را انتخاب نمایید. پنجره جدیدی باز می شود که لیستی از اندازه‌گیرهای امپدانسی که در مدارتان وجود دارد را نشان می‌دهد.



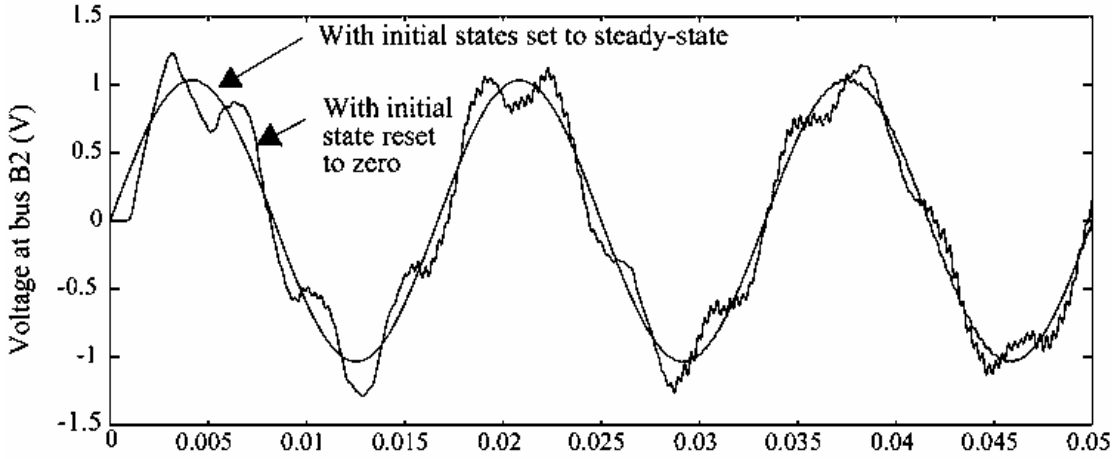
در این مورد، فقط یک امپدانس اندازه گرفته شده‌است و به عنوان **ZB2** در پنجره شناخته شده‌است (اسم قطعه **ZB2**). در محدوده فرکانس، مقدار **0:2:1500** (که بین صفر تا **1500** هرتز با پله‌های **2Hz** می‌باشد) را وارد نمایید. تقسیم‌بندی لگاریتمی **(Logaritmic_Impedance)** را برای نمایش اندازه **Z** انتخاب کنید. گزینه **Save_data_to_workspace** را انتخاب نمایید و **ZB2** را در جای اسم متغییر **(Variable_name)** که شامل امپدانس فرکانسی **vs** خواهدشد، تایپ نمایید.

وقتی که محاسبات تمام شد، پنجره نمودار اندازه و فاز به عنوان تابع فرکانسی نمایش داده می‌شود. نمودار اندازه باید مانند شکل ۱ـ۳ (برای خط یک بخشی) باشد. اگر به محیط کار خود **(workspace)** نگاه کنید باید یک متغییر با اسم **ZB2** داشته باشید. **ZB2** یک ماتریس دو ستونی می‌باشد که ستون اول شامل فرکانس می‌باشد و ستون دوم شامل امپدانس مختلط می‌باشد.

حالا از منوی **Simulation** گزینه **Simulation parameters** را انتخاب نموده و در قسمت حلال **(Solver)**، الگوریتم **ode23tb** را انتخاب نمایید. تغییرات وابسته **(Relative tolerance)** را به **1e-4** تغییر داده و سایر متغییرها را در حالت **auto** نگه دارید. زمان پایان **(Stop time)** را به **0.05** تغییر دهید. اسکوپ‌ها را باز نمایید و شبیه‌سازی را آغاز نمایید.

به شکل موج ولتاژ در ابتدا و انتها روی اسکوپ **U1** و **U2** نگاه کنید. از آنجا که متغییرهای حالت به صورت خودکار پردازش شده‌اند، سیستم در حالت پایدار شروع شده و شکل موج‌های سینوسی نمایش داده شده‌اند.

سرانجام پنجره **powergui** را باز کرده و تنظیم مقادیر اولیه **(Initial States Setting)** را باز نموده و تمام مقادیر را با کلیک روی دکمه **Set all states to zero** به صفر تغییر دهید و دکمه **Apply** را کلیک نمایید. شبیه‌سازی را تکرار کنید و زمان گذرا را از صفر که خط باردار می‌شود، مشاهده نمایید.



شکل ۱ـ۵: ولتاژ انتها **(U2)** با خط شامل ۱۰ بخش **PI**