به نام خدا

MATLAB

قابل استفاده برای

کلیه دانشجویان مهندسی و علوم پایه

مدرس:

هوشمند عزیزی

دانشگاه فنی و حرفه ای کرمانشاه

زمستان۹۳

فرمت نمایش اعداد :

با توجه به دقت و تعداد ارقام اعشاری قابل قبول در محاسبات می توان فرمت نمایش اعداد را در محیط matlab انتخاب کرد.با فرض اینکه $\sqrt{2}$ وارد شود نمایش این عدد به صورت زیر می باشد.

دستور		توضیحات
Format long	1.41421356237310	۱۶ رقم
Format long e	1.414213562373095e+000	۱۶ رقم به اضافه توان
Format short	1.4142	نمایش پیش فرض-۴ رقم اعشار
Format short e	1.4142e+000	۵ رقم به اضافه توان
Format bank	1.41	۲ رقم اعشاری
Format +	+	مثبت به منفی یا صفر
Format rat	1393/985	نمایش کسری

با تایپ کردن هر کدام از این دستورات در محیط matlab محاسبات با این فرمت انجام می شود.برای محاسباتی که نیاز به دقت بالا داریم و می خواهیم خطاهای گرد کردن توسط کامپیوتر حداقل شود می توان از Format long استفاده کرد.

در باره ی متغییر ها :

در نامگذاری متغییرها حروف بزرگ و کوچک یکسان در نظر گرفته نمی شوند. نام متغییر باید حداکثر از ۱۹ کارکتر تشکیل شود و کارکترهای بعد از آن حذف می شوند.اولین کارکتر تشکیل دهنده ی نام متغییر ، با حرف الفبا شروع می شود و به دنبال این کارکتر می تواند کارکترهای حروف الفبا، ارقام و علامت – قرار بگیرد.از کارکترهای نقطه گذاری نباید استفاده شود زیرا این کارکترها در matlab معانی ویژه ای دارند.

چند متغییر ویژه :

Ans= متغییری که به طور پیش فرض برای نتایج استفاده می شود.مثلا اگر تایپ کنیم $\sqrt{2}$ بعد از زدن Enter، مقدار $\sqrt{2}$ در متغییر ans ، ذخیره می شود.

Pi= همان π≈۳,۱۴ می باشد.

eps= کوچکترین عدد در matlab که اگر با یک جمع شود،عددی بزرگتر از یک را می دهد.

Flops=تعداد عملیات ممیز شناور.

Inf=بی نهایت $\left(\frac{1}{0}\right)$.هرگاه در محاسبات عمل تقسیم بر صفر رخ دهد این متغییر به عنوان خطا داده می شود.

Nan = مقدار مبهم $\left(\frac{1}{0}\right)$.هر گاه تقسیم صفر بر صفر رخ دهد این متغییر به عنوان خطا داده می شود.

Real min = کوچکترین عدد حقیقی مثبت.

Real max = بزرگترین عدد حقیقی مثبت.

تذکر : اگر بعد از فرمان، از نمادی استفاده نشود ، نتیجه آن فرمان نمایش داده می شود ولی علامت سیمیکالون (;) از چاپ نتیجه جلوگیری می کند.

>> $\sqrt{2}$ **>>** $\sqrt{2}$ **,** **>>** $\sqrt{2}$ **;**

در حالت اول و دوم بعد از زدن enter حاصل $\sqrt{2}$ **داده می شود ولی در حالت آخر ، خروجی داده نمی شود بلکه مقدار در متغییر ans یا متغییر داده شده ذخیره می شود.دانستن این مطلب در محیط برنامه نویسی ضروری می باشد.**

توابع ریاضی:

توابع معروفی که در matlab قابل دسترسی می باشند در زیر داده شده اند.

abs(x) ⟵ قدر مطلق عدد حقیقی x یا طول عدد مختلط x

acos(x) ⟵ Arc cos (x)

a cos h(x) ⟵ Arc cos h (x)

a sin(x) ⟵ Arc sin (x)

a sin h(x) ⟵ Arc sin h (x)

a tan (x) ⟵ Arc tg (x)

con j(x) ⟵ مزدوج عدد مختلط x

cos (x) ⟵ **کسینوس**

cos h(x) ⟵ **کسینوس هیپربولیک**

exp(x) ⟵ e^x

fix(x) ⟵ **به سمت صفر گرد می کند**

gcd(x) ⟵ **بزرگترین مقسوم علیه مشترک دو عدد صحیح x و y**

imag(x) ⟵ **قسمت موهومی عدد مختلط x**

Lcm(x,y) ⟵ **کوچکترین مضرب مشترک دو عدد صحیح x و y**

log(x) ⟵ **Ln(x)**

log10(x) ⟵ **لگاریتم در مبنای** $\log_{10} x$

real(x) ⟵ **قسمت حقیقی عدد مختلط x**

rem(x,y) ⟵ **باقیمانده تقسیم**

round(x) ⟵ **به سمت نزدیکترین عدد صحیح گرد می کند**

sign(x) ⟵ **علامت x را بر می گرداند**

sin(x) ⟵ **سینوس**

sin h(x) ⟵ **سینوس هیپربولیک**

sqrt(x) ⟵ **جذر عدد x**

tan(x) ⟵ **تانژانت**

tan h(x) ⟵ **تانژانت هیپربولیک**

چند تابع M– فایل :

از این توابع برنامه نویس بر اساس نیاز خود می تواند در برنامه استفاده کند.

Input= برای گرفتن ورودی از کاربر منتظر می ماند.در نوشتن برنامه ها می توان در مواقع لزوم از این تابع استفاده کرد.

Key board= موقتا ، کنترل برنامه به صفحه کلید داده می شود.در این حالت برنامه تا قبل از کلمه ی **key board** اجرا شده است و می توان هر تغییراتی در برنامه داد.برای خروج از این حالت و ادامه ی اجرای برنامه کلمه ی **retrun** را باید تایپ کرد.

Pause= توقف در اجرای برنامه ایجاد می شود تا زمانی که کاربر کلیدی را فشار دهد.

Pause(n)= به مدت **n** ثانیه اجرای ادامه ی برنامه متوقف می شود.

کنترل پنجره فرمان :

clc= متن موجود در پنجره فرمان را پاک می کند و یک صفحه ی جدید در اختیار شما قرار می دهد.

Diary= متن پنجره فرمان را در یک فایل ذخیره می کند.در مواقعی که می خواهیم خروجی برنامه ذخیره شود از این دستور استفاده می کنیم.

Home= مکان نما را به گوشه چپ بالا،منتقل میکند.

More= پنجره فرمان را صفحه به صفحه متوقف می کند.در مواقعی که می خواهیم اجرای برنامه چندین صفحه می باشد برای مطالعه هر صفحه از این دستور می توان استفاده کرد.

فرمان **help** :

اگر نام عنوان را بدانیم و بعد از کلمه ی **help** آن را تایپ کنیم ، راهنماییهای لازم و مثال در مورد آن عنوان آورده می شود ولی اگر عنوان ناشناخته باشد،پیغام خطا داده می شود.

مثلا با تایپ

```
>>help sqrt
```

تمام اطلاعات مربوط به تابع **sqrt** همراه با یک مثال آورده می شود.

اگر فرمان help بدون عنوان تایپ شود بعد از اجرا تمام عنوان های شناخته شده برای matlab را لیست می کند.از داخل این لیست می توان عنوان مورد نظر را انتخاب کرد.

آرایه ها :

برای ساختن آرایه ها چندین روش وجود دارد با توجه به اینکه اندیس آرایه در matlab با ۱ شروع می شود.

۱-عناصر آرایه را خودمان تایپ کنیم.باید بین این عناصر یک فاصله زده شود. مثلا:

```
>> x = [ 10  2  3  14 ] ;
```

یک آرایه با ۴ عنصر می باشد که داریم

$$X(1) = 10 , x(2) = 2 , x(3) = 3 , x(4) = 14$$

حال آرایه ی x به عنوان دامنه برای هر تابعی که تعریف شود می تواند عمل کند.مثلا اگر

```
>> y = x ^ 2 ;
```

در این صورت y آرایه ای با ۴ عنصر به صورت زیر می باشد.

$$Y(1) = 100 , y(2) = 4 , y(3) = 9 , y(4) = 196$$

```
>> x (3)

Ans = 1.4
```

خروجی ۱,۴ را می دهد.

دستور :

```
>> x (2 : 5 )

Ans = 1.2     1.4        1.6     1.8
```

عناصر x(۲) تا x(۵) را می دهد.

دستور :

>> x (4 : -1 : 2)

Ans = 1.6 1.4 1.2

با عنصر (۴)x شروع می کند هر بار یکی کم می کند تا عنصر (۲)x

دستور :

>> x (2 : 2 : 5)

Ans = 1.2 1.6

با عنصر (۲) x شروع می کند هر بار دو واحد اضافه می کند و هنگامی که به عنصر (۵)x رسید متوقف می شود.در این مثال افزودن ۲ به عدد۴ عدد ۶ را تولید می کند که از ۵ **بزرگتر است بنابراین عنصر ششم یعنی (۶)x درنظر گرفته نمی شود.**

دستور :

>> y = x ([2 1 4])

Y = 1.2 1 1.6

باعث می شود آرایه y با ۳ عنصر به صورت زیر ساخته شود.

Y(1) = x(2) , y(2) = x(1) , y(3) = x(4)

اگر بین عناصر یک آرایه سمیکالون قرار داده شود، آن آرایه ها حالت ستونی چاپ می شود.

>> x (5 ; -1 ; 6)

بعد از زدن Enter بردار ستونی x تولید می شود

X =

5

-1

6

با استفاده از عملگر ترانهاده (´.) نیز می توان بردار ستونی ایجاد کرد

>> x = 1 : 4

با اجرای دستور :

X = 1 2 3 4

۲- در حالت x = a : h : b دراین صورت آرایه ی x با شروع a و با ختم به b با طول گام h تولید می شود.

مثلا اگر :

>> x = 0 : 0.2 : 1 ;

در این صورت آرایه ی x با ۶ عنصر به صورت زیر تولید می شود.

X(1) = 0 , x(2) = 0.2 , x(3) =0.4 , x(4) = 0.6 , x(5) = 0.8 , x(6) = 1

حال از آرایه ی x می توان به عنوان دامنه ی یک تابع استفاده کرد. مثلا اگر :

>> y = sin (x);

در این صورت آرایه ی y با ۶ عنصر به صورت زیر ساخته می شود.

y(1) = 0 , y(2) = 0.1987 , y(3) =0.3894 , y(4) = 0.5646 , y(5) = 0.7174 , y(6) = 0.8415

اگر h داده نشود یعنی x=a : b در این صورت h =1 در نظر گرفته می شود.

۳- اگر تایپ کنیم x = Lin space (a,b,n)، در این صورت با شروع a و ختم b ، آرایه ی x با n عنصر ساخته می شود. مثلا اگر:

>> x = Lin space (0,2,5) ;

در این صورت آرایه ی x با ۵ عنصر به صورت زیر ساخته می شود.

X(1) = 0 , x(2) = 0.5 , x(3) =1 , x(4) = 1.5 , x(5) = 2

یعنی توسط این تابع می توانیم مستقیما تعداد عناصر آرایه (n عنصر) را مشخص کرد.

تابع Log space آرایه ای با مقیاس لگاریتمی ایجاد می کند. درحالت Log space (a,b,n) آرایه ای n عنصری با شروع 10^a و 10^b ختم ایجاد می شود.مثلا اگر:

>> x = Log space (0 , 2 , 3) ;

در این صورت آرایه ی x با ۳ عنصر به صورت زیر ساخته می شود.

X(1) = 1 , x(2) = 10 , x(3) =100

فراخوانی عناصر آرایه :

دستور x(i) باعث می شود عنصر i ام آرایه x داده شود.

دستور x(i : j) باعث می شود از عنصر i ام تا عنصر j داده شود.

دستور x(i : -1 : j) باعث می شود از عنصر i شروع کند هر بار یکی کم کند و در j توقف کند.(i ≥ j)

دستور x(i : n : j) باعث می شود با عنصر i شروع شود در هر مرحله n گام به جلو رود و به این طریق تا عنصر j ام داده شود.

دستور y=x([2 3 8]) باعث می شود آرایه ی جدید y ساخته شود که عنصر اول آن عنصر دوم x، عنصر دوم آن عنصر سوم x وعنصر سوم y، عنصر هشتم xباشد.یعنی

Y(1) = x(2) , y(2) = x(3) , y(3) = x(8)

مثلا اگر

>> x = 1 : 0.2 : 2 ;

در این صورت آرایه ی x به صورت زیر ساخته می شود.

X(1) = 1 , x(2) = 1.2, x(3) =1.4 , x(4) = 1.6 , x(5) = 1.8 , x(6) =2

در این صورت دستور:

>> y = x´

بردار ستونی y تولید می شود

Y = 1

2

3

4

عملگر نقطه – ترانها ده (.´) به عنوان ترانهادی مزدوج تعریف می شود. یعنی آرایه ی سطر به ستونی و یا آرایه ی ستونی به آرایه ی سطری تبدیل می شود و عناصر آن مزدوج می شوند.می دانیم اگر z= a + i b در این صورت مزدوج آن z= a – i b می باشد.

```
>> a = (x * i) + 1

   1 + i        1 + 2i        1 + 3i      1 + 4i
```

با اجرای دستور

```
>> b = a .´
```

آرایه حالت ستونی و مزدوج می شود و در آرایه ی جدید b ذخیره می شود.

```
b =

   1 - i

   1 - 2i

   1 - 3i

   1 - 4i
```

ساختن ماتریس :

کافی است هر گاه عناصر یک سطر تمام شد سمیکالون زده شود.

```
>> A = [1   -1   2 ;  5  0  2 ; 1   -1]

A=

   1    -1    2

   5    0     2

   1    0     -1
```

برای جمع و تفریق دو آرایه یا ماتریس هم بعد از علامت های (+) و (–) استفاده می کنیم.برای ضرب دو ماتریس (شرایط ضرب برقرار باشد) و یا ضرب یک عدد در ماتریس از علامت (*) استفاده می شود.

علامت ضرب نقطه ای (*.) باعث می شود که در ضرب آرایه ها یا ماتریس،عناصر متناظر در هم ضرب شوند.برای تقسیم آرایه به آرایه نیز از علامت (/.) یا (\.) استفاده می شود. در هر دو حالت آرایه ای که

در زیر اسلش قرار دارد بر آرایه ای که در بالای اسلش قرار دارد تقسیم می شود.اگر A.\B دراین صورت عناصر ماتریس A متناظرا بر عناصر ماتریس B تقسیم می شود که نتیجه ی آن با B./A یکی می باشد.

علامت ^ برای توانهای ماتریس (A^2=A*A) و ^. برای توانهای عنصر به عنصر استفاده می شود.

دستکاری در آرایه ها :

به طور خلاصه چند دستور زیر می تواند در برنامه نویسی استفاده شود.

```
>> x = A (i , : ) ;
```

سطر i ام ماتریس A را در بردار x قرار می دهد.

```
>> x = A (: , i);
```

ستون i ام ماتریس A را در بردار x قرار می دهد.

```
>> A (: , i) = [ ] ;
```

با حذف ستون i ام ماتریس A، ماتریس Aی جدید می سازد.

اندازه آرایه :

در این مورد چند تابع مهم معرفی می شوند.

Whose : متغییرهای موجود در محیط کاری و اندازه آنها را نمایش می دهد.

s = size(A) : بردار s را با دو عنصر برمی گرداند.اولین عنصر تعداد سطرهای A و دومین عنصر تعداد ستون های A می باشد.

[r , c]= = size(A): در متغییر r تعداد سطرها و در متغییر c تعداد ستونها قرار داده می شود.

r = size (A , 1) : تعداد سطرهای A در متغییر r ذخیره می شود.

c = size (A , 2) : تعداد ستون های A در متغییر c ذخیره می شود.

n = Lenth (A) : ماکزیمم تعداد سطر یا ستون ماتریس A را در متغییر n ذخیره می کند.

توابع دستکاری آرایه :

Flipud(A) : ماتریس **A** را واژگون می کند.

rot90(A) : ماتریس **A** را ۹۰ درجه دوران می دهد.

diag(A) : قطر ماتریس **A** را به صورت بردار ستونی می دهد.

diag(V) : یک ماتریس قطری ایجاد می کند که قطر آن بردار **V** باشد.

tril(A) : قسمت پایین مثلثی ماتریس **A** را می دهد.

triu(A) : قسمت بالا مثلثی ماتریس **A** را می دهد.

حل معادلات خطی :

اگر A یک ماتریس مربعی باشد برای حل دستگاه Ax=b می توان از روشهای زیر استفاده کرد.

```
>> x = inv (A) * b
```

که inv(A) یکی از تابع های matlab می باشد و معکوس A را محاسبه می کند.چون برای ماتریسهای با بعد بزرگ محاسبه ی معکوس A مشکل و وقت گیر می باشد این روش کمتر استفاده می شود.روش استفاده از عملگر تقسیم چپ ماتریسی به صورت زیر برای حل دستگاه استفاده می شود.

```
>> x = A\b
```

در این روش از تجزیه LU ماتریس A استفاده می شود.

اگر تعداد معادلات بیشتر از تعداد مجهولات باشد،بهترین روش برای حل دستگاه غیر مربعی Ax=b، استفاده از عملگر / یا \ می باشد.در این روش جواب کمترین مربعات برای دستگاه پیدا می شود.(نسبت به نرم ۲ خطا حداقل می باشد.)

مثلا اگر :

```
>> A = [ 1  2 3 : 4 5 6 : 7 8 0 : 2 5 8 ] ;

>> b = [ 366  ;  804  ;  351  ;  514 ] ;

>> x = A\b

X =
```

247.9818

-173.1091

114.9273

دقت شود این جواب در دستگاه **Ax=b** صدق نمی کند،بلکه خطای رخ داده شده نسبت به فرم ۲ حداقل می باشد. داریم:

```
>> e = A * x – b
e =
   -119.4545
     0
    35.8364
```

بردار خطا یعنی **e** کمترین مقدار $\|e\|_2$ را دارد، نسبت به هر **x** دیگری که به عنوان جواب تقریبی برای این دستگاه در نظر گرفته شود.

روش دیگر،محاسبه ی **x=pinv(A)*b** می باشد،که اساس این روش برای حل استفاده از شبه وارون می باشد.در این روش طول یا نرم **x** (جواب) از نرم همه ی جوابهای ممکن کوچکتر خواهد بود.

```
>> A = [ 1  4  7  2 ; 2  5  8  5 ; 3 6 0  8 ] ;
>> b = [ 366    804    351 ]´ ;
>> x = A \ b
X =
    0
   -165.9000
    99.000
    168.3000
>> y = pinv(A)*b
Y =
```

30/8182

-168/9818

99/0000

159/0545

در جواب x، مقدار خطا یعنی e = A*x-b نسبت به نرم ۲ ، نسبت به تمام جوابهای تقریبی دیگر حداقل می باشد و لی در جواب y ، نرم این جواب نسبت به نرم x کمتر می باشد.

```
>> norm (x)
ans =
        256.2200
>> norm (y)
ans =
        254.1931
```

همانگونه که مشاهده می شود

norm (y) < norm (x)

```
>> norm (A*x-b)
ans =
        3.5951e-013
>> norm (A*y-b)
ans =
        4.8233e-013
```

ولی در نرم خطا برای جواب x از جواب y کمتر می باشد.چون برای x نرم خطا 3.5951×10^{-13} و برای y ، $4.8233\times^{10-13}$ می باشد

توابع ماتریسی :

chol (A) : تجزیه چولسکی ماتریس A

cond(A) :عدد شرطی ماتریس.

d = eig (A) : بردارهای ویژه ی ماتریس **A**

[V , D] = eig (A : مقادیر ویژه و بردارهای ویژه ی ماتریس **A**

det (A) : دترمینال ماتریس **A**

expm (A) : ماتریس نمایی **e^A .A**

expm2 (A) : ماتریس نمایی با استفاده از سری تیلور.

expm3(A) : ماتریس نمایی با استفاده از مقادیر ویژه و بردارهای ویژه.(به یک کتاب جبر خطی عددی مانند **[10]** مراجعه شود.)

hess(A) : فرم هسنبرگی ماتریس **A** را تولید می کند.

inv(A) : معکوس ماتریس **A** را محاسبه می کند.

Logm(A) : محاسبه ی لگاریتم ماتریس **A.(Ln A)**

Lu(A) : تجزیه **Lu** ماتریس **A** به روش حذف گاوسی.

norm(A,1) : نرم یک ماتریس یا بردار **A**.

norm(A,2) : نرم دوم(نرم اقلیدسی) ماتریس یا بردار **A**.

norm(A,inf) : نرم بی نهایت.

norm(A,P) : نرم **P** برای بردار **A**.

pinv(A) : متعامد سازی.

poly(A) : چند جمله ای مشخصه ی ماتریس **A.**

Poly valm(A) : محاسبه ی چند جمله ای مشخصه ماتریس **A.**

qr(A) : تجزیه مثلثی متعامد ماتریس **A .**

qz(A,B) : مقادیر ویژه ی تعمیم یافته.

rank(A) : مینیمم تعداد سطرها یا ستونهای مستقل خطی ماتریس **A.**

rref(A) : سطری پلکانی شده ماتریس **A** .

schur(A) : تجزیه شور ماتریس **A** .

sqrtm(A) : ریشه ی دوم ماتریس **A** .

svd(A) : تجزیه ی مقدار تکین ماتریس **A** .

trace(A) : مجموع درآیه های قطری ماتریس **A** .

ماتریس های ویژه :

در matlab تعدادی ماتریس مبهم وجود دارند که دارای کاربردهایی در برنامه نویسی می باشند.

Zeros(n) : یک ماتریس **n×n** ، با درآیه های صفر تولید می کند.

ones(m,n) : یک ماتریس **m×n** ، با درآیه های یک تولید می کند.

Rand(m,n) : : یک ماتریس **m×n** که اعدادش اعداد تصادفی با توزیع یکنواخت در فاصله ی صفر و یک می باشند ، ساخته می شود.

rand(n) : یک ماتریس **n×n** ، تصادفی تولید می کند.

eye(n) : ماتریس همانی با بعد **n** تولید می کند.

eye(m,n) : یک ماتریس **m×n** همانی تولید می کند.

hadamard(n) : یک ماتریس هادامارد **n×n** تولید می کند.

hankel(n) : یک ماتریس هنکل **n×n** تولید می کند.

hilb(n) : ماتریس هیلبرت **n×n** تولید می شود.

Invhilb(n) : وارون ماتریس هیلبرت **n×n** تولید می شود.

magic(n) : مربع جادویی **n×n** تولید می شود.

pascal(n) : ماتریس مثلثی پاسکال **n×n** را تولید می کند.

vander(n) : ماتریس واندرموند **n×n** تولید می شود.

wilkinson(n) : ماتریس تست **n×n** مقدار ویژه ی ویلکینسون تولید می شود.

ساختارهای تصمیم گیری :

در matlab سه ساختار تصمیم گیری یا کنترلی وجود دارد.حلقه ی For ، حلقه ی while و ساختار if-else-End . این ساختارها بسیاری از فرمانهای matlab را شامل می شوند و در قسمت برنامه نویسی بسیار و به سهولت استفاده می شوند.برای وارد شدن به محیط برنامه نویسی از منوی File گزینه ی M File– را کلیک می کنیم.بعد از نوشتن برنامه در محیط داده شده،برنامه را ذخیره می کنیم. برای اجرای برنامه کافی است که در محیط اصلی Matlab نام برنامه را نوشته و دکمه ی Enter را بزنیم.

حلقه ی For :

با این حلقه می توان گروهی از فرمانها را به تعداد دفعات ثابت از قبل مشخص شده،تکرار کرد.فرم کلی حلقه ی For به صورت زیر می باشد.

For x = array

مجموع دستورات

End

یعنی برای هر for یک end لازم می باشد.

```
For     n = 1 : 10
X (n) = sin (n * pi/10) ;
End
>>  x =
        0.3090    0.5878    0.8090    0.9511    1.000    0.9511              0.8090
0.5878    0.3090    0.0000
```

در این مثال با مقدار دهی به n با ۱۰ تکرار مقادیر محاسبه می شود.

حلقه ی while :

بر خلاف حلقه ی for که تعدادی از دستورات را به تعداد دفعات ثابت انجام می دهد،حلقه ی while گروهی از فرمانها را به تعداد دفعات نامشخص ، تا بر قرار بودن شرط اجرا می کند.

while (شرط)

دستورات

End

تا موقعی که ارزش شرط درست می باشد این مجموعه دستورات اجرا می شود. مانند حلقه ی For متناظر با دستور while , end لازم می باشد.

```
i = 1 ;
while   i < 10
x (i) = i ;
i = i + 1 ;
end
```

خروجی این برنامه به صورت زیر می باشد.

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

به عنوان نمونه به قسمت برنامه های نوشته شده مراجعه شود.

ساختار if – else – if :

در بسیاری مواقع ، دنباله ای از فرمانها باید به طور شرطی و بر مبنای تست رابطه ای ، ارزیابی شوند. این نوع ساختار در محیط های برنامه نویسی بسیار استفاده می شود.

```
For    i = 1 : 10
  if   ( i >= 1 )  AND  ( i < 5 )
x ( i ) = i ^ 2 ;
else
x ( i ) = i ;
        end
 end
```

خروجی این برنامه به صورت زیر می باشد.

X = 1 4 9 16 5 6 7 8 9 10

در قسمت شرط (if) باید تمام شرایط برقرار باشند تا دستورات بعد از if اجرا شود،در غیر این صورت دستورات بعد از elseاجرا می شود.

چند جمله ایها :

در matlab یک چند جمله ای با بردار سطری ضرایبش در ترتیب نزولی ، نمایش داده می شود. مثلا چند جمله ای $x^4 - 12x^3 + 25x + 16$ به صورت زیر وارد می شود.

>> P = [1 -12 0 25 16]

P = 1 -12 0 25 116

دقت شود جمله هایی که ضریب صفر دارند باید وارد شوند. با مشخص بودن بردار ضرایب چند جمله ای به کمک توابع roots ریشه های چند جمله ای محاسبه می شوند.

>> x = roots (p)

X = 11.7473

2.7028

-1.2251 + 1.4672i

-1.2251 – 1.4672i

با مفروض بودن ریشه ای یک چند جمله ای می توان آن چند جمله ای را ساخت. بدین منظور تابع poly به کار می رود. با استفاده از مثال قبل اگر قرار دهیم.

>> p = poly (x)

P = 1.0e + 02*

0.0100 -0.1200 -0.0000 0.2500 1.1600+0.0000i

بردار p به عنوان ضرایب چند جمله ای که بردارx ریشه های آن می باشد داده می شود.)
($1.0e + 02 = 10^2$ به علت خطای گرد کردن ممکن است ضرایب به صورت مختلط تولید شوند.برای استخراج ضرایب به صورت حقیقی از تابع real استفاده می شود.

>> P = real (p)

P = 1.000 -12.0000 -0.0000 25.0000 116.0000

ضرب چند جمله ایی :

با مشخص بودن ضرایب چند جمله ایها ، تابع conv ، عمل ضرب را انجام می دهد.مثلا اگر:

در این صورت

>> a = [1 2 3 4] ; b = [1 4 9 16];

>> c = conv (a , b)

C = 1 6 20 50 75 84 64

یعنی حاصلضرب a(x) در b(x)به صورت زیر می باشد.

$$c(x) = x^6 + 6x^5 + 20x^4 + 50x^3 + 75x^2 + 84x + 64$$

برای ضرب بیش از دو چند جمله ای باید از تابع conv را به طور مکرر ستفاده کرد.

جمع چند جمله ای ها :

در matlab تابع مستقیمی برای جمع چند جمله ایها وجود ندارد.اگر دو بردار ضرایب چند جمله ایها اندازه یکسان داشته باشند، می توان از جمع آرایه ای استفاده کرد. در مثال قبل قرار می دهیم :

>> d = a + b

d = 2 6 12 20

یعنی

$$d(x) = a(x) + b(x) = 2x^3 + 6x^2 + 12x + 20$$

در صورتی که بردار ضرایب دو چند جمله ای برابر نباشند با اضافه کردن صفر آنها را برابر می کنیم. مثلا

>> e = c + [0 0 0 d]

e = 1 6 20 52 81 96 84

برای اینکه دو چند جمله ای c(x) و d(x) قابل جمع باشند ، ضرایب x^6 و x^5 و x^4 را در d(x) صفر قرار می دهیم تا هر دو چند جمله ای ضرایب آنها آرایه ای ۷ عضوی شود.

در جعبه ابزار matlab پیشرفته تابع mmpadd برای جمع دو چند جمله ای استفاده می شود.

```
>> f = mmpadd ( c , d )
```

f = 1 6 20 52 81 96 84

در این حالت مانند قبل دیگر لازم نیست بعد بردار ضرایب دو چند جمله ای یکسان باشند. از mmpadd می توان برای عمل تفریق نیز استفاده کرد.

```
>> g = mmpadd (c , -d)
```

g = 1 6 20 48 69 72 46

یعنی

$$g(x) = c(x) - d(x) = x^6 + 6x^5 + 20x^4 + 48x^3 + 69x^2 + 72x + 44$$

تقسیم چند جمله ایها :

برای تقسیم دو چند جمله ای در matlab از تابع decnov استفاده می شود.

```
>> [q , r ] = decnov (c , d)
```

q = 1 2 3 4

r = 0 0 0 0 0 0 0

در این مثال چند جمله ای c(x) بر d(x) تقسیم می شود.q(x) به عنوان خارج قسمت و r(x) باقیمانده می باشد.c(x) = d(x) q(x) r(x)

مشتق گیری :

در matlab تابع polyder برای مشتق گیری از یک چند جمله ای استفاده می شود.از قبل تابع g(x) را داریم.قرار می دهیم

```
>> h = polyder(g)
```

h = 6 30 80 144 138 72

یعنی

محاسبه چند جمله ایها :

با استفاده از تابع polyval به صورت polyval(P , x) می توان چند جمله ای p(x) را در نقطه ای داده شده ی x محاسبه کرد.

```
>> x = 1 : 1 : 5 ;
>> p = [ 1    6 ] ;
>> v = polyval (p,x)
```

v = 7 8 9 10 11

در این مثال P(x) = x + 6 در نقاط 5,4,3,2,1 محاسبه می شود و در بردار v ذخیره می شود. با استفاده از تابع plot می توان نمودار این تابع را رسم کرد.

```
>> plot(x , v) ;
```

نموداری رسم می شود که نقاط بردار x طول و نقاط بردار v عرض آن می باشند.

چند جمله ایهای کسری :

تابع residue ، عمل بسط جزیی کسرها را انجام می دهد.مثلا اگر کسر زیر را داشته باشیم :

در این حالت صورت کسر به طریق f = 10 * [1,2] وارد می شود و مخرج به صورت g = poly ([-1 ; -3 ; -4]) وارد می شود.

```
>> f = 10 * [1,2]
>> g = poly ([-1 ; -3 ; -4]);
>> [x, y, k] = residue (f , g)
```

X =

-6.6667

5.0000

1.6667

Y =

-4.0000

3.0000

-1.0000

K =

[ ]

بردار x صورت کسرهای تجزیه شده ، بردار y متانظر (ریشه های مخرج) و k جمله ی ثابت بسط جزیی کسر می باشد. یعنی

تابع residue عکس عمل فوق را نیز انجام می دهد.

>> [n , d] = residue [x , y , k]

n = - 0.000 10.000 20.000

d = 1.000 8.000 19.000 12.000

که n ضرایب صورت کسر یعنی صورت 10x +20 و d ضرایب چند جمله ای مخرج می باشد یعنی مخرج به صورت زیر می باشد.

تابع polyder از چند جمله ایهای کسری نیز مشتق می گیرد به این طریق که می نویسیم
polyder(x,y) که x صورت کسر و y مخرج کسر می باشد.

>> [a,b] = polyder (f , g)

a = -20 -140 -320 -260

b = 1 16 102 328 553 456 144

a ضرایب صورت و b ضرایب مخرج حاصل مشتق گیری می باشد. با توجه به توابع f و g

$\frac{d}{ds}$

برازش منحنی :

در matlab تابع polyfit مساله برازش منحنی کمترین مربعات را حل می کند. برای استفاده از این تابع باید داده ها و درجه ی چند جمله ای که می خواهد بهترین برازش برای داده ها باشد به تابع داده شود.اگر مرتبه را n=1 بدهیم،بهترین تقریب خط راست حاصل می شود که معمولا رگرسیون خطی نامیده می شود. اگر مرتبه را n=2 قرار بدهیم،یک چند جمله ای درجه ی ۲ پیدا می شود.این تابع به صورت Polyfit(x,y,n) به کار می رود که x نقاط طول ، y نقاط عرض و n درجه ی چند جمله ای برازش دهنده می باشد.

```
>> x = [ 0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1 ];
>> y = [ -0.447  1.978  3.28  6.16  7.08  7.34  7.66  9.56  9.48  9.30  11/2 ];
>> p = polyfit (x , y , 2)
P =      -9.8108          20.1293          -0.0317
```

یعنی بهترین چند جمله ای درجه ی ۲ برازش دهنده بر نقاط (x , y) به صورت زیر می باشد.

درونیاب یک بعدی :

ساده ترین درونیابها در matlab نمودارهایی می باشند که توسط تابع plot رسم می شوند.این نوع درونیاب،درونیاب خطی می باشد. یعنی نقاط داده شده توسط خطوط راست به هم وصل می شوند.اگر تعداد نقاط درونیاب زیاد باشد وصل شدن خطوط راست به یکدیگر قابل مشاهده نیست و نمودار تابع به صورت منحنی رسم می شود.

```
>> x1 = Lin space (0 , 2*pi , 60)
>> x2 = Lin space (0 , 2*pi , 6)
>> plot (x1,sin(x1),x2,sin(x2),´—´ )
>> xlabel ( ´ x ´ ) , ylabel(´ sin(x)´) , title ( ´ linear interpolation´ )
```

در حالت اول درونیاب با ۶۰ نقطه و در حالت دوم با ۶ نقطه رسم می شود. برای برچسب زدن روی محور xها از دستور xlabel و برچسب محور yها از ylabel استفاده می شود.برای نوستن عنوان نمودار از دستور title استفاده می شود.

برای محاسبه ی درونیاب یک نقطه ای از تابع interp1به صورت interp1(x,y,x0) استفاده می شود. که x نقاط طول و y نقاط عرض برای محاسبه ی تابع درونیاب و x0 نقطه ای می باشد که می خواهیم تابع درونیاب در آن محاسبه شود.

```
>> x = 1 : 12 ;
>> y = [ 5  8  9  15  25  29  31  30  22  25  27  24 ] ;
>> interp1 (x , y , 9.3)
ans = 22.9000
```

می توان مقدار تابع درونیاب را در بیشتر از یک نقطه نیز محاسبه کرد.

```
>> interp1 (x , y , [ 3.2  6.5  7.1  11.7 ] )
ans =
        10.2000
        30.0000
      30.9000
        24.9000
```

تابع interp1 ، درونیاب اسپیلاین درجه ی سه را نیز ارائه می کند.به صورت

Interp 1(x , y , x0 ,´ spline´)

```
>> interp1(x , y , 9.3 ,´ spline´)
ans = 21.8577
```

نتیجه بدست آمده همانطور که می دانید از درونیاب معمولی بهتر می باشد. قابل ذکر است که در تابع interp1 دو محدودیت وجود دارد. اولا x_0 باید در محدوده ی دامنه ی تابع باشد تا مقدار درونیاب در نقطه ی x_0 داده شود (اگر x_0در محدوده ی دامنه نباشد ، برونیابی نامیده می شود.) مثلا دستور زیر اجرا نمی شود.

```
>> interp1 ( x , y , 12.5 )
```

چون x بین ۱ و ۱۲ تغییر می کند.

ثانیا متغییر مستقل (دامنه) باید یکنواخت باشد. یعنی مقادیر متغییر مستقل همیشه باید افزایش یا کاهش یابد. در مثال زیر interp1 تابع قابل اجرا نمی باشد.

```
>> x = [ 7 : 12    1 : 6 ]
X = 7  8  9  10  11  12  1  2  3  4  5  6
```

چون متغییر مستقل x یکنواخت نمی باشد.

درونیاب دو بعدی :

مانند درونیاب یک بعدی می باشد با این تفاوت که تابع دو متغییره z=f(x,y) درونیابی می شود. مثال زیر را در نظر می گیریم.

```
>> x = 1 : 5 ;
>> y = 1 : 3 ;
>> z = [ 82  81  80  82  84 ; 79  63  61  65  81 ; 84  84  82  85  86]
```

در این مثال x,y متغییرهای مستقل و z تابع می باشد. مثلا z(1,1)=81 ، z(5,3)=86 . فرض کنید می خواهیم در y ثابت ۲، مقدار درونیاب z را در نقاط با طول گام ۰,۲ محساسبه کنیم. قرار می دهیم :

```
>> x1 = 1 : 0.2 : 5 ;
>> y1 = 2 ;
>> z1 = interp 2 (x , y , z , x1 , y1 ) ;
>> z2 = interp 2 (x , y , z , x1 , y1 , ´ cubic´ ) ;
>> plot = (x1 , z1 , ´ —´ , x1 , z2 ) ;
```

در این مثال z1 مقادیر درونیاب دو بعدی خطی تابع z =f (x , y) برای نقاط x1 و y1 ثابت می باشد و z2 درونیاب درجه سوم دوبعدی برای تابع z می باشد. با کمک تابع plot نمودار (x1 , z1)به صورت خط و تابع (x1 , z2) در حالت عادی رسم می شود. با توجه به شکل درونیاب z2 بهتر می باشد. حال اگر بخواهیم درونیاب را در دو جهت انجام دهیم ، می نویسیم :

```
>> x1 = 1 : 0.2 : 5 ;
>> y1 = 1 : 0.2 : 3 ;
>> z1 = interp 2(x , y , z , x1 , y1 ,´cubic´) ;
```

در این حالت متناظر با هر نقطه ی x1 و y1 یک مقدار برای بردار z1 تولید می شود. چون x1 شامل ۴۱ عنصر و y1 شامل ۲۱ عضو می باشد پس z1 شامل ۲۱×۴۱ عنصر می باشد که مقادیر درونیاب تابع z در نقاط گره ای x1 و y1 می باشد. با استفاده از تابع mesh می توان شکل ۳بعدی حاصل را رسم کرد.

```
>> mesh(x1, y1 , z1)
```

به طور کلی برای تابع درونیاب دو بعدی تابع interp2 به صورت زیر استفاده می شود.

Interp2(x , y , z , x1 , y1 , method)

که y و x متغییرهای مستقل و z تابع می باشد. X1 بردار نقاط گره ای طولها ، y1 بردار نقاط گره ای عرضها و method نوع درونیاب را مشخص می کند. اگر method بیان نشود ، درونیاب خطی دو بعدی ، اگر cubic نوشته شود، درونیاب درجه سوم دو بعدی و اگر nearest نوشته شود ، درونیاب دو بعدی نزدیکترین همسایگی اجرا می شود.

اسپلاین درجه ی سوم :

تابع اسپلاین در ساده ترین حالت ، داده های x و y و مقادیر دلخواه x1 را دریافت می کند و چند جمله ای اسپلاین درجه ی سوم درونیاب که x و y را برازش می کنند ، پیدا می کند و سپس با محاسبه ی چند جمله ای برای هر عنصر x1 مقدار y1 متناظر ب آن را محاسبه می کند.

مثال زیر را در نظر می گیریم :

```
>> x = 0 : .2 ;
>> y = tan (pi * x/25) ;
>> x1 = Lin space ( 0 , 12 ) ;
```

```
>> y1 = spline ( x , y , x1 ) ;
>> plot = ( x , y , x1 , y1 , '—' ) ;
```

در این مثال نقاط گره ای x با مقادیرشان y داده شده است. حال به کمک تابع spline درونیاب spline برای نقاط (x , y) نوشته می شود و این تابع درونیاب در نقاط x1 محاسبه می شود و مقادیر آن در بردار y1 ذخیره می شود. به کمک تابع plot نقاط (x , y) و (x1 , y1) به صورت خطی به هم وصل می شوند.

اگر تابع spline فقط با دو آرگومان اول فراخوانی شود در این صورت آرایه ای که شامل تمام اطلاعات مورد نیاز برای محاسبه ی اسپلاین درجه ی سوم می باشد را ارائه می کند.

```
>> P = spline (x , y) ;
```

با استفاده از تابع PPval با معین بودن فرم P ، اسپلاین درجه ی سوم را می توان در مجموعه ای از نقاط محاسبه کرد.

```
>> x1 = Lin space (10 , 12) ;
>> y1 = PPval (p , x1) ;
```

در این صورت مقادیر تابع درونیاب P که قبلا برای داده های (x , y) ساخته شده است در نقاط x1 محاسبه می شوند و در بردار y1 ذخیره می شوند. حتی می توان خارج از ناحیه ای که چند جمله ایهای اسپلاین محاسبه شده اند مقادیر اسپلاین را بدست آورد. اگر داده ها بعد از آخرین یا قبل از اولین نقطه ی دامنه ظاهر شوند ، برای یافتن مقادیر درونیابی شده از اولین و آخرین چند جمله ای درجه ی سوم استفاده شود.

```
>> x1 = 10 : 15 ;
>> y1 = PPval (p , x1) ;
Y1 = 3.0777   5.2422   15.8945   44.0038   98.5389   188.4689
```

در matlabمی توان به کمک تابع unmkpp اطلاعات مربوط به اسپلاین ساخته شده و ضابطه ی چند جمله ایهای اسپلاین را مشاهده کرد. فرض کنید تابع اسپلاین P ساخته شده باشد. قرار می دهیم :

```
>> [ a , b , c , d ] = unmkpp (p)
a =  0   1   2   3   4   5   6   7   8   9   10
b =
```

0.0007	−0.0001	0.1257	0
0.0007	0.0020	0.1276	0.1263
0.0010	0.0042	0.1339	0.2568
0.0012	0.0072	0.1454	0.3958
0.0024	0.0109	0.1635	0.5498
0.0019	0.0181	0.1925	0.7265
0.0116	0.0237	0.2344	0.9391
−0.0083	0.0586	0.3167	1.2088
0.1068	0.0336	0.4089	1.5757
−0.1982	0.3542	0.7967	2.1251
1.4948	−0.2406	0.9102	3.0777
1.4948	4.2439	4.9136	5.2422

c = 12

d = 4

در این مثال، a نقاط انفصال را مشخص می کند. B ماتریسی است که سطر iام آن ضرایب iامین چند جمله ای اسپلاین می باشد. مثلا ضابطه ی اولین چند جمله ای این اسپلاین به صورت زیر می باشد.

$$S0\ (x) = \ 0.0007x^3 - 0.0001x^2 + 0.1257x$$

c تعداد چند جمله ایها (تعداد ضابطه ها) برا برای این اسپلاین بیان می کند و d تعداد ضرایب هر چند جمله ای را مشخص می کند. دقت شود این فرم کلی می باشد. همانطور که می دانیم چند جمله ایهای اسپلاین درجه سه، حداکثر از درجه ی ۳ می باشند. تابع mkpp با دریافت اجزای شکسته شده ی بالا ، فرم p را به حالت اول باز می گرداند. برای تولید p ، نقاط انفصال و ضرایب هر ضابطه کافی می باشد.

```
>> p = mkpp (a , b) ;
```

انتگرال گیری :

اگر برای داده های (x , y) تابع درونیاب اسپلاین نوشته شده باشد برای محاسبه ی انتگرال هر ضابطه ی این اسپلاین از تابع spintgrl استفاده می کنیم.

```
>> x = ( 0 : 0.1 : 1 ) * 2 * pi ;
>> y = sin (x) ;
>> p = spline (x , y) ;
>> s = spintgrl (p) ;
```

در این صورت s ، درونیاب اسپلاین درجه سوم انتگرال تابع $y = \sin x$ در نقاط x می باشد. یعنی تابع s در نقاط گره ای با تابع زیر برابر می باشد.

$$\int sin\ x\ dx = -cos\ x$$

دیفرانسیل گیری :

تابع spderiv از ضابطه های تابع اسپیلاین مشتق می گیرد. در مثال بالا اگر قرار دهیم

```
>> q = spderiv (p) ;
```

در این صورت ضابطه های تابع q با مشتق گیری از ضابطه های تابع p تولید می شوند.

رسم نمودار :

تابع fplot با دقت بالایی برای رسم نمودار تابع $y = f(x)$ استفاده می شود. مثالهای زیر را در نظر می گیریم.

```
>> f = ´ 2*exp (-x).* sin(x)´ ;
>> fplot (f , [ 0 , 8] ) ;
```

یا اینکه می توان نوشت :

```
>> fplot (´ 2*exp (-x). * sin(x)´ , [0 , 8] ;
```

محاسبه ی ماکزیمم و می نیمم تابع :

در matlab برای محاسبه ی می نیمم تابع یک بعدی دو تابع fmin و fmins ارئه شده است. از آنجایی که ماکزیمم f(x) با می نیمم –f(x) برابر است، از این توابع می توان برای یافتن ماکزیمم نیز استفاده کرد. تابع fmin مانند تابع fplot عمل می کند.

```
>> x =  fmin (´ 2*exp (-x).* sin(x)´ , 2 , 5)
```

x = 3.9270

یعنی مقدار تقریبی می نیمم تابع

$$2e^{-x}sin\ x$$

در فاصله ی

[2 , 5]

در نقطه ی

x=3.9270

رخ می دهد.

>> y = fmin (´ 2*exp (-x).* sin(x)´ , 0 , 3)

y = 0.7854

و مقدار ماکزیمم این تابع در نقطه ی y=0.7854 حاصل می شود. با استفاده از آزمون مشتق به طور تحلیلی طول نقطه ی ماکزیمم $\frac{\pi}{4}$ و طول نقطه ی می نیمم $\frac{5\pi}{4}$
برای این تابع می باشد. در این صورت مقدار خطا به صورت زیر می باشد.

یافتن صفر :

در matlab تابع fzero برای بدست آوردن ریشه ی تابع f به صورت زیر استفاده می شود.

fzero (´f´ , x0)

که f تابع مفروض و x0 نقطه ای می باشد که ریشه ی تابع f در نزدیکی x0 خواسته شده است.

>> fzero (´sin(x)´ , 1)

از این تابع برای بدست آوردن نقاطی که مقدار ثابت c باشند یعنی f(x) = c نیز استفاده می شود. به این طریق که قرار می دهیم g(x) = f(x) – c ، سپس ریشه های g نقاط با مقدار ثابت c برای f می باشند.

انتگرال گیری :

در matlab برای محاسبه ی عددی ناحیه ی زیر یک تابع در فاصله ای معین از سه تابع quad ,trapz و quad8 استفاده می شود. تابع trapz به روش ذوزنقه ای مقدار انتگرال معین را محاسه می کند.

```
>> x = 0 : 0.1 : 2*pi ;

>> y = sin (x) ;

>> A = trapz (x , y)

A =   0.00346
```

توابع quad8 از تابع quad دقیقتر می باشد و این دو تابع مانند fzero فراخوانی می شوند.

```
>> A = quad (´sin(x)´, 0 , 2 * pi )

    A = 0

>> A =   quad (´sin(x)´, 0 , 2 * pi )
   A = −2.2204e −016
```

نمودار های دوبعدی :

همانطور که قبلا گفته شد می توان از تابع plot استفاده کرد.

```
>> x = Lin space ( 0 , 2 * pi , 30 ) ;

>> y = sin (x) ;

>> z = cos (x) ;


>> plot (x , y , x , z);
```

علامت رنگ و شکل خط :

برای تعیین علامت ، رنگ و شکل نمودار می توانیم در فرمان plot ، بعد از طول و عرض آرگومان اضافی وارد کرد. برای نوع رنگ و نوع خط نمودار تابع ، جدول زیر استفاده می شود.

سمبل	رنگ		سمبل	شکل خط
y	زرد		.	نقطه
m	ارغوانی		°	دایره
c	فیروزه ای		×	علامت ×
r	قرمز		+	جمع
g	سبز		*	ستاره
b	آبی		—	خط توپر
W	سفید		:	نقطه چین
K	سیاه			خط چین- نقطه چین
			- -	خط چین

```
>> x = 0 : 0.1 : 2 * pi ;

>> y = sin (x) ;

>> z = cos (x) ;

>> plot = ( x , y , ' g: ' , x , z , ' r- ')
```

افزودن شبکه نقاط و برچسب :

فرمان grid on خطوط شبکه را در محل علامتهای موجود در نمودار جاری اضافه می کند. یعنی حالت شطرنجی ایجاد می کند و دستور grid off خطوط شبکه ای را حذف می کند.

برای قرار دادن رشته ی متنی در نمودار با استفاده از ماوس در پنجره ای که نمودار رسم شده است آیکون **text** را کلیک می کنیم. حال در هر قسمتی از نمودار آماده ایم تا هر متنی که بخواهیم را تایپ کنیم.

فرمان **xlabel** و **ylabel** برای برچسب گذاری محور افقی و عمودی استفاده می شود. فرمان **title** یک خط در بالای نمودار اضافه می کند.

نمودارهای فرعی :

برای اینکه در یک پنجره چندین نمودار را رسم کنیم از تابع **subplot (m , n , p)** استفاده می کنیم. این تابع پنجره را به **m** و **n** قسمت تقسیم می کند و قسمت **p**ام را برای رسم نمودار فعال می کند. نمودارها از چپ به راست و از بالا به پایین شماره گذاری می شوند.

```
>> x = Lin space (0 , 2 * pi , 30) ;
>> y = sin (x) ;
>> z = cos (x) ;
>> subplot  (1 , 2 , 2) ;
>> plot (x , z) ;
```

نمودار های سه بعدی :

در **matlab** برای رسم نمودارهای سه بعدی می توان از تابع **plot3** استفاده کرد. اگر **z=f (x , y)** به صورت ساده بیان شده باشد برای محاسبه ی تمام مقادیر **z** ابتدا باید تمام زوج های مرتب **(x , y)** را تولید کرد و سپس **z** متناظر با هر کدام از این زوج ها محاسبه شود. برای ساختن تمام زوج های مرتب از تابع **mesh grid** استفاده می شود.

```
>> x = -3 : 3 ;
>> y = 1 : 5 ;
>> [ x , y ] = meshgrid (x , y) ;
>> z = (x + y). ^2 ;
>> plot3 (x , y ,z) ;
```

بردار x شامل ۷ عنصر و بردار y شامل ۵ عنصر می باشد. لذا تابع mesh grid ، ۳۵ زوج مرتب می سازد ، سپس تابع z روی این زوج های مرتب عمل می کند. بنابراین z شامل ۳۵ عنصر می باشد.

تابع mesh نیز برای رسم رویه های سه بعدی به صورت mesh (x , y , z) استفاده می شود. توابع دیگر که برای رسم نمودارهای سه بعدی استفاده می شوند به صورت زیر می باشند.

contour : نمودار تراز دو بعدی. به عبارت دیگر نمودار تراز سه بعدی که از بالا دیده می شود.

contour3 : نمودار تراز.

fill3 : چند ضلعی های تو پر.

meshc : نمودار شبکه ای که در قسمت زیرین آن تراز قرار دارد.

meshz : نمودار شبکه ای با صفحه ی صفر.

pcolor : نمودار شبه رنگ دو بعدی. به بیان دیگر نمودار رویه، که از بالا دیده می شود .

quiver : نمودار دوبعدی با فلش سرعت .

surf : نمودار رویه .

surfc : نمودار رویه ، که در زیر آن نمودار تراز قرار دارد .

surfl : نمودار رویه با روشنایی .

waterfall : نمودار شبکه ای بدون خطوط عرضی .

دقت شود قبل از اجرای این توابع باید به کمک تابع mesh grid نقاط (x , y , z) تولید شوند.

محاسبات نمادین(سمبلیک)

مجموعه ای از توابع که برای دستکاری و حل عبارات نمادین مانند ترکیب کردن،ساده کردن،دیفرانسیل گیری،انتگرال گیری و حل معادلات جبری و معادلات دیفرانسیل به کار می روند را در این قسمت بررسی می کنیم .

استخراج صورت و مخرج یک کسر :

با استفاده از تابع numden این کار انجام می شود. قرار می دهیم :

[n , d] = numden

به این ترتیب ، صورت عبارت در متغییر n و مخرج در متغییر d ذخیره می شود.

```
>> f = ' a * x^2/ (b – x) ' ;

>> [ n , d ] = numden (f)

    n = a * x ^ 2

    d = b – x
```

برای عباراتی که به صورت آرایه ی نمادین می باشند نیز این تابع استفاده می شود. به این طریق دو آرایه ی جدید n و d تولید می شود، که در آرایه ی n صورت ها و در d مخرج ها قرار می گیرد.

```
>> k = sym ( '[ 3/2 , (2 * x+1)/3 ; 4/x^2 , 3*x+4 ] ' )

K =

[ 3/2          (2 * x + 1 )/3]
[4/x^2           3 * x + 4   ]


>> [ n , d ] = numden (k)

n =

[3   2 * x + 1 ]
[4    3 * x + 4 ]

d =

]][ 2      3]
[x^2       1]
```

دقت شود در این مثال برای نمایش ماتریسی ، تابع sym لازم می باشد. اگر بدون این تابع بیان شود به صورت یک رشته ی کارکتری ذخیره می شود.

```
>> k = ( '[3/2 , (2*x+1)/3 ; 4/x^2 , 3*x+4] ' )

   k =   [ 3/2 , (2*x+1)/3 ;  4/x^2  , 3*x+4 ]
```

عملیات جبری :

برای جمع دو چند جمله ای تابع **symadd** ، برای تفریق تابع **symsub** ، برای ضرب تابع **symmul** و برای تقسیم تابع **symdiv** استفاده می شود. مثلا اگر $f = 2x^2 + 3x - 5$, $g = x^2 - x + 7$ داریم :

```
>> f = ′ 2 * x ^ 2 + 3 * x – 5 ′ ;

>> g = ′ x ^ 2 – x + 7′;

>> symadd (f , g)

    ans =   3 * x ^ 2 + 2 * x +2
```

از تابع **sym pow** برای به توان رساندن استفاده می شود.

```
>> sympow ( f , ′ 3 ′)

    ans = ( 2 * x^2 + 3 * x – 5)^3
```

تابع همه منظوره ی دیگری به نام **symop** وجود دارد که امکان می دهد، عبارتهای جدید با استفاده از متغییرها ، عبارتها و عملگرهای نمادین ایجاد شود. این تابع آرگومانها را متصل می کند و عبارت نتیجه را بر می گرداند.

```
>> f = ′ cos (x) ′ ;

>> g = ′ sin (2 * x) ′ ;

>> sympow (f , ′ / ′ , g , ′ + ′ , 3)

    ans =  cos (x) / sin (2*x) +3
```

این تابع می تواند روی آرایه ها نیز عمل کند.

توابع پیشرفته :

تابع **compose** برای ترکیب ، تابع **finverse** برای محاسبه ی معکوس تابع و تابع **sym sum** مجموع نمادین عبارت را محاسبه می کند.

```
>> f = ′ 1 / (1 + x^2) ′ ;

>> g =′ sin (x) ;

>> compose (f , g) ;

    ans = 1/ (1 + sin (x)^2)
```

>> finverse (f) ;

تابع sym sum چهار فرم دارد.

(a و b عدد می باشند)

حالت چهارم برای مواقعی به کار می رود که تابع چند متغییر مستقل داشته باشد. در این حالت متغییر مستقل حرف s در نظر گرفته می شود. برای محاسبه ی $\sum_1^n (2n - 1)^2$ داریم :

>> sym sum (′ (2*n – 1) ^ 2′ , 1 , ′n′)

ans = 11 / 3*n +8 / 3 – 4 * (n + 1) ^ 2+4 / 3 * (n + 1) ^ 3

>> factor (ans)

ans = -1/5 * x + 3/50 * x ^2 – 41 / 750 * x ^ 3 + 293 / 7500 * x ^ 4 – 1207 / 37500 * x ^ 5 +o(x^6)

>> pretty (f)

f =

عبارتهای نمادین را می توان در فرمهای مختلفی نمایش داد که در وضعیت های مختلف ممکن است برخی از فرمها نسبت به بقیه ارجح تر باشد.

```
>> f = ′ ( x ^ 2 – 1 )*(x – 2) * (x – 3) ′ ;
>> collect (f)
    ans =  x ^ 4 – 5 * x ^ 3 + 5 * x ^ 2 – 6
```

این دستور با جمع عبارات مشابه ، فرم ساده شده را می دهد.

```
>> horner (ans)
    Ans =  -6 +(5+(5+(-5+x)*x)*x)*x
```

عبارت ans را به صورت هورنر نمایش می دهد.

```
>> f = sym (′ [ a*x , b*x^2 ; c*x^3 , d*s] ′) ;
>> diff (f)
    Ans
```

$$\begin{bmatrix} a & 2 * b * x \\ 3 * c * x\text{^}2 & 0 \end{bmatrix}$$

انتگرال گیری :

تابع int(f) سعی می کند عبارت نمادین دیگری مانند F پیدا کند که diff (F) .

```
>> int (′Log(x) / exp(x^2) ′)
    ans = int ( Log(x) / exp(x^2) , x)
```

به عبارت دیگر matlab قادر به محاسبه ی انتگرال این تابع نمی باشد.

```
>> f = ′ sin (s+2*x) ′ ;
>> int (f)
    ans = -1/2 * cos (s + 2 + x)
```

>> int (f , ′s′)

ans = -cos (s + 2 * x)

انتگرال تابع f نسبت به متغییر s گرفته می شود.

>> int (f , pi/2 , pi)

ans = - cos (s)

این دستور انتگرال معین را در فاصله ی[π/2,π] محاسبه می کند.

>> int (f , ′s′ , pi/2 , pi)

ans = cos (2*x) – sin (2*x)

این دستور انتگرال تابع f را در فاصله ی [π/2,π] با توجه به اینکه s متغییر مستقل می باشد،محاسبه می کند.

>> int (f , ′m′ , ′n′)

ans = -1/2 * cos (s+2*n) + ½ * cos (s+2*m)

این دستور انتگرال معین تابع f نسبت به متغییر x را در فصله ی [m,n] محاسبه می کند.

>> factor (ans)

ans = (x-1)*(x-2)*(x-3)*(x+1)

این دستور عبارت ans را تجزیه می کند.

>> expand(ans)

ans = x^4 – 5 * x ^ 3 +5* x ^ 2 + 5 * x -6

تابع simplify یکی از ابزار قوی و همه منظوره است که عبارت را توسط انواع بسیاری از اتحادهای جبری شامل مجموع عبارتها ، توانهای صحیح و کسری ، توابع مثلثاتی ، توابع نمایی ، توابع لگاریتم ، توابع بسل ، توابع فوق هندسی و توابع گاما ، ساده می کند.

>> simplify (′Log(2*x/y) ′)

ans = Log2+Logx-Logy

>> simplify (′(-a^2+1)/(1-a) ′)

ans = a+1

تابع simplify نسبت به تابع simple رسمیت کمتری دارد. تابع simple، توابع ساده سازی مختلف را امتحان می کند، سپس فرمی که کمترین تعداد کارکتر را در عبارت نتیجه ارائه می دهد انتخاب می کند.

>> f = ′ (1 / x^3 + 6 /x^2 + 12/x+8) ^ (1/3) ′

هدف ساده کردن عبارت ریاضی زیر می باشد.

$$f = \sqrt[3]{1/x^3 6/x^2 + 12/x + 8}$$

>> simple (f)

ans = (2 * x + 1) / x

برای امتحان اینکه آیا عبارت ساده تر می شود یا خیر دوباره تابع simple را به کار می بریم.

>> simple (ans)

ans = 2 + 1 / x

حل یک معادله ی جبری :

برای این کار matlab از تابع solve استفاده می کند.

>> solve (′ a * x ^ 2 + b * x + c′)

هدف حل معدله ی $ax^2+bx+c=0$ می باشد.

ans =

[1/2/a*(-b+(b^2-4*a*c)^(1/2))]

[1/2/a*(-b+(b^2-4*a*c)^(1/2))]

اگر بخواهیم معادله را بر مبنای متغییر دیگر ، غیر از متغییر پیش فرض x ، حل کنیم باید در تابع solve آن را مشخص کنیم.

>> solve (′ a * x^2 + b * x + c′ , ′b′)

این معادله نسبت به متغییر b حل می شود.

ans = -(a*x^2+c)/x

این تابع معادلات نمادینی را نیز که حاوی علامت تساوی باشند می تواند حل کند.

```
>> solve (′cos(x)=sin(x) ′)

    ans = 1/4 * pi

>> t = solve (′tan(2*x)=sin(x) ′)

    t =

        [    0     ]

        [ a cos (1/2+1/2*3^(1/2)) ]

        [ a cos (1/2-1/2*3^(1/2))  ]
```

برای مشاهده مقادیر عددی قرار می دهیم.

```
>> numeric (t)

    ans = 0

            0+0.8314 i

              1.9455
```

حل چند معادله ی جبری :

چند معادله ی جبری را می توان همزمان حل کرد. دستور **solve (s1, s2,...., sn)** ، **n** معادله بر حسب متغییرهای پیش فرض حل می کند. دستور **solve(s1, s2,...., sn ,′x1,x2,....,xn′)** ، **n** معادله ی نمادین را برای **n** مجهول **x1،x2،.....xn** حل می کند.

فرض کنید می خواهیم مجموعه معادلات نمادین زیر را حل کنیم.

در این صورت قرار می دهیم

```
>> eq1 = ′d + n + p / 2 = q′ ;

>> eq2 = ′P = n + d + q - 10′ ;

>> eq3 = ′ q + d = p + n / 4 ′ ;

>> eq4 = ′ q + p = n + 8d – 1 ;

>> [ x , y , z , t ] = solve (eq1 , eq2 , eq3 , eq4 , ′p , n , d , q′)

   x =  16          y = 8          z = 3          t = 15
```

که مقدار **p** ، **y** مقدار **n** ، **z** مقدار **d** و **t** مقدار **q** می باشد.

حل یک معادله ی دیفرانسیل :

تابع **d solve** برای حل یک معادله ی دیفرانسیل به کار می رود. با توجه به اینکه حرف **D** معرف مشتق اول ، حرف **D2** مشتق دوم و الی آخر می باشد. فرض کنید می خواهیم معادله ی دیفرانسیل

را حل می کنیم. می نویسیم

```
>> d solve (′ Dy = 1 + y ^ 2′)

   ans =  - tan ( - x + c1)
```

که **c1** ثابت انتگرال گیری می باشد. با شرط اولیه **y(0) = 1** داریم

```
>> y = d solve (′Dy = 1 + y ^ 2 ′ , ′ y (0) = 1′)

   y = tan (x + ¼ * pi)
```

متغییر مستقل را می توان مشخص کرد.

```
>> d solve (′Dy = 1 + y ^2 ′ , ′ y (0) = 1 ′ , ′z′ )

   ans = tan (z + ¼ * pi)
```

فرض کنید می خواهیم معادله ی مرتبه ی دوم زیر را حل کنیم.

$$\frac{d^2y}{\mathrm{dx}^2} = cos(2x) - y \ , \qquad \mathrm{y}'(0) = 0 \quad , \quad \mathrm{y}(0) = 1$$

```
>> y = d solve (‘ D2y = cos (2*x) – y ‘   ,  ‘Dy (0) = 0’        ,       ‘y (0) = 1 ‘ )

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

 >> y = simple (y)

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

حل چند معادله ی دیفرانسیل :

تابع **d solve** می تواند چند معادله ی دیفرانسیل را نیز همزمان حل کند. فرض کنید

```
>> [ f , g] = d solve ( ‘ Df = 3 * f + 4 * g’  , ‘Dg = -4 * f + 3 * g ‘)

    f = c1 * exp (3*x) * sin (4*x) + c2 * exp (3*x) * cos (4*x)

    g = -c2 * exp (3*x) * sin (4*x) + c1 * exp (3*x) * cos (4*x)
```

با افزودن شرایط اولیه ی **f(0) =0** و **g(0) = 1** داریم :

```
>> [ f , g] = d solve ( ‘ Df = 3 * f + 4 * g’  , ‘Dg = -4 * f + 3 * g ‘   , ‘ f (0) = 0  ,  g(0) =
1’)

    f = exp (3*x) * sin (4*x)

    g = exp (3*x) * cos(4*x)
```

ماتریس های نمادین :

این نوع ماتریسها را از قبل می دانیم که توسط تابع **sym** می توان تولید کرد.

```
>> A = sym (‘ [ cos (t) , sin (t) ;  - sin (t) , cos (t) ] ‘ )

    A =

        [ cos (t)   ,   sin (t) ]

        [ -sin (t)   ,   cos(t) ]
```

تابع **sym** را می توان به فرمولی که عناصر مجزا را مشخص می کند، بسط داد ، با توجه به اینکه **i** و **j** موقعیت سطر و ستون ماتریس می باشند. فرض کنید می خواهیم ماتریس ۳×۳ تولید کنیم که a_{ij} = **(i + j)/(i – j + s)** ، که **s** یک ثابت دلخواه می باشد.

```
>> A = sym (3 , 3 ,'( i + j ) / ( i – j + s ) ' )

   A =

        [ 2/s , 3/ ( -1 + s ) , 4/ ( -2 + s ) ]

        [ 3/ ( 1 + s ) , 4/s , 5/( -1 + s) ]

        [ 4/ ( 2 + s ) , 5/ ( 1 + s) , 6/s]
```

برای دسترسی به عنصر **(m , n)** ام ماتریس نمادین **A** از دستور **sum (A , m , n)** استفاده می شود.

```
>> sym (A , 2 , 2)

   ans = 4/s
```

همچنین تابع **sym** برای تغییر عنصر آرایه ی نمادین استفاده می شود. دستور **sym (A , m ,n , ' x ')** باعث می شود که عنصر **(m , n)** ام ماتریس **A** ، به **x** تبدیل شود.

عملیات جبری :

با استفاده از تابع **sym add** می توان برای جمع ماتریس های نمادین و یا جمع یک عبارت نمادین به تمام عناصر یک ماتریس استفاده کرد. مثلا دستور **sym add(A , ' t ')** باعث می شود. که حرف **t** به تمام عناصر ماتریس **A** اضافه شود.تابع **sym sub** برای تفریق ، تابع **sym mul** برای ضرب و تابع **sym div** برای تقسیم استفاده می شود. توسط تابع **sym pow** توان و تابع **transpose** برای محاسبه ی ترانهاده ی ماتریس نمادین استفاده می شود.

وارون و دترمینال ماتریسهای نمادین توسط توابع **inverse** و **determ** محاسبه می شوند. ماتریس هیلبرت ۳×۳ را در نظر می گیریم.

```
>> H = sym ( hilb (3))

   H = [ 1 , 1/2 , 1/3 ]

       [ 1/2 , 1/3 , 1/4 ]
```

[1/3 , 1/4 , 1/5]

>> determ (H)

ans = 1.2160

>> j = inverse (H)

j = [9 , -36 , 30]

[-36 , 192 , -180]

[30 , -180 , 180]

تابع **Linsolve** برای حل همزمان معادلات خطی به کار می رود. **Linsolve (A , B)** ، معادله ی ماتریسی **AX = B** راحل می کند که **X** یک ماتریس مربعی می باشد. جواب **Linsolve(A,I) (AX = I)** وارون ماتریس **A** می باشد($X = A^{-1}$).

تابع **charpoly** چند جمله ای مشخصه ی یک ماتریس را پیدا می کند.

>> A = sym (' [1 , 1/2 ; 1/3 , 1/4] ')

A =

[1 , 1/2]

[1/3 , 1/4]

>> charpoly (A)

ans = x ^ 2 – 5 / 4 * x + 1 / 12

مقادیر ویژه و بردارهای ویژه ماتریس های نمادین توسط تابع **eigensys** محاسبه می شود.

>> A = sym (' [1/2 , 1/4 ; 1/4 , 1/2] ') ;

>> eigensys (A)

ans =

[3/4]

[1/4]

>> [V , E] = eigensys (A)

V = [-1 , 1]

[1 , 1]

E = [1/4]

[3/4]

در این صورت $\begin{bmatrix}-1\\1\end{bmatrix}$ بردار متناظر با مقدار ویژه ی ۱/۴ و $\begin{bmatrix}1\\1\end{bmatrix}$ بردار متناظر با مقدار ویژه ی ۳/۴ برای ماتریس A می باشند.

برای محاسبه ی مقادیر تکین ماتریس A از تابع singvals استفاده می کنیم.

اکنون به مثال هایی چند می پردازیم:

انتگرال گیری عددی:

$$h = \frac{b - a}{n}$$

$$x_0 = \mathrm{a}\ و\ x_n = \mathrm{b}$$

$$x_{i+1} = x_i + \mathrm{h} \rightarrow exampel: x_1 = x_0 + h$$

$$\int_a^b \mathrm{f(x)}\,dx = h * (f(x_0)/2 + f(x_1) + f(x_2) + \cdots + f(x_n)/2)$$ فرمول ذوزنقه

$$\int_a^b \mathrm{f(x)}\,dx = \left(\frac{h}{3}\right)\left(f(x_0) + 4 * f(x_1) + 2 * f(x_2) + \cdots + f(x_n)\right)$$ فرمول سیمسون

سوال:انتگرال زیر را با هر دو روش محاسبه کنید؟

$$\int_0^2 \mathrm{x} * \mathrm{e}^{\mathrm{x}}\mathrm{dx}$$

قاعده ذوزنقه

تابع ثابت و داده های عددی متغیر فرض شده است:

```
clc % is function = x.e^x
clear
format short e
s=0;
a=input('interval start =?');
b=input('interval end =?');
n=input(' add step = ?');
h=(b-a)/n;
for i=0:n
    if i==0&n
       s=s+((i*h)*exp(i*h)/2);
    else
       s=s+(((i*h)*exp(i*h)));
    end
end
s=(h)*s;
disp(s)
return
```

حل مساله(جواب واقعی (۸,۳۸۹۰۵۶۰۹۸) جواب با درایو شده است.)

interval start =?0

interval end =?2

add step = ?100

۸,۵۳۷۵e+000

****

قاعده سیمسون

```
clc % is function = x.e^x
clear
format short e
s=0;
a=input('interval start =?');
b=input('interval end =?');
n=input(' add step = ?');
h=(b-a)/n;
for i=0:n
    if i==0&n
       s=s+((i*h)*exp(i*h));
    elseif (i/2)~=floor(i/2)
       s=s+(4*((i*h)*exp(i*h)));
    else
      s=s+(2*((i*h)*exp(i*h)));
    end
end
s=(h/3)*s;
disp(s)
return
```

حل مساله

interval start =?0

interval end =?2

add step = ?100

۸,۴۸۷۶ e+000

****

حال با تلبع آن ها می نویسیم:

فایل جدا صدا زده شود. mتابع بصورت

تابعdist

```
function g=dist(h)
s=0;
s=h*exp(h);
g=s;
```

برنامه اصلی

```
clc % is function = x.e^x
clear
format short e
s=0;
a=input('interval start =?');
b=input('interval end =?');
n=input(' add step = ?');
h=(b-a)/n;
for i=0:n
    if i==0&n
       s=s+(dist(i*h));
    elseif (i/2)~=floor(i/2)
       s=s+(4*dist(i*h));
    else
      s=s+(2*dist(i*h));
    end
end
s=(h/3)*s;
disp(s)
return
```

اجرا

interval start =?0

interval end =?2

add step = ?100

۸,۴۸۷۶ e+000

****

حال با تلبع آنرا می نویسیم:سیمسون

تابع در دل برنامه نوشته شود و داده ومتغیر هر دو متغیر:

```
clc % is function = x.e^x
clear
format short e
s=0;
fun=input('input function = ','s');
f=sym(fun);
a=input('interval start =?');
b=input('interval end =?');
n=input(' add step = ?');
h=(b-a)/n;
for i=0:n
    if i==0&n
       s=s+(subs(f,i*h));
    elseif (i/2)~=floor(i/2)
       s=s+(4*subs(f,i*h));
    else
      s=s+(2*subs(f,i*h));
    end
end
s=(h/3)*s;
disp(s)
return
```

اجرا

```
input function = x*exp(x)
    interval start =?0
     interval end =?2
      add step = ?100
        8.4876e+000
```

****

تابع فاکتوریل

```
function d=fact(n)
s=1;
```

```
for i=1:n
    s=s*i;
end
disp(s)
```

اجرا

fact(10(

۳۶۲۸۸۰۰

****

مجموع اعداد از n تا m

```
clc
clear
n=input('enter min number n = ?');
m=input('enter max number m = ?');
s=0;
for i=n:m
    s=s+i;
end
d=['majoeh adad = ',num2str(s),' ok'];
disp(d)
return
```

اجرای آن:

enter min number n = ?10

enter max number m = ?100

majoeh adad = 5005 ok

مشخص کردن انواع مثلث

```
clc
clear
a=input('enter a : ');
b=input('enter b : ');
```

```
c=input('enter c : ');
s=0;
if (a==b)&(b==c)&(b==c)
    d=['motesaviol azla'];
    disp(d)
elseif (a==c)|(a==b)|(b==c)
    d=['motesaviol saghin'];
    disp(d)
elseif (a^2==b^2+c^2)|(b^2==a^2+c^2)|(c^2==b^2+a^2)
    d=['ghaemolzavieh'];
    disp(d)
else
    d=['mokhtaefol azla'];
    disp(d)
end
```

حل معادله درجه ۲

```
clc
clear
a=input('enter min number a = ?');
b=input('enter max number b = ?');
c=input('enter max number c = ?');
s=0;
d=b^2-4*a*c;
if a==0
    ['                              dgree eqation is 1
!!!!']
elseif d<0
    k=['                            no answer!!!!!!!!'];
    disp(k)
elseif d==0
    x1=(-b/(2*a));
    k=['                            roots = ',num2str(x1)];
    disp(k)
else
    x1=(-b+sqrt(d))/(2*a);
    x2=(-b-sqrt(d))/(2*a);
    k=['                          roots = ',num2str(x1),'
and ',num2str(x2)];
    disp(k)
end
```

```
return
```

اجرا

enter min number a = ?1

enter max number b = ?5

enter max number c = ?6

roots = -2 and -3

فاکتوریل

```
clc
clear
n=input('enter min number n = ?');
s=1;
if n<0
    d=[' no answer!!!!! azizim adad bayad + bashad.'];
    disp(d)
else
    for i=1:n
    s=s*i;
    end
d=[' factoril adad = ',num2str(s),' ok'];
disp(d)
end
return
```

اجرا

enter min number n = ?10

factoril adad = 3628800 ok

برنامه اجرای $S = \frac{1}{2} - \frac{3}{4} + \frac{5}{6} \cdots \frac{n}{n+1}$

```
clc
clear
n=input('enter min number n = ?');
s=0;
for i=1:n
    s=s+(-1)^(i+1)*(i/(i+1));
end
```

```
d=['majoeh adad = ',num2str(s),' ok'];
disp(d)
return
```

اجرا

enter min number n = ?10

majoeh adad = -0.26346 ok

حلقه ها

دستور While

```
clc
clear
n=input('enter n =? ');
i=0;
while i<n
    i=i+1 ;

   disp(i)
end
 return
```

دستور switch

```
clc
clear
d=input('enter d , if be 0<=d<=11 : ')
switch d
    case {2,8}
        disp('emame aval');
    case {1,7}
        disp('markaze ostane golestan?');
    case {3,9}
      disp('markaze ostane hormozgan?');
    case {4,5}
      disp('hesab kon 12*15=?');
    case {6}
     disp('4*2/3-1?');
    case {10,11}
     disp('pitakhte tiland?');
    otherwise
        disp('bye bye');
end
```

ویا

```
clc
clear
a=input('enter a : ');
b=input('enter b : ');
c=input('enter c : ');
s=0;
n=input('enter 1=+ 2=- 3=* 4=/ =? ');
switch n
    case {1}
        s=a+b+c;
        disp(s);
    case {2}
        s=a-b-c;
        disp(s);
    case {3}
      s=a*b*c;
        disp(s);
    case {4}
      s=a/b/c;
      disp(s);
    otherwise
        disp('error bye bye');
end
```

انواع مثلث

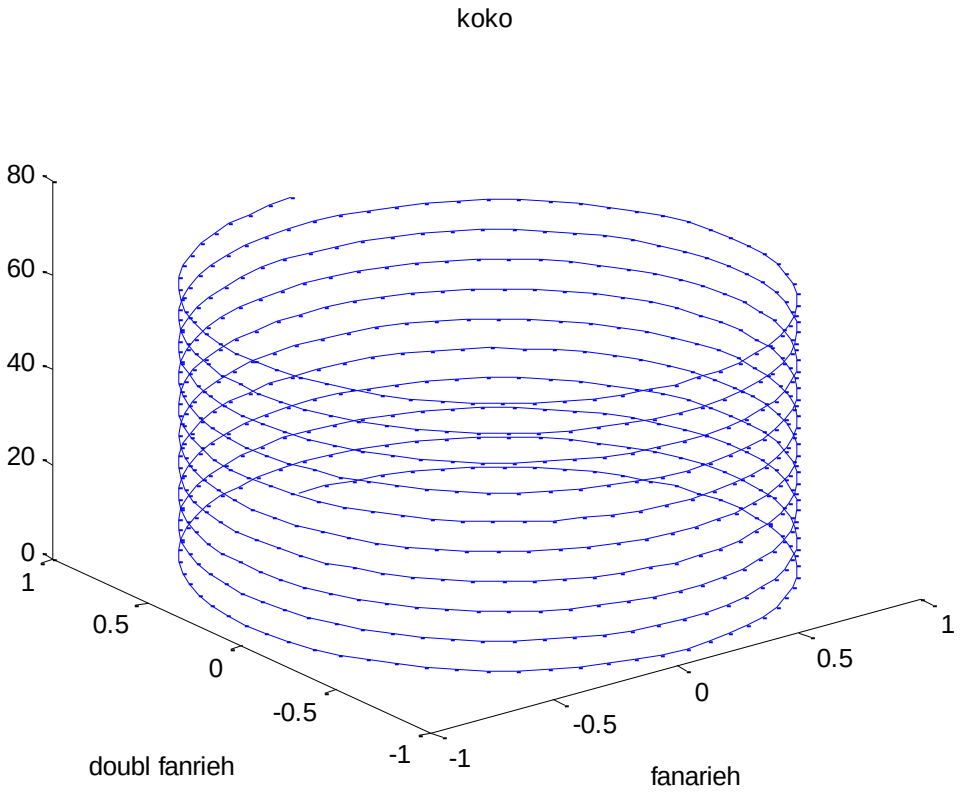
```
clc
clear
a=input('enter a : ');
b=input('enter b : ');
c=input('enter c : ');
s=0;
if (a==b)&(b==c)
    d=['motesaviol azla'];
    disp(d)
elseif (a==c)|(a==b)|(b==c)
    d=['motesaviol saghin'];
    disp(d)
elseif (a^2==b^2+c^2)|(b^2==a^2+c^2)|(c^2==b^2+a^2)
    d=['ghaemolzavieh'];
    disp(d)
else
    d=['mokhtaefol azla'];
    disp(d)
end
```

ایجاد منو

```
clc
clear
k=0;
while k~=3
    k=menu('click on your option','do this','do
that','quit');
    if k==1
        disp(' what is your name ?')
        pause
    elseif k==2
        disp('how do you do?!')
        pause
    end
end;
```

رسم فنر

```
clc
x=[0:.1:20*pi];
y=sin(x);
z=cos(x);
plot3(y,z,x,'r')
xlabel('ali')
ylabel('reza')
title('bahram')
```



جدول ضرب

```
clc
clear
for i=1:10
    for j=1:10

        A(i,j)=i*j;
    end
    disp(i)
end
```

اجرا

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۲	۴	۶	۸	۱۰	۱۲	۱۴	۱۶	۱۸	۲۰
۳	۶	۹	۱۲	۱۵	۱۸	۲۱	۲۴	۲۷	۳۰
۴	۸	۱۲	۱۶	۲۰	۲۴	۲۸	۳۲	۳۶	۴۰
۵	۱۰	۱۵	۲۰	۲۵	۳۰	۳۵	۴۰	۴۵	۵۰
۶	۱۲	۱۸	۲۴	۳۰	۳۶	۴۲	۴۸	۵۴	۶۰
۷	۱۴	۲۱	۲۸	۳۵	۴۲	۴۹	۵۶	۶۳	۷۰
۸	۱۶	۲۴	۳۲	۴۰	۴۸	۵۶	۶۴	۷۲	۸۰
۹	۱۸	۲۷	۳۶	۴۵	۵۴	۶۳	۷۲	۸۱	۹۰
۱۰	۲۰	۳۰	۴۰	۵۰	۶۰	۷۰	۸۰	۹۰	۱۰۰

تجزیه کسرها

$$\frac{10 * (s+2)}{(s+1)(s+3)(s+4)}$$

```
clc
clear
num=10*[1 2];
den=poly([-1;-3;-4])
[res,poles,k]=residue(num,den)
```

اجرا

صورت کسرها=res

-۶,۶۶۶۷

۵,۰۰۰۰

۱,۶۶۶۷

ریشه مخرج کسرها=poles

۴,۰۰۰۰-

۳,۰۰۰۰-

۱,۰۰۰۰-

=باقیماندهk

[]

دیفرانسیل چندجمله ای ها ی کسری

$$\frac{d}{\mathrm{ds}}\left\{\frac{10*(\mathrm{s}+2)}{(\mathrm{s}+1)(\mathrm{s}+3)(\mathrm{s}+4)}\right\}$$

```
clc
clear
num=10*[1 2];
den=poly([-1;-3;-4]);
[b,a]=polyder(num,den)
```

اجرا

=چندجمله ای صورتb

۲۶۰- ۳۲۰- ۱۴۰- ۲۰-

=چندجمله ای مخرجa

۱۴۴ ۴۵۶ ۵۵۳ ۳۲۸ ۱۰۲ ۱۶ ۱

اعدا د اول بین دو عدد

```
clc
clear
range=input('range=? [a b]');
x1=range(1);
x2=range(2);
a=1;
for j=x1:x2
    s=0;
    for i=1:j-1
        r=j-fix(j/i)*i;
        if r==0
            s=s+i;
```

```
        end
    end
    if s==1
        nums(a)=j;
        a=a+1;
    end
end
disp('adade aval=')
disp(nums)
```

اجرا

range=? [a b][5 30[

adade aval=

۲۹ ۲۳ ۱۹ ۱۷ ۱۳ ۱۱ ۷ ۵

تابع اندازه

```
function u=dist(x1,y1,x2,y2)
dx=x2-x1;
dy=y2-y1;
u=sqrt(dx^2+dy^2);
```

اجرا

dist(2,1,4,5(

ans=

۴,۴۷۲۱

تابع بالا با دو خروجی

```
function [u,v]=dist2(x1,y1,x2,y2)
dx=x2-x1;
dy=y2-y1;
u=sqrt(dx^2+dy^2);
v=atan(dy/dx);
```

اجرا

]a r]=dist2(1,5,4,8(

a=

۴,۲۴۲۶ e+000

r=

۷,۸۵۴۰ e-001