فصل ۴

رسم نمودارها

۱.۴ نمودارهای دوبعدی

۱.۱.۴ رسم یک نمودار ساده

با استفاده از تابع plot می‌توان یک شکل ساده دوبعدی که از اتصال نقطه به نقطه مختصات تشکیل شده است را رسم نمود، به‌عنوان مثال دستورات

```
>> x=1:10;
>> y=sin(x);
>> plot(x,y)
```

نموداری را تولید می‌کنند که از به هم پیوستن نقاط $(x(i), y(i))$ ایجاد شده است، همزمان با اجرای دستور plot پنجره‌ای باز می‌شود که در آن نمودار مورد نظر رسم شده است. در حالت کلی می‌توان به‌جای استفاده از $plot(x, y)$ از $plot(x, y, 'string')$ استفاده نمود که در آن $'string'$ سه مشخصه نمودار مورد نظر یعنی رنگ، نشان و قلم خط را مشخص می‌کند. به‌عنوان مثال دستور

```
>> plot(x,y,'r \ast --')
```

بیان می‌کند که نقاط $(x(i), y(i))$ با ستاره قرمز رنگ رسم و با خط چین قرمز رنگ به هم متصل شوند. همچنین $plot(x, y, 'y+')$ بیان می‌کند که نقاط $(x(i), y(i))$ با علامت $+$ زرد رنگ و بدون این‌که به هم متصل گردند نشان داده شوند.

نکته. سه مشخصه در $'string'$ به هر ترتیبی می‌توانند قرار گیرند به عنوان مثال دو دستور

```
>> plot(x,y,'ms--')
>> plot(x,y,'s--m')
```

با هم معادل خواهند بود. در جدول ذیل فهرستی از گزینه‌های مختلف برای مشخصه یک نمودار آورده شده است

نشان	
دایره	o
ستاره	*
نقطه	.
جمع	+
ضرب	×
مربع	s
لوزی	d
مثلث رو به بالا	^
مثلث رو به پایین	v
مثلث رو به راست	>
مثلث رو به چپ	<
ستاره پنج گوش	p
ستاره شش گوش	h

رنگ	
قرمز	r
سبز	g
آبی	b
آبی تیره	b
سرخ	m
زرد	y
سیاه	k
سفید	w

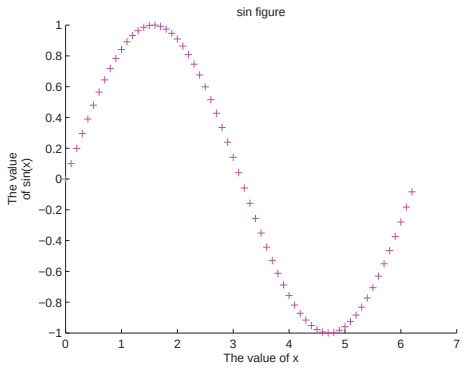
سبک خط	
خط	−
خط چین	− −
نقطه چین	:
خط و نقطه	−.

مثال ۱.۱.۴. کد ذیل را در نظر بگیرید

```
x=0:.1:2*pi;
y=sin(x);
plot(x,y,'m+')
xlabel('The value of x')
ylabel({'The value','of sin(x)'})
title('sin figure')
box off
```

در این مثال title نام نمودار می‌باشد که در بالای نمودار به‌صورت رشته چاپ می‌شود، همچنین خروجی توابع xlabel و ylabel نیز به‌صورت رشته می‌باشد که برای نامگذاری محورها به‌کار می‌رود. دستور box off جعبه ایجاد شده در اطراف نمودار را حذف نموده و فقط محورها نمایش داده می‌شوند (شکل ۱.۴).

نکته. اگر دستور رسم نموداری بعد از دستور رسم نمودار دیگری بیاید، شکل جدید جایگزین شکل قبل از آن خواهد شد، برای ایجاد شکل جدید و شکل قبلی در یک صفحه می‌توان از



شکل ۱.۴

دستور hold on استفاده نمود، دستور hold off نیز ایجاب می‌کند که هر شکل در یک پنجره مجزا رسم شود.

مثال ۲.۱.۴. نمودار ایجاد شده توسط دستورات

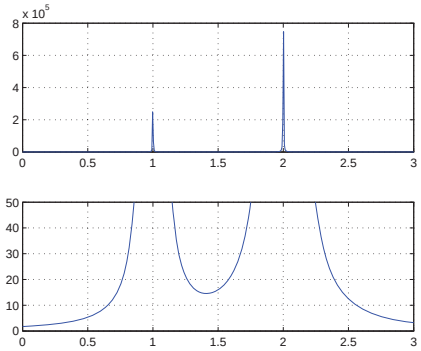
```
>> x = linspace(0,3,500);
>> plot(x,1./(x-1).^2 + 3./(x-2).^2)
>> grid on
```

را در نظر بگیرید، استفاده از دستور grid on باعث می‌شود که صفحه نمایش ما به‌صورت شبکه‌ای نمایش داده شود (نمودار بالا در شکل ۲.۴)، توجه داریم که نمودار ما در نقاط ۱ و ۲ ناپیوستگی دارد با اضافه نمودن دستور ylim([ymin ymax]) یا xlim([xmin xmax]) می‌توان محور x یا y را محدود نمود و نمودار را به‌صورت جزئی رسم و مشاهده نمود، به‌عنوان مثال با اضافه نمودن دستور ylim([۰ ۵۰]) نمودار سمت پایین شکل ۲.۴ نمایش داده خواهد شد که محور y ها را محدود نموده و جزئی از نمودار را نشان می‌دهد.

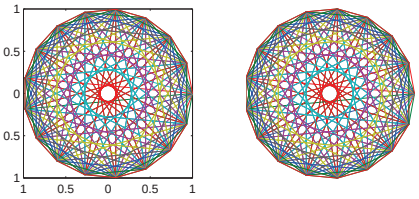
مثال ۳.۱.۴. دستورات

```
>> plot(fft(eye(17))), axis equal, axis square
>> plot(fft(eye(17))), axis equal, axis off
```

را در نظر بگیرید، استفاده از دستور axis off باعث می‌شود محورها چاپ نشوند(شکل ۳.۴ سمت راست)، همچنین دستور axis([xmin xmax ymin ymax]) باعث می‌شود که محور x ها از xmin تا xmax و محور y ها از ymin تا ymax نمایش داده شوند، برای برگشت به



شکل ۲.۴



شکل ۳.۴ استفاده از axis off

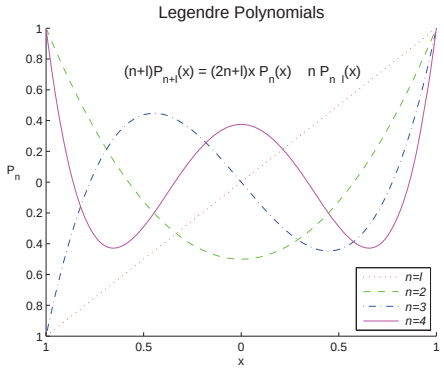
حالت پیش فرض می‌توان از دستور axis outo استفاده نمود، برای مطالعه بیشتر به راهنمای نرم افزار مراجعه نمایید.

مثال ۴.۱.۴. با استفاده از دستورات

```
x = -1:.01:1;
p1=x;
p2=(3/2)*x.^2 - 1/2;
p3=(5/2)*x.^3 - (3/2)*x;
p4=(35/8)*x.^4 - (15/4)*x.^2 + 3/8;
plot(x,p1,'r:',x,p2,'g--',x,p3,'b-.',x,p4,'m-')
box off
legend('\it n=l','\it n=2','\it n=3','\it n=4','Location','
```

```
    SouthEast')
xlabel('x')
ylabel('P_n','Rotation',0)
title('Legendre Polynomials','FontSize',14)
text(-.6,.7,'(n+1)P_{n+1}(x) = (2n+1)x P_n(x) - n P_{n-1}(x)',...
'FontSize',12)
```

نمودار چند جمله‌ای لژاندر از درجه ۱ تا ۴ در یک صفحه رسم شود، در این جا با استفاده از دستور legend جعبه‌ای در صفحه نمودار برای ما چاپ خواهد شد که در آن مشخصات نمودارهای رسم شده شرح داده می‌شوند(شکل ۴.۴). در حالت کلی با نوشتن دستور



شکل ۴.۴

legend(′string۱′, ′string۲′, ..., ′stringn′)

یک جعبه علائم در کنار نمودار چاپ خواهد شد، مکان این جعبه به‌صورت پیش فرض در سمت راست و بالای نمودار می‌باشد که مانند مثال قابل تغییر می‌باشد. مکان این جعبه بعد از رسم نمودار نیز با استفاده از ماوس یا راست کلیک روی جعبه و انتخاب گزینه location قابل تغییر می‌باشد. در این مثال از دستور text نیز استفاده نموده‌ایم، در حالت کلی این دستور به‌صورت text(x, y,′ string′) می‌باشد که ′string′ را در مختصات (x, y) چاپ خواهد نمود. استفاده از تابع gtext به جای تابع text این امکان را فراهم می‌کند که بعد از رسم نمودار مختصات محل را با ماوس روی شکل انتخاب نماییم. دقت کنید که در توابع ylabel و text از نماد خاصی استفاده کرده‌ایم، این حروف چینی ازسیستم LaTeX یا TeX تبعیت می‌کند برخی از این دستورات را می‌توان در ضمیمه ب مشاهده نمود. اگر با حروف چینی LaTeX یا TeX آشنایی ندارید می‌توانید از دستور texlabel(′string′) که اجازه می‌دهد ′string′ به سبک نوشتاری متلب باشد استفاده نمایید. به‌عنوان مثال دو دستور

```
text(5,5,'\alpha^{3/2}+\beta^{12}-\sigma_i')
text(5,5,texlabel('alpha^(3/2)+beta^12-sigma_i'))
```

باهم معادل می‌باشند.

نسخه‌های ۷ به بعد متلب مفسر LaTeX می‌باشند که حروف چینی تحت LaTeX پشتیبانی می‌کنند، برای استفاده از این امکانات نیاز داریم ابتدا تنظیمات مربوطه را اعمال نماییم.

مثال ۵.۱.۴. (استفاده از حروف چینی تحت LaTeX)
دستورات

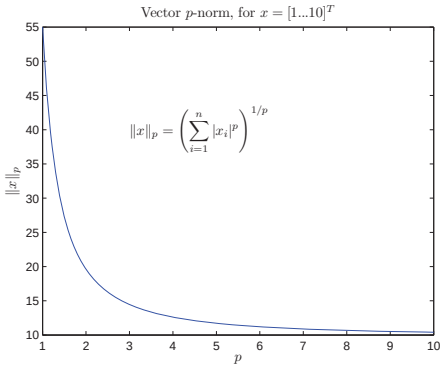
```
x = 1 :10;
y = zeros(100,1);
z = linspace(1,10,100);
for i = 1:100
y(i) = norm(x,z(i));
end
plot(z,y)
options = {'Interpreter','latex','FontSize',12};
ylabel('$\|x\|_p$',options{:})
xlabel('$p$',options{:})
title(' Vector $p$-norm, for $x =[1 ... 10]^T$', options{:})
text(3,40,'$$\|x\|_p = \left(\sum_{i=1}^n|x_i|^p\right)^{1/p}$$'
    ,...
options{:})
```

را در نظر بگیرید، نتیجه اجرای این دستورات شکل ۵.۴ می‌باشد که در آن از حروف چینی تحت LaTeX استفاده کرده‌ایم، توجه داریم که در این‌جا برای جلوگیری از تکرار، از آرایه سلولی options استفاده نموده‌ایم.

۲.۱.۴ رسم چند نمودار در یک صفحه

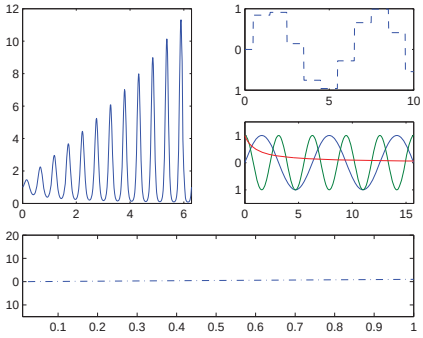
با استفاده از تابع subplot می‌توان چند نمودار را در یک صفحه رسم نمود، دستور subplot(m, n, p) یا معادل آن subplot(mnp) صفحه را به m در n ناحیه تقسیم می‌کند و p شماره ناحیه مورد نظر را نشان می‌دهد که شماره‌بندی نواحی به‌صورت سطر به سطر می‌باشد. به‌عنوان مثال دستور subplot(۴۲۵) صفحه را به هشت ناحیه تقسیم می‌کند و ناحیه مورد نظر ما پنجمین ناحیه می‌باشد.

مثال ۶.۱.۴. در مثال ذیل صفحه را به شش ناحیه تقسیم می‌کنیم و در نواحی مختلف نمودارهای خود را رسم می‌کنیم (شکل ۶.۴).



شکل ۵.۴

```
subplot(3,2,[1 3]), fplot('exp(sqrt(x)*sin(12*x))',[0 2*pi])
subplot(322), fplot('sin(round(x))',[0 10],'--')
subplot(324), fplot('[sin(x),cos(2*x),1/(1+x)]',[0 5*pi -1.5 1.5])
subplot(3,2,5:6), fplot('x',[0.01 1 -15 20],'-.')
```



شکل ۶.۴

در بخش‌های بعد با دیگر توابع مهم رسم در دوبعد آشنا خواهیم شد.

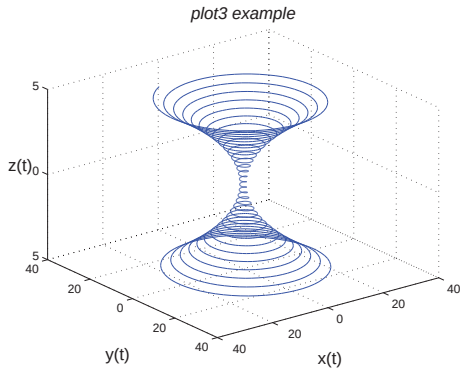
۲.۴ نمودارهای سه بعدی

تابع plot۳

این تابع حالت سه بعدی تابع plot می‌باشد و با عبور از نقاط (x(i), y(i), z(i)) یک منحنی سه بعدی رسم خواهد کرد، به‌عنوان مثال شکل ۷.۴ را می‌توان با دستورات

```
t = -5:.005:5;
x = (1+t.^2).*sin(20*t);
y = (1+t.^2).*cos(20*t);
z = t;
plot3(x,y,z)
grid on
FS = 'FontSize';
xlabel('x(t)',FS,14), ylabel('y(t)',FS,14)
zlabel('z(t)',FS,14,'Rotation',0)
title('\it{plot3 example}',FS,14)
```

رسم نمود. دراین مثال از توابع xlabel، ylabel و title که قبلا توضیح داده شده‌اند استفاده



شکل ۷.۴

نموده‌ایم و به‌طور مشابه تابع zlabel را نیز تعریف نموده‌ایم، رنگ، نشان و سبک خط در این تابع همانند تابع plot تعریف می‌گردد. توابع comet و comet۳ نیز مشابه توابع plot و plot۳ می‌باشند با این تفاوت که این توابع رسم نمودار را با یک نوک متحرک مرحله به مرحله به ما نمایش می‌دهند. با اجرای دستورات

```
t = -10*pi:pi/250:10*pi;
```

```
comet3((cos(2*t).^2).*sin(t),(sin(2*t).^2).*cos(t),t);
```

به تفاوت این توابع پی خواهید برد.

رسم یک رویه

بسیاری از توابع ریاضی دو متغیره می‌باشند و برای دو متغیر x و y یک متغیر z به‌صورت $z = f(x, y)$ را تولید می‌کنند، یک راه تولید z استفاده از حلقه‌های تودرتو می‌باشد. امکان این کار در متلب بدون استفاده از حلقه نیز وجود دارد. اگر شما یک بردار از مقادیر x و یک بردار از مقادیر y را داشته باشید، متلب تابع مفید meshgrid را برای تولید ماتریس‌های X و Y برای استفاده در ترسیم اشکال سه بعدی دارد، نحوه استفاده از این تابع به‌صورت

$$[X, Y] = \text{meshgrid}(x, y)$$

می‌باشد، ماتریس X از تکرار بردار x به عنوان سطرهای ماتریس و ماتریس Y از تکرار بردار y به‌عنوان ستون‌های ماتریس حاصل می‌شود، تجسم این مسئله در ابتدا اندکی پیچیده به نظر می‌رسد، مثال ذیل تجسم آن‌را راحت‌تر خواهد نمود.

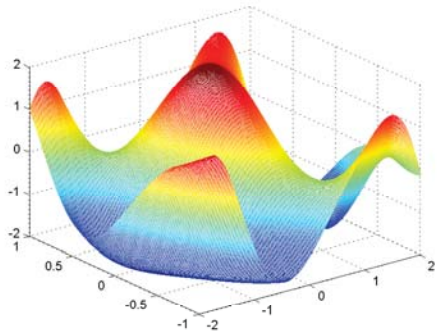
```
>> x=[ -1 0 1];
>> y=[9 10 11 12];
>>  [X,Y]=meshgrid(x,y)
X =
    -1     0     1
    -1     0     1
    -1     0     1
    -1     0     1

Y =
     9     9     9
    10    10    10
    11    11    11
    12    12    12
```

حال با استفاده از تابع mesh می‌توان یک رویه با استفاده از پارامترهای X، Y و Z رسم نمود. به‌عنوان مثال با اجرای دستورات

```
x = -2:.01:2; y = -1: .01:1;
```

```
[X,Y] = meshgrid(x,y);
Z = sin(3*Y-X.^2+1)+cos(2*Y.^2-2*X);
mesh(X,Y,Z)
```

شکل ۸.۴ را خواهیم داشت. تابع surf نیز عملکردی مشابه تابع mesh دارد با این تفاوت که



شکل ۸.۴

رویه را به‌صورت پررنگ‌آمیزی می‌کند. تفاوت این دو تابع را می‌توان با اجرای دستورات

```
[x y ] = meshgrid(-8 : 0.5 : 8);
r = sqrt(x.^2 + y.^2) + eps;
z = sin(r)./r;
subplot(1,2,1); mesh(z); title('(mesh function)')
subplot(1,2,2); surf(z); title('(surf function)' )
```

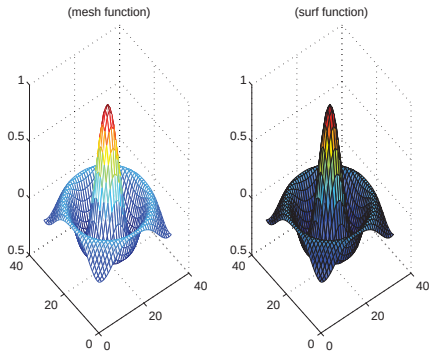
برای رسم نمودار کلاه مکزیکی مشاهده نمود (شکل ۹.۴).

۱.۲.۴ توابعی با کاربرد آسان

توابعی مانند ezplot، ezmesh و... را توابعی با کاربرد آسان می‌نامیم، استفاده از آن‌ها آسان بوده و تابع را روی بازه پیش‌فرض $[-2\pi,\ 2\pi]$ و یا $[-2\pi,\ 2\pi]\times[-2\pi,\ 2\pi]$ ترسیم می‌کنند.

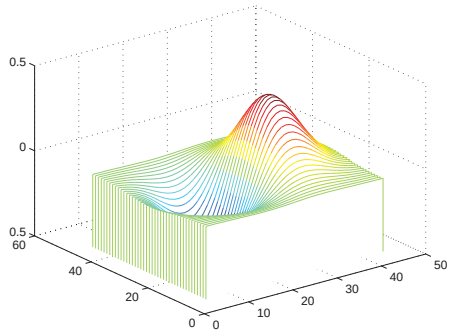
```
1) f = @(x,y) x.*exp(-x.^2-y.^2); ezmesh(f)
2) ezmash('x.*exp(-x.^2-y.^2)')
```

در ادامه چند نمونه از توابع جالب گرافیکی را به‌همراه چند مثال می‌آوریم، مطالعه جزئیات را به خواننده واگذار می‌کنیم:

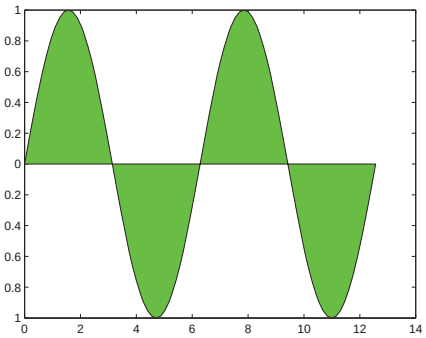


شکل ۹.۴ نمودار کلاه مکزیکی با توابع mesh و surf

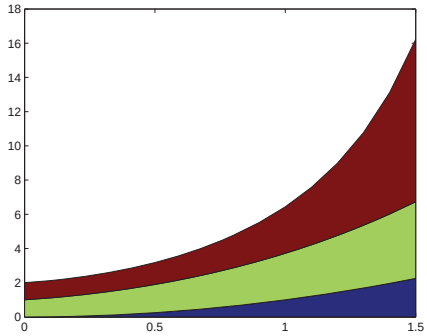
```
>> [x y] = meshgrid(-2:0.1:2); z = x.*exp(-x.^2-y.^2); waterfall(z)
>> t = 0:pi/20:4*pi; fill(t,sin(t),'g')
>> x = 0:0.1:1.5; area(x', [x.^2' exp(x)' exp(x.^2)'])
>> t = 0:pi/50:2*pi; r = exp(-0.05*t); stem3(r.*sin(t), r.*cos(t),t)
>> x = 0:pi/40:pi; stairs(x,sin(x))
>>[x y ] = meshgrid(-8 : 1 : 8); r = sqrt(x.^2 + y.^2) + eps;
>> z = sin(r) ./ r; ribbon(z)
```



شکل ۱۰.۴ تابع waterfall



شکل ۱۱.۴ تابع fill



شکل ۱۲.۴ تابع area

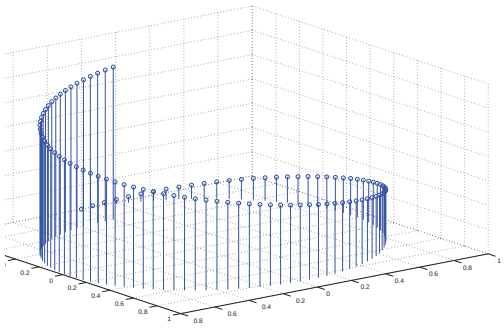
۳.۴ نمودارهای خاص برای نمایش داده‌ها

در این بخش به معرفی چند تابع مهم برای نمایش داده‌ها می‌پردازیم.

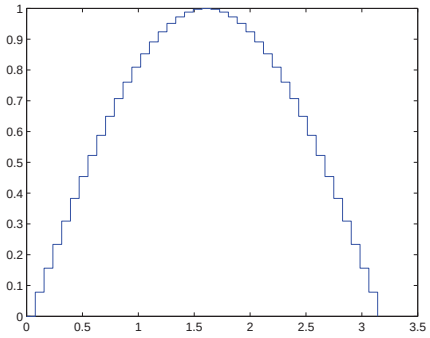
تابع hist

یک راه مهم برای نمایش داده‌ها استفاده از این تابع می‌باشد، برای آشنایی با این تابع مثال ذیل را مطرح می‌کنیم: فرض کنید ۱۲ دانشجو در یک امتحان شرکت کرده و نمرات آن‌ها در بردار a به‌صورت

```
a=[0  25  29  35  50  55  55  59  72  75  95  100]
```



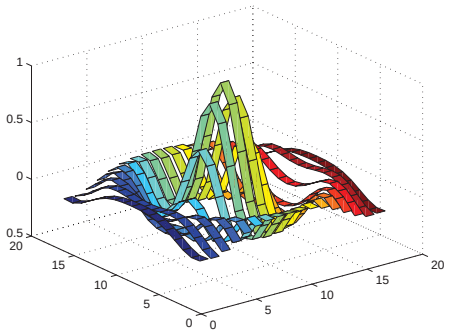
شکل ۱۳.۴ تابع stem3



شکل ۱۴.۴ تابع stairs

ثبت شده است. دستور hist(a) یک نمودار ستونی برای ما ترسیم خواهد نمود (شکل ۱۶.۴) که در آن توزیع داده‌ها را در ده رده تعیین می‌کند، این ده رده که به‌صورت پیش‌فرض تعیین می‌شود بین کمترین و بیشترین داده قرار می‌گیرند و با تخصیص آرگومان دوم به تابع hist می‌توان تعداد رده‌ها را تغییر داد، با اجرای دستور [n x] = hist(a) دو بردار x و n نمایش داده می‌شوند که در آن n تعداد فراوانی را در هر رده و x میانه هر رده را مشخص می‌کند. به‌عنوان مثال داریم

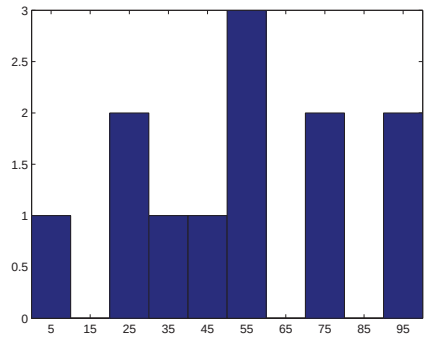
```
>> hist(a)
>> [n x]=hist(a)
n =
     1     0     2     1     1     3     0     2     0     2
```



شکل ۱۵.۴ تابع ribbon

```
x =
     5    15    25    35    45    55    65    75    85    95
```

که نتایج نشان می‌دهد در رده اول یعنی ۰-۹ یک نمره و در رده دوم یعنی ۱۰-۱۹ هیچ



شکل ۱۶.۴ تابع hist

نمره‌ای وجود ندارد.

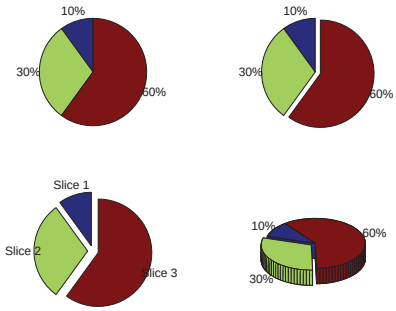
تابع pie

نموداری که توسط این تابع رسم می‌شود درصد شرکت یک عنصر از یک بردار یا یک ماتریس نسبت به شرکت کل عناصر را نمایش می‌دهد، این تابع بردار x را به‌عنوان ورودی گرفته و

متناسب با هر عنصر تکه‌ای از نمودار را به آن اختصاص می‌دهد. این تابع آرگومان دوم را نیز می‌گیرد که این آرگومان یک بردار هم‌بعد با بردار x می‌باشد که عناصر آن صفر یا یک می‌باشند. برای حالت یک، تکه‌ای که برای عنصر x(i) نمایش داده می‌شود جدا از نمودار قرار می‌گیرد، در حالت پیش‌فرض صفر، تکه‌ها به‌هم چسبیده خواهند بود. با استفاده از یک آرایه سلولی می‌توان هر تکه از نمودار را نامگذاری نمود. با دنبال کردن دستورات

```
x=[1 3 6];
subplot(2,2,1)
pie(x)
subplot(2,2,2)
pie(x, [0 0 1])
subplot(2,2,3)
pie(x,[1 1 1],{'Slice 1','Slice 2','Slice 3'})
subplot(2,2,4)
pie3(x, [0 1 0])
```

برای ایجاد شکل ۱۷.۴ با این تابع به‌خوبی آشنا می‌شویم.



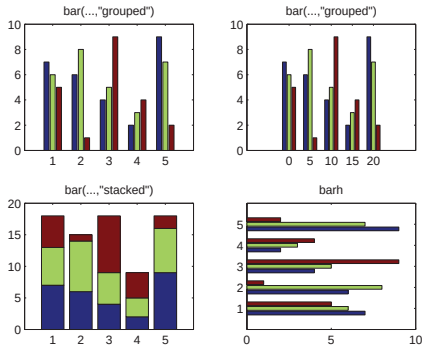
شکل ۱۷.۴ تابع pie

تابع bar

از تابع bar برای ترسیم نمودارهای میله‌ای استفاده می‌شود که شامل چهار حالت دوبعدی(افقی و عمودی) و سه بعدی(افقی و عمودی) می‌باشد، هرکدام از این حالت‌ها شامل حالت پشته‌ای و گروهی نیز می‌باشند. ساده‌ترین کاربرد این تابع اعمال آن روی یک ماتریس $m \times n$ می‌باشد که در حالت دوبعدی نمودار میله‌ای عناصر هر سطر به‌هم می‌چسبند. دستورات

```
Y = [7 6 5;6 8 1;4 5 9;2 3 4;9 7 2];
subplot(2,2,1)
bar3(Y)
title('bar3(...,"detached")')
subplot(2,2,2)
bar3(Y,'grouped')
title('bar3(...,"grouped")')
subplot(2,2,3)
bar3(Y,'stacked')
title('bar3(...,"stacked")')
subplot(2,2,4)
bar3h(Y)
title ('bar3h')
```

نحوه استفاده از این تابع را برای ایجاد شکل ۱۸.۴ نشان می‌دهد. در حالتی که تابع دو آرگومان



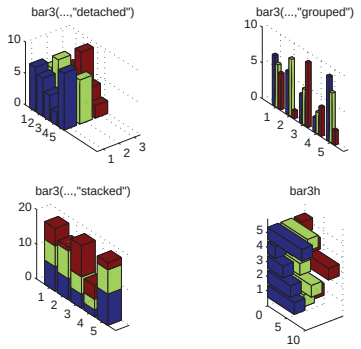
شکل ۱۸.۴ تابع bar

ورودی را می‌گیرد، آرگومان اول موقعیت نمودارهای میله‌ای را روی محور نمایش می‌دهند. برای تابع bar3 حالت پیش‌فرض 'datached' می‌باشد، با اجرای دستورات

```
Y = [7 6 5;6 8 1;4 5 9;2 3 4;9 7 2];
subplot(2,2,1)
bar(Y)
title('bar(...,"grouped")')
subplot(2,2,2)
bar(0:5:20,Y)
title('bar(...,"grouped")')
subplot(2,2,3)
```

```
bar(Y,'stacked')
title('bar(...,"stacked")')
subplot(2,2,4)
barh(Y)
title ( 'barh')
```

برای ایجاد شکل ۱۹.۴ می‌توان به تفاوت آن‌ها پی برد. در حالت 'datached' برخی از نمودارها



شکل ۱۹.۴ تابع bar3

در پشت نمودارهای دیگر پنهان می‌شوند که برای حل این مشکل می‌توان شکل را با ماوس چرخاند.

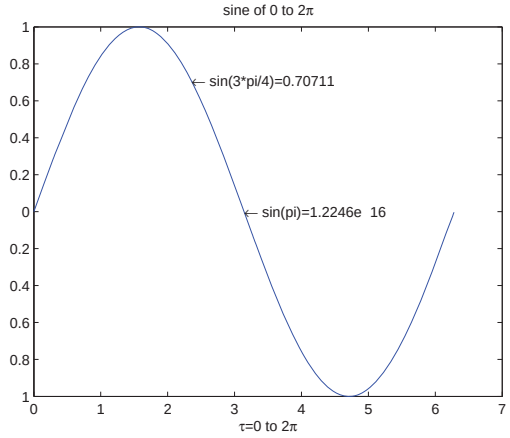
۴.۴ چند مثال

مثال ۱.۴.۴. نمودار ۲۰.۴ را رسم نمایید.

```
x=0:0.01:2*pi;
y=plot(x,sin(x));
xlabel('\tau=0 to 2\pi')
title('sine of 0 to 2\pi')
text(3*pi/4, sin(3*pi/4),['\leftarrow sin(3*pi/4)=',...
    num2str(sin(3*pi/4))])
text(pi, sin(pi),['\leftarrow sin(pi)=',num2str(sin(pi))])
```

مثال ۲.۴.۴. نمودار ۲۱.۴ را رسم نمایید.

```
x=0:2*pi/40:2*pi;
y = sin(x);
plot(x,y,'ro')
hold on
y = cos(x);
plot(x,y,'b+')
legend('sin', 'cos')
title('sin and cos on one graph')
```



شکل ۲۰.۴

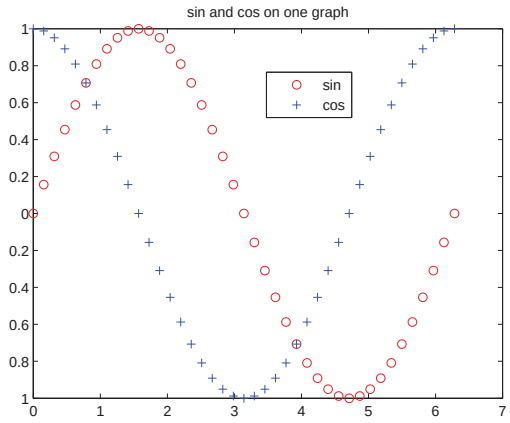
مثال ۳.۴.۴. رسم نمودارهای قطبی
نقطه (x,y) در مختصات کارتزین را می‌توان به‌صورت (θ,r) در مختصات قطبی نمایش داد که در آن

$$\mathrm{x}=\mathrm{r}\cos(\theta),\quad \mathrm{y}=\mathrm{r}\sin(\theta),\quad 0\le\theta\le 2\pi.$$

با استفاده از دستور

```
polar(theta, r)
```

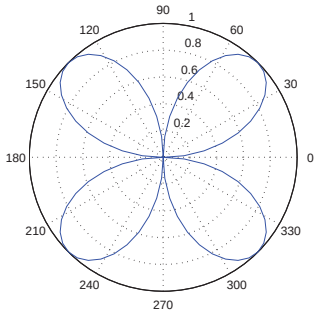
می‌توان یک نمودار در مختصات قطبی را رسم نمود. به‌عنوان مثال عبارت



شکل ۲۱.۴ نمودار در مختصات قطبی

```
x=0:pi/40:2*pi;
polar(x,sin(2*x)), grid
```

شکل ۲۲.۴ را تولید می‌کند.



شکل ۲۲.۴

مثال ۴.۴.۴. تابع pause

در برخی موارد نیاز داریم اجرای یک دستور یا یک برنامه مدتی متوقف شود، برای این کار از تابع pause استفاده می‌کنیم. به‌عنوان مثال با اجرای تکه برنامه زیر

```
for n=3:10
mesh(magic(n))
pause
end
```

پس از اجرای هر تکرار، نرم افزار منتظر فشردن یک کلید توسط کاربر می‌ماند و با فشردن هر کلید یک تکرار انجام می‌شود. می‌توان به‌جای این که نرم‌افزار منتظر فشردن کلید توسط کاربر بماند از دستور pause(n) استفاده نمود که در آن هر تکرار پس از توقف n ثانیه اجرا خواهد شد.

مثال ۵.۴.۴. تابع menu

تابع menu یک فهرست از انتخاب‌ها را برای کاربر ایجاد می‌کند که این فهرست به‌صورت یک جعبه گرافیکی نمایش داده می‌شود. نحوه استفاده از این تابع به‌صورت

```
choice=menu('mtitle','opt1','opt2',... ,'opt n')
```

می‌باشد که در آن mtitle به‌صورت یک رشته نام و opt1، ...، optn انتخاب‌های کاربر می‌باشند. خروجی این تابع شماره گزینه انتخاب شده خواهد بود. به عنوان مثال با اجرای دستورات

```
choice=menu('please choose color of plot','red','blue','green')
x=0:0.01:50;
color=['r', 'b', 'g'];
plot(x,sin(x),color(choice))
```

نرم‌افزار با ایجاد یک جعبه (شکل ۲۳.۴) منتظر انتخاب رنگ نمودار توسط کاربر می‌ماند و پس از انتخاب کاربر نمودار را رسم می‌کند.

شکل ۲۳.۴

فصل ۵

جعبه ابزار سمبولیک ریاضی

۱.۵ حساب دیفرانسیل و انتگرال

۱.۱.۵ محاسبه حد یک تابع

برای محاسبه حد یک تابع ابتدا نیاز داریم یک عبارت سمبولیک تعریف کنیم، برای این منظور ابتدا با استفاده از دستور syms متغیر یا متغیرهای مورد نیاز را به‌صورت سمبولیک تعریف می‌کنیم. می‌دانیم

$$\mathrm{f}'(\mathrm{x}) = \lim_{\mathrm{h}\to 0} \frac{\mathrm{f}(\mathrm{x}+\mathrm{h}) - \mathrm{f}(\mathrm{x})}{\mathrm{h}}.$$

بنابراین برای محاسبه مشتق تابع $\cos x$ با استفاده از تعریف فوق، دستورات زیر را به کار می‌بریم.

```
>> syms h n x
>> limit((cos(x+h)-cos(x))/h,h,0)
ans =
-sin(x)
```

همچنین برای نشان دادن $\lim_{\mathrm{n}\to\infty} \left(1+\frac{\mathrm{x}}{\mathrm{n}}\right)^{\mathrm{n}} = \mathrm{e}^{\mathrm{x}}$، دستورات زیر را به کار می‌بریم.

```
>> limit((1+x/n)^n,n,inf)
ans =
exp(x)
```

به‌دست آوردن حدچپ و راست

فرض کنید می‌خواهیم حد چپ و راست تابع $\frac{x}{|x|}$ وقتی‌که $x \to 0$ را به‌دست آوریم، لذا دستورهای زیر را می‌نویسیم.

```
>> limit(x/abs(x),x,0,'left')
ans =
-1
>> limit(x/abs(x),x,0,'right')
ans =
1
```

چون حد چپ و راست باهم برابر نمی‌باشند پس حد تابع موجود نمی‌باشد و اگر از آن حد بگیریم خواهیم داشت

```
>> limit(x/abs(x),x,0)
ans =
NaN
```

۲.۱.۵ مشتق‌گیری

مشتق گیری با استفاده از تابع diff انجام می‌گیرد. به عنوان مثال برای محاسبه مشتقات توابع $\sin(5x)$ و $e^x \cos x$ به‌صورت زیر عمل می‌کنیم

```
>> syms x
>> f=sin(5*x);
>> diff(f)
ans =
5*cos(5*x)
>> g=exp(x)*cos(x);
>> diff(g)
ans =
exp(x)*cos(x) - exp(x)*sin(x)
```

برای به‌دست آوردن مشتق nام تابع از دستور $\mathrm{diff(f, n)}$ استفاده می‌کنیم

```
>> diff(g,2)
ans =
```

```
(-2)*exp(x)*sin(x)
>> diff(diff(g))
ans =
(-2)*exp(x)*sin(x)
```

برای مشتق‌گیری از یک تابع چند متغیره ابتدا باید مشخص کرد که نسبت به کدام متغیر مشتق گرفته می‌شود، در غیر این‌صورت به‌طور پیش‌فرض اولین متغیر به‌ترتیب حروف الفبا انتخاب خواهد شد

```
>> syms s t
>> f=sin(s*t)*s;
>> diff(f,t)
ans = s^2*cos(s*t)
>> diff(f,s)
ans =
sin(s*t) + s*t*cos(s*t)
>> diff(f,t,2)
ans =
-s^3*sin(s*t)
```

۳.۱.۵ انتگرال‌گیری

اگر f یک عبارت سمبولیک باشد انتگرال آن از دستور int(f) و انتگرال آن نسبت به متغیری مانند v، از دستور int(f, v) محاسبه می‌گردد. انتگرال معین نیز به‌صورت int(f, a, b) محاسبه می‌گردد.

```
>> int(x)
ans =
x^2/2
>> int(a*cos(a*x),a)
ans =
(cos(a*x) + a*x*sin(a*x))/x^2
>> int(sin(2*x),0,pi/2)
ans =
1
```

انتگرال‌گیری از یک تابع با پارامترهای حقیقی

به عنوان مثال اگر بخواهید مقدار

$$\int_{-\infty}^{+\infty} e^{-ax^2} dx$$

را بدون نسبت دادن مقداری به a به‌دست آورید، متلب a را به‌عنوان یک عدد مختلط در نظر خواهد گرفت و جواب انتگرال یک تابع تکه‌ای خواهد بود که به a بستگی خواهد داشت. اگر بخواهید انتگرال را با مقدار حقیقی و مثبت a به‌دست آورید آنگاه باید دستورات ذیل را وارد نمایید

```
>> syms a positive;
>> syms x
>> f = exp(-a*x^2);
>> int(f,x,-inf,inf)
ans =
pi^(1/2)/a^(1/2)
```

۴.۱.۵ جمع سمبولیک

با استفاده از تابع symsum می‌توان جمع یک عبارت سمبولیک را به‌دست آورد، مثال‌های زیر کاربرد این دستور را روشن می‌کنند.

```
>> syms k n x
>> symsum(k^2)
ans =
k^3/3 - k^2/2 + k/6
>> symsum(k)
ans =
k^2/2 - k/2
>> symsum(k^2,0,10)
ans =
385
>> symsum(x^k/sym('k!'),k,0,inf)
ans =
exp(x)
```

در دستور آخر برای این‌که متلب عملگر فاکتوریل را بشناسد، یک عبارت سمبولیک را ایجاد کرده‌ایم، می‌دانیم سری $1 + x + x^2 + \ldots$ به $\frac{1}{1-x}$ همگراست در صورتی که $|x| < 1$، پس خواهیم داشت

```
>> symsum(x^k,k,0,inf)
ans =
piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x - 1)])
```

۵.۱.۵ سری تیلور

با استفاده از دستور $\mathrm{taylor(f, n, a)}$ می‌توان تا جمله‌ی n ام سری تیلور تابع f حول نقطه a را به‌دست آورد که در حالت $\mathrm{a} = 0$ سری مورد نظر سری مک‌لورن خواهد بود.

```
>> syms x
>> f=1/(5+4*cos(x));
>> t=taylor(f,8)
t =
(49*x^6)/131220 + (5*x^4)/1458 + (2*x^2)/81 + 1/9
>> g=exp(sin(x));
> t=taylor(g,2,2)
t =
exp(sin(2)) + cos(2)*exp(sin(2))*(x - 2)
```

۲.۵ ساده‌سازی و جانشانی

سه عبارت ذیل را در نظر بگیرید

```
syms x
f = x^3-6*x^2+11*x-6
g = (x-1)*(x-2)*(x-3)
h = -6+(11+(-6+x)*x)*x
```

این سه عبارت هرسه نمایش‌های مختلفی از یک تابع ریاضی می‌باشند و هرکدام برتری خاصی نسبت به دیگری دارند، به عنوان مثال عبارت f بیشتر برای نمایش یک چندجمله‌ای بکار می‌رود، عبارت g، تجزیه f می‌باشد و برای پیدا کردن ریشه‌های یک چندجمله‌ای مفید می‌باشد، عبارت h حالت آشیانه‌ای یا هورنر یک چندجمله‌ای می‌باشد و برای محاسبات عددی

این حالت عملیات ریاضی کمتری نیاز دارد و دقت محاسبات در آن بالا می‌باشد. در این‌جا چند تابع را معرفی می‌کنیم که برای ساده کردن عبارت‌های جبری کمک می‌کنند.
* تابع collect عبارت f را به‌صورت یک چندجمله‌ای برحسب توان‌های یک متغیر مرتب می‌کند

```
>> d=x*(x*(x-6)+11)-6;
collect(d)
ans =
x^3 - 6*x^2 + 11*x - 6
>>w=x+t*x^3+(t^3+t*x)
>> collect(w,x)
ans =
x^3*t+(1+t)*x+t^3
```

* تابع expand یک عبارت را بسط می‌دهد

```
>> expand(x*(x*(x-6)+11)-6)
ans =
x^3 - 6*x^2 + 11*x - 6
>> expand(cos(x+y))
ans =
cos(x)*cos(y) - sin(x)*sin(y)
```

* تابع horner چندجمله‌ای را به‌صورت آشیانه‌ای تبدیل می‌کند

```
> horner(x^3-6*x^2+11*x-6)
ans =
x*(x*(x - 6) + 11) - 6
```

* تابع factor یک عبارت را به عامل‌های اول تجزیه و یک عبارت گویا را در صورت امکان ساده می‌کند. به مثال زیر توجه کنید f به‌صورت یک چندجمله‌ای گویا باشد را به‌صورت ضرب چندجمله‌ای‌هایی با درجه کمتر تجزیه می‌کند

```
> factor(x^3-6*x^2+11*x-6)
ans =
(x - 3)*(x - 1)*(x - 2)
>> factor(x^6+1)
```

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

* تابع simplify یک عبارت را ساده می‌کند و تابع simple به‌دنبال ساده‌ترین حالت می‌گردد

```
>> syms x;
f = cos(x)^2 + sin(x)^2; f = simple(f)
g = cos(3*acos(x)); g = simplify(g)
f =
1
g =
4*x^3 - 3*x
```

* برای جانشانی عبارت‌های سمبولیک دو تابع subs و subexpr را داریم، دستورات

```
>> syms a x; s=solve('x^3+a*x+1')
s =
((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3)
 - a/(3*((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))

(3^(1/2)*(a/(3*((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))
 + ((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))*i)/2
 + a/(6*((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))
 - ((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3)/2

a/(6*((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))
 - (3^(1/2)*(a/(3*((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))
 + ((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3))*i)/2
 - ((a^3/27 + 1/4)^(1/2) - 1/2)^(1/3)/2
```

معادله $x^3 + ax + 1$ را برای x حل می‌کند، و می‌بینیم که s یک عبارت بزرگ و شامل زیرعبارت‌هایی می‌باشد که چندین بار تکرار شده‌اند، با اجرای دستور subexpr خواهیم داشت

```
>> subexpr(s)
sigma =
(a^3/27 + 1/4)^(1/2) - 1/2
```

```
ans =
sigma^(1/3) - a/(3*sigma^(1/3))
a/(6*sigma^(1/3)) - sigma^(1/3)/2 - (3^(1/2)*i*(a/(3*sigma^(1/3)))...
+ sigma^(1/3)))/2
a/(6*sigma^(1/3)) - sigma^(1/3)/2 + (3^(1/2)*i*(a/(3*sigma^(1/3))) ...
+ sigma^(1/3)))/2
```

می‌بینیم که اعمال این تابع باعث می‌شود عامل‌هایی که زیاد تکرارشده‌اند در یک متغیر ذخیره شوند و عبارت در نهایت ظاهری ساده‌تر داشته باشد. برای زیباتر شدن جواب نمایش داده شده می‌توان از تابع pretty استفاده نمود، در این‌صورت داریم

```
pretty(ans)
```

```
  +-                                                                  -+
  |                          1/3        a                             |
  |                    sigma    - ----------                         |
  |                                      1/3                         |
  |                                3 sigma                           |
  |                                                                  |
  |                   1/3                                            |
  |       a       sigma       1/2 /      a            1/3 \         |
  |  ---------- - -------- - 3    | ---------- + sigma    | 1/2 I   |
  |         1/3       2           |        1/3            |         |
  |  6 sigma                      \ 3 sigma               /         |
  |                                                                  |
  |                   1/3                                            |
  |       a       sigma       1/2 /      a            1/3 \         |
  |  ---------- - -------- + 3    | ---------- + sigma    | 1/2 I   |
  |         1/3       2           |        1/3            |         |
  |  6 sigma                      \ 3 sigma               /         |
  +-                                                                  -+
```

* کاربرد تابع subs برای جایگذاری یک مقدار جدید به جای متغیرهای یک عبارت می‌باشد

```
>> syms a b c
>> subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})
ans =
```

```
sin(2) + cos(alpha)
>> S =(a^2 - a*b - a*c + b^2 - b*c + c^2)^(1/2);
>> subs(S,{a,b,c},{10,2,10})
ans =
8
```

۳.۵ چندجمله‌ای‌ها

چندجمله‌ای‌های سمبولیک را به‌راحتی می‌توان با استفاده از متغیرهای سمبولیک و نمادگذاری معمول در متلب ایجاد نمود. با استفاده از تابع coeffs می‌توان ضرایب و جملات متناظر یک چندجمله‌ای را (بدون ترتیب) خارج نمود

```
>> syms x
>> p = (2/3)*x^3-x^2-3*x+1
p =
(2*x^3)/3 - x^2 - 3*x + 1

>> [c, terms] = coeffs(p,x)
c =
[ 2/3, -1, -3, 1]
terms =
[ x^3, x^2, x, 1]
```

توابع poly2sym و sym2poly چندجمله‌ای سمبولیک و بردار ضرایب یک چندجمله‌ای را به یکدیگر تبدیل می‌کنند(ترتیب ضرایب در این حالت به‌صورت استاندارد از بزرگترین توان به کوچکترین توان خواهد بود)

```
>> a = sym2poly(p)
a =
    0.6667   -1.0000   -3.0000    1.0000

>> q = poly2sym(a)
q =
   (2*x^3)/3 - x^2 - 3*x + 1
```

تقسیم یک چند جمله‌ای بر چندجمله‌ای دیگر را می‌توان با استفاده از تابع quorem انجام داد، خروجی، تابع خارج قسمت و باقیمانده خواهد بود

```
>> [q,r] = quorem(p,x^2)
q =
2/3*x-l
r =
1-3*x
```

مقدار یک چندجمله‌ای در یک نقطه را می‌توان با استفاده از تابع polyval به‌صورت $\text{polyval}(p, x)$ به‌دست آورد که در آن p بردار ضرایب چندجمله‌ای و x یک عدد و یا یک متغیر از پیش تخصیص یافته می‌باشد. به عنوان مثال دستورات ذیل مقدار چندجمله‌ای $f(x) = 5x^4 + 2x^2 - 1$ را به‌ازای $x = 5$ محاسبه و همچنین نمودار آن‌را به ازای $3 \leq x \leq 7$ برای ما ترسیم خواهد نمود(امتحان کنید)

```
>> p=[5 0 0 2 -1]
>> polyval(p,5)
ans =
        3134
>> x = 3:.1:7;
>> y=polyval(p,x);
>> plot(x,y)
```

با استفاده از تابع roots می‌توان ریشه‌های یک چندجمله‌ای را به‌دست آورد، همچنین وقتی ریشه‌های یک چندجمله‌ای را داشته باشیم با استفاده از دستور poly می‌توان خود چندجمله‌ای را به‌دست آورد

```
>> p=[1 3 6 4];
>> r=roots(p)
r =
  -1.0000 + 1.7321i
  -1.0000 - 1.7321i
  -1.0000
>> poly(r)
ans =
    1.0000    3.0000    6.0000    4.0000
```

مثال زیر نحوه جمع، ضرب و تقسیم چندجمله‌ای‌های
$$f(x) = 3x^3 - 2x - 6 \qquad p(x) = 5x^4 - 10x^2 + 1$$
را نشان می‌دهد:

```
>> f=[3 0 -2 -6];
>> p=[5 0 -10 0 1];
>> A=p+[0 f];    %Addition
>> poly2sym(A)
ans =
5*x^4 + 3*x^3 - 10*x^2 - 2*x - 5

>> M=conv(f,p)   %Multiplication
M =
    15     0   -40   -30    23    60    -2    -6

>> [q r]=deconv(p,f)   %Division
q =
     0
r =
     3     0    -2    -6
```

۴.۵ حساب با دقت متغیر

سه نوع از اعمال محاسباتی عبارتند از:

Numeric	شناور ممیز حساب MATLAB
Rational	دقیق سمبولیک حساب Maple
VPA	متغیر دقت حساب Maple

به‌عنوان مثال حساب دقیق کسری را به‌صورت ذیل می‌توان به‌دست آورد

```
S=simple(sym('13/17+17/23'))
S =
588/391
```

با محاسبات عددی نیز آشنا هستیم، به‌عنوان مثال داریم

```
>> format long
>> pi*log(2)
ans =
   2.177586090303602
```

محاسبات عددی در متلب به‌طور تقریبی با حساب ممیز شناور[۱] ۱۶ رقمی انجام می‌شود. با استفاده از vpa [۲] می‌توان نتایج محاسبات را با دقت دلخواه به‌دست آورد. به‌عنوان مثال داریم

```
>> vpa('pi * log(2)')
>> vpa(sym(pi) * log(sym(2)))
>> vpa('pi * log(2)', 50)
ans =
2.1775860903036021305006888982376

ans =
2.1775860903036021305006888982376

ans =
 2.17758609030360213050068888982376139473385837003693
```

دقت پیش فرض vpa، ۳۲ رقم می‌باشد بنابراین دو عبارت اول فوق تا ۳۲ رقم و عبارت سوم تا ۵۰ رقم درست می‌باشد. دقت پیش‌فرض را می‌توان با استفاده از تابع digits به‌صورت زیر تغییر داد

```
>> digits
Digits = 32

>> digits(20)
>> vpa(pi)
 ans =
 3.1415926535897932385
```

[۱] Floating-point arithmetic
[۲] Variable precision arithmetic