به نام خدا

دانشگاه آزاد اسلامی واحد نجف‌آباد

دانشکده مهندسی برق

مقدمه‌ای بر توابع ریاضی، رسم شکل و برنامه‌نویسی در

نرم افزار MATLAB

مدرس:

ایمان صادق‌خانی

– نحوه تعریف و واردکردن یک ماتریس

برای جداکردن درایه‌های هر سطر از کلید Space یا نماد "،" استفاده می‌شود. برای تعریف سطر بعدی از نماد " ; " استفاده نموده و یا کلید Enter را فشار دهید.

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

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

پس از واردکردن ماتریس A و زدن کلید Enter، این ماتریس در Workspace ذخیره می‌شود. با دوبار کلیک‌کردن بر نماد ماتریس A در Workspace، پنجره Variable Editor باز خواهد شد که امکان ویرایش ماتریس A را فراهم خواهد نمود.

نکته: استفاده از نماد ";" در آخر هر دستور، مانع نمایش خروجی آن دستور در محیط کار و شلوغ‌شدن آن می‌شود. دستور clc محیط کار را پاک می‌کند. همچنین دستور clear all متغیرهای موجود در Workspace را حذف خواهد نمود.

ترانهاد ماتریس A با دستور 'A بدست خواهد آمد.

دستور diag(A) قطر اصلی ماتریس A را در یک ستون نمایش خواهد داد.

دستور sum(A) درایه‌های ماتریس A را بصورت ستونی جمع می‌کند. برای ماتریس فوق داریم:

```
>> sum(A)

ans =
      12    15    18
```

دستور magic(n) یک ماتریس $n \times n$ ایجاد می‌کند که مجموع درایه‌های هر ستون آن با هم برابر است.

مثال: دستورات زیر را وارد نموده و نتیجه را مشاهده نمایید:

```
>> sum(magic(5))

>> sum(diag(A))
```

دستور size(A) تعداد سطر و ستون‌های یک ماتریس را نشان می‌دهد. برای مثال:

```
>> A=[1 3;5 6;4 9]
>> size(A)

ans =

   3    2
```

دستور length(A) از بین تعداد سطر و ستون هر کدام بزرگتر بود، آنرا نشان می‌دهد. در صورتی که ماتریس به صورت ستونی یا سطری باشد، این دستور تعداد درایه‌های ماتریس را نشان خواهد داد.

دستور sort(A) درایه‌های یک ماتریس را از کوچک به بزرگ مرتب می‌کند. برای مثال:

```
>> sort(A)

ans =

   1    3
   4    6
   5    9
```

– نمایش آرایه‌های یک ماتریس

برای دسترسی به یک درایه خاص از ماتریس A از دستور A(n,m) استفاده می‌کنیم که n شماره سطر و m شماره ستون درایه مورد نظر می‌باشد. همچنین می‌توان از دستور A(i) استفاده نمود که i شماره درایه مورد نظر بوده که بصورت زیر بدست می‌آید:

$$A = \begin{bmatrix} A(1,1) & A(1,2) & A(1,3) \\ A(2,1) & A(2,2) & A(2,3) \\ A(3,1) & A(3,2) & A(3,3) \end{bmatrix}$$

$$A = \begin{bmatrix} A(1) & A(4) & A(7) \\ A(2) & A(5) & A(8) \\ A(3) & A(6) & A(9) \end{bmatrix}$$

– آشنایی با عملگر ":"

این عملگر را با مثال توضیح خواهیم داد.

```
>> A = 1:2:10
```

این دستور اعداد ۱ تا ۱۰ را با گام ۲ نمایش می‌دهد.

برای مشاهده سطر اول ماتریس A از دستور A(1,:) استفاده می‌کنیم.

مثال: دستورات زیر را وارد نموده و خروجی را مشاهده کنید:

```
>> A(:,2)
>> sum(A(:))
```

– ساخت چند ماتریس پرکاربرد

دستور zeros(n,m) یک ماتریس $n \times m$ با درایه‌های صفر می‌سازد.

دستور ones(n,m) یک ماتریس $n \times m$ با درایه‌های یک می‌سازد.

دستور eye(n,m) یک ماتریس $n \times m$ با درایه‌های صفر و قطر اصلی یک می‌سازد.

دستور rand(n,m) یک ماتریس $n \times m$ با درایه‌های تصادفی در بازه صفر و یک می‌سازد.

– حذف یک سطر یا ستون ماتریس

برای حذف سطر دوم ماتریس A از دستور زیر استفاده می‌شود:

```
>> A(2,:) = []
```

بطور مشابه دستور A(:,2) = [] ستون دوم ماتریس A را حذف خواهد نمود.

– دیگر دستورات پرکاربرد

دستور det(A) دترمینان ماتریس A و دستور inv(A) معکوس این ماتریس را محاسبه خواهد نمود.

– متغیرها

انواع متغیرها در متلب عبارتند از:

```
>> a1 = 50;
>> a2 = 'k';
>> a3 = 'I am student';
```

– اعداد

انواع اعداد در متلب عبارتند از:

```
>> a1 = 100;
>> a2 = 2.3e3;
>> a3 = 1+2j;
>> a4 = 1+2i;
```

– عملگرها

دستور A*B عمل ضرب دو ماتریس A و B را انجام می‌دهد، البته باید قاعده ضرب ماتریس‌ها برقرار باشد. برای ضرب نظیر به نظیر درایه‌های دو ماتریس (ضرب نقطه‌ای) از دستور A.*B استفاده نمایید.

همچنین اگر درایه‌های ماتریس A اعداد مختلط باشند، دستور A' علاوه بر ترانهاده کردن ماتریس A، درایه‌های این ماتریس را نیز مزدوج می‌کند. برای جلوگیری از مزدوج‌شدن درایه‌ها از دستور A.' استفاده نمایید.

– توابع در متلب

برای نمایش کلیه توابع متلب از دستور help elfun استفاده کنید.

دستور sin(n) سینوس عدد n را نمایش می‌دهد. البته توجه نمایید که پیشفرض متلب ((رادیان)) بوده و عدد n را بصورت رادیان درنظر می‌گیرد. برای واردکردن اعداد بصورت رادیان باید از نماد π استفاده نمود. این نماد در متلب بصورت pi تعریف شده است. مثلاً برای محاسبه سینوس ۳۰ درجه از دستور sin(pi/6) استفاده می‌شود. راه دیگر استفاده از دستور sind(n) است که عدد n را برحسب درجه درنظر می‌گیرد.

برخی توابع پرکاربرد در متلب عبارتند از:

کاربرد	نام تابع
توابع مثلثاتی:	
سینوس بر حسب رادیان	sin
سینوس بر حسب درجه	sind
معکوس سینوس	asin
سینوس هایپربولیک	sinh
معکوس سینوس هایپربولیک	asinh
کسینوس بر حسب رادیان	cos
کسینوس بر حسب درجه	cosd
معکوس کسینوس	acos
کسینوس هایپربولیک	cosh
معکوس کسینوس هایپربولیک	acosh
تانژانت بر حسب رادیان	tan
تانژانت بر حسب درجه	tand
معکوس تانژانت	atan
تانژانت هایپربولیک	tanh

معکوس تانژانت هایپربولیک	atanh
کتانژانت بر حسب رادیان	cot
کتانژانت بر حسب درجه	cotd
معکوس کتانژانت	acot
کتانژانت هایپربولیک	coth
معکوس کتانژانت هایپربولیک	acoth
سکانت	sec
کسکانت	csc
توابع نمایی:	
محاسبه e^n	exp(n)
لگاریتم در پایه e (تابع Ln)	log
لگاریتم در پایه ۱۰	log10
لگاریتم در پایه ۲	log2
محاسبه 2^n	pow2(n)
محاسبه جذر n	sqrt(n)
توابع مختلط:	
محاسبه اندازه عدد مختلط n	abs(n)
محاسبه زاویه عدد مختلط n بر حسب رادیان	angle(n)
ایجاد عدد مختلط n + j m	complex(n,m)
محاسبه مزدوج عدد مختلط n	conj(n)
محاسبه قسمت موهومی عدد مختلط n	imag(n)
محاسبه قسمت حقیقی عدد مختلط n	real(n)
توابع گردکردن:	
گردکردن عدد n به سمت نزدیک‌ترین عدد صحیح	round(n)
گردکردن عدد n به سمت صفر	fix(n)
گردکردن عدد n به سمت منفی بینهایت	floor(n)
گردکردن عدد n به سمت مثبت بینهایت	ceil(n)

نکته: تابع abs(n) درصورتیکه n یک عدد حقیقی باشد، نقش قدرمطلق را انجام می‌دهد. همچنین برای محاسبه $\sqrt[n]{A^m}$ از دستور A^(m/n) استفاده نمایید.

نکته: تبدیل دکارتی به قطبی با استفاده از دستور(cart2pol(real,image و تبدیل قطبی به دکارتی نیز با استفاده از دستور(pol2cart(angle,absolute صورت می‌گیرد. دقت نمایید که زاویه در این دستورات برحسب رادیان است. برای مثال:

```
>> [angle,absolute]=cart2pol(3,4)
angle =
   0.9273
absolute =
    5

>> [real,image]=pol2cart(0.9273,5)
real =
   3.0000
imag =
   4.0000
```

– چند جمله‌ای‌ها در متلب

نرم‌افزار متلب دارای چندین دستور ساده برای کار با چندجمله‌ای‌هاست که اساس کار این توابع بردارهاست. می‌توانید با تایپ دستور help polyfun این توابع را مشاهده نمایید. چند جمله‌ای‌ها بصورت یک ماتریس سطری و با ضرایب خود تعریف می‌شوند.

$$f(x)=9x^3-5x^2+3x+7$$

```
>> f = [9 -5 3 7];
```

دقت کنید درصورتیکه یکی از توان‌های x (مثلاً x^2) موجود نبود، بجای ضریب آن در ماتریس عدد صفر را قرار می‌دهیم.

برای جمع دو چند جمله‌ای باید مرتبه آنها برابر باشد:

$$g(x)=6x^2-2x+2$$

```
>> g = [6 -2 2];
```

همانگونه که مشاهده می‌کنید درجه f از مرتبه ۳ ولی درجه g از مرتبه ۲ است. بنابراین برای جمع دو چند جمله‌ای باید مرتبه هر دو یکی شود. اینکار با اضافه‌کردن یک صفر بجای ضریب x^3 در g صورت می‌گیرد:

```
>> g1 = [0 g];
>> h = f + g1;
```

برای ضرب و تقسیم دو چند جمله‌ای دیگر نیازی نیست مرتبه آنها یکی باشد:

$$f(x)g(x) = (9x^3 - 5x^2 + 3x + 7)(6x^2 - 2x + 2)$$

```
>> p = conv(f,g)
```

$$\frac{f(x)}{g(x)} = \frac{9x^3 - 5x^2 + 3x + 7}{6x^2 - 2x + 2}$$

```
>> [a b] = deconv(f,g)
```

همانگونه که مشاهده می‌کنید برای اینکه خارج قسمت و باقیمانده تقسیم دو چندجمله‌ای را داشته باشیم، بایستی حاصل تقسیم را در یک ماتریس سطری دو ستونه بریزیم.

برای محاسبه مقدار چند جمله‌ای f در نقطه a از دستور polyval(f,a) استفاده می‌کنیم که a یک عدد است.
برای محاسبه ریشه‌های چند جمله‌ای f از دستور roots(f) استفاده می‌نماییم.
برای بدست‌آوردن یک چند جمله‌ای از روی ریشه‌های آن ابتدا باید آن ریشه‌ها را در یک ماتریس ستونی قرار داده و سپس از دستور poly استفاده کنیم:

```
>> r = [1;2.3;-0.8;4];
>> poly(r)
```

– حل دستگاه n معادله n مجهولی

یکی دیگر از کاربردهای ماتریس‌ها حل دستگاه معادلات است. دستگاه زیر را درنظر بگیرید:

$$\begin{cases} 3x_1 + 5x_2 - 2x_3 = 10 \\ x_1 + 2x_2 + x_3 = -2 \\ -x_1 + x_2 + 3x_3 = 6 \end{cases}$$

می‌توان معادله را با ماتریس‌های زیر مشخص نمود:

ماتریس ضرایب:
$$A=\begin{bmatrix}3 & 5 & -2\\ 1 & 2 & 1\\ -1 & 1 & 3\end{bmatrix}$$

ماتریس مجهولات:
$$X=\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

ماتریس جواب‌ها:
$$B=\begin{bmatrix}10\\ -2\\ 6\end{bmatrix}$$

ماتریس‌های A و B را وارد کرده و با یکی از دو دستور زیر ماتریس مجهولات (X) بدست خواهد آمد:

```
>> X = inv(A)*B
>> X = A\B
```

– نحوه نمایش اعداد در متلب

پیش‌فرض متلب برای نمایش اعداد چهار رقم اعشار می‌باشد. برای تغییر آن از تب Home (منوی File) گزینه Preferences را انتخاب نموده و در قسمت Command Window منوی کشویی Numeric format را باز کنید. حال می‌توانید یکی از گزینه‌ها را انتخاب نمایید. راه دیگر استفاده از دستور format می‌باشد یعنی نحوه نمایش دلخواه را پس از دستور format تایپ نمایید. برای مثال برای نمایش اعداد تا ۱۶ رقم اعشار از دستور format long استفاده می‌شود. در جدول زیر فرمت‌های مختلف نمایش اعداد بیان شده است.

دستور	مفهوم آن در متلب
format short	نمایش اعداد با چهار رقم اعشار
format long	نمایش اعداد با شانزده رقم اعشار
format short e	پنج رقم (چهار رقم اعشار) 1.2327e+03
format long e	۱۶ رقم (۱۵ رقم اعشار)
format bank	دو رقم اعشار
format +	اعداد بصورت مثبت، منفی یا صفر نمایش داده می‌شوند
format rat	اعداد بصورت کسری نمایش داده می‌شوند

– رسم توابع در متلب

این دستور براساس آرگومان‌های ورودی‌اش توابع مختلف را رسم می‌کند. اگر این دستور بصورت تک آرگومانی مورد استفاده قرار گیرد و متغیر حقیقی باشد، آنرا بر حسب اندیسش (شماره ستون درایه‌های آن متغیر در Workspace) رسم می‌کند و اگر متغیر موهومی باشد، قسمت حقیقی برحسب قسمت موهومی رسم می‌گردد:

```
>> y = 2:0.3:10;
>> plot(y)
```

```
>> z = 0.1+0.9i;
>> n = [0:0.01:10];
>> plot(z.^n)
```

اگر از این دستور بصورت دو آرگومانی استفاده شود، آرگومان دوم را برحسب آرگومان اول رسم می‌نماید:

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

همچنین می‌توان بدون تعریف متغیر واسط y و بطور مستقیم تابع را در دستور plot وارد نمود. در این حالت در صورت تغییر x دیگر نیازی به تعریف مجدد y نمی‌باشد:

```
>> plot(x,sin(x))
```

– چند تکنیک در دستور plot

با دستورات xlabel، ylabel و title می‌توان توضیحاتی به محور x، محور y و عنوان نمودار اضافه نمود:

```
>>  xlabel('x=0:2\pi')
>> ylabel('sin(x)')
>> title('y=sin(x)')
```

برای رسم همزمان چند تابع بصورت زیر عمل می‌نماییم:

```
>> plot(x,sin(x),x,cos(x))
```

برای نمایش اطلاعات بر روی نمودار از دستور legend استفاده می‌گردد:

```
>> legend('sin(x)','cos(x)')
```

برای رسم همزمان همچنین می‌توان از دستورات hold on و hold off استفاده نمود. دستور hold on شکل قبلی را حفظ کرده و شکل جدید را بر روی شکل قبلی رسم می‌کند. دستور hold off تمامی شکل‌های قبلی را حذف کرده و شکل جدید را به تنهایی رسم می‌نماید:

```
>> plot(x,sin(x))
>> hold on
>> plot(x,cos(x))
>> hold off
>> plot(x,2*sin(x+2))
```

دستور figure باعث بازشدن یک شکل جدید می‌شود. یعنی شکل قبلی در پنجره خود حفظ شده و یک پنجره جدید باز می‌شود و شکل جدید در پنجره جدید رسم می‌گردد.
با استفاده از دستور subplot(a,b,n) می‌توان چند نمودار را در یک پنجره و بصورت مجزا نشان داد که a، b و n هر سه عدد هستند. a تعداد شکل‌ها در راستای عمودی (تعداد سطر)، b تعداد شکل‌ها در راستای افقی (تعداد ستون) و n شماره شکل موردنظر است. شکل‌ها به ترتیب سطر (سطر اول، سطر دوم و ...) و از چپ به راست شماره‌گذاری می‌شوند:

```
>> subplot(2,2,1)
>> plot(x,sin(x))
>> subplot(2,2,2)
>> plot(x,cos(x))
>> subplot(2,2,3)
>> plot(x,2*sin(x))
>> subplot(2,2,4)
>> plot(x,cos(x-2))
```

دستور axis برای تعیین محدوده محورهای افقی و عمودی بکار می‌رود. شکل کلی دستور بصورت axis([xmin xmax ymin ymax]) می‌باشد:

```
>> plot(x,sin(x))
>> axis([0 2*pi -1.5 1.5])
```

از دستور gird برای مشبک‌کردن نمودار استفاده می‌شود.

دستور text(m,n,'matlab') کلمه matlab را در مختصات (m,n) قرار می‌دهد:

```
>> plot(x,sin(x))
>> text(2,0,'made by matlab')
```

برای مشخص‌کردن نوع خط و رنگ نمودار از شکل کلی دستور plot استفاده می‌کنیم:

```
plot(x,y,'color_style_marker')
```

که color رنگ نمودار و style و marker نوع خط نمودار را بصورت زیر مشخص می‌کنند:

- حروف c، m، y، r، g، b، w و k به ترتیب مطابق با رنگ‌های آبی متمایل به سبز، قرمز متمایل به صورتی، زرد، قرمز، سبز، آبی، سفید و سیاه می‌باشند.
- برای خط توپر از ((-))، برای خط تیره از ((:)) و برای خط نقطه‌چین از ((.-)) استفاده می‌شود.
- برای نمایش نمودار با ((+))، ((o))، ((*)) و ((x)) از همین علائم استفاده می‌شود. برای نمایش مربع، لوزی، پنج ضلعی، شش ضلعی بر روی نمودار به ترتیب از حروف s، d، p و h استفاده می‌گردد. برای نمایش مثلث بر روی نمودار از ((<))، ((>)) و ((^)) استفاده می‌شود که این علائم جهت رأس مثلث را نشان می‌دهند.

```
>> plot(x,sin(x),'r')
>> plot(x,cos(x),'g:^)
>> plot(x,2*sin(x+5),'-.d')
```

مثال: دستور زیر را وارد نموده و نتیجه را مشاهده نمایید:

```
>> x = 0:pi/100:2*pi;
>> plot(x,sin(x),'r',x,sin(x-2*pi/3),'g:',x,sin(x+2*pi/3),'y-.')
>> legend('Va','Vb','Vc')
```

توجه: برای اعمال دستورات xlabel، ylabel، title، legend، axis، grid و text به صورت‌های بیان‌شده، باید نمودار از قبل رسم شده و باز باشد.

– رسم نمودار قطبی

رسم نمودارهای قطبی نیز مشابه نمودارهای دکارتی بوده و تنها از دستور polar بجای plot استفاده می‌کنیم.

$$r = \frac{2}{1-0.5\cos(\theta)} \qquad 0 \le \theta \le 2\pi$$

```
>> th = 0:0.1:2*pi;
>> r = 2./(1-0.5*cos(th));
>> polar(th,r)
```

– رسم نمودارهای سه‌بعدی

برای رسم توابع سه‌بعدی از دستور plot3 استفاده می‌گردد.

$$\begin{cases} x = t \\ y = \sin(t) \\ z = \cos(t) \end{cases}$$

```
>> t = 0:pi/100:2*pi;
>> plot3(t,sin(t),cos(t))
```

مثال: دستور زیر را وارد نموده و نتیجه را مشاهده نمایید.

```
>> t = 0:pi/100:50*pi;
>> plot3(sin(t),cos(t),t)
```

– رسم توابع دو متغیره در متلب

تابع z = f(x,y) نشان‌دهنده سطحی است که در محورهای xyz رسم شده است. فرض کنید می‌خواهیم تابع زیر را رسم کنیم:

$$z = xe^{-[(x-y^2)^2+y^2]} \qquad -2 \le x \le 2 \quad -2 \le y \le 2$$

برای رسم این سطح ابتدا باید مجموعه نقاطی در صفحه ایجاد کنیم. اینکار با استفاده از تابع meshgrid انجام می‌شود:

```
>> [x y] = meshgrid(x,y);
```

برای رسم تابع دو متغیره از دستور mesh استفاده می‌گردد:

```
>> z = x.*exp(-((x-y.^2).^2+y.^2));
>> mesh(x,y,z)
```

توجه: به کاربرد عملگر نقطه در تعریف تابع بالا توجه نمایید.

برای رسم توابع دو متغیره از دستورهای دیگری نیز می‌توان استفاده نمود:

- دستور contour(x,y,z) پستی و بلندی‌های تابع را نمایش می‌دهد (کاربرد در نقشه‌برداری).
- دستور contour3(x,y,z) پستی و بلندی‌های تابع را بصورت سه بعدی نمایش می‌دهد.
- دستور surf(x,y,z) همانند دستور mesh است با این تفاوت که سطح را سایه می‌زند.
- از دستورات meshc(x,y,z)، meshz(x,y,z) و waterfall(x,y,z) نیز برای رسم توابع دو متغیره استفاده می‌شود.

– نمودار میله‌ای

برای رسم نمودار میله‌ای از دستور bar استفاده می‌شود. اگر بصورت یک بعدی استفاده شود، متغیر را بر حسب اندیسش رسم می‌کند و اگر بصورت دو آرگومانی استفاده شود، متغیر دوم را بر حسب متغیر اول رسم می‌کند.

```
>> a = [3 9 6 10];
>> bar(a)
```

```
>> a = [3 9 6;10 5 3;9 5 7];
>> b = [1388 1389 1390];
>> bar(b,a)
```

برای رسم نمودار میله‌ای سه‌بعدی از دستور bar3 و برای رسم این نمودار بصورت افقی از دستور barh استفاده می‌شود:

```
>> a = [3 9 6;10 5 3;9 5 7];
>> b = [1388 1389 1390];
>> bar3(b,a)
>> barh(b,a)
```

– نمودار دایره‌ای

برای رسم نمودار دایره‌ای از دستور pie استفاده می‌شود:

```
>> q = [3 9 15 2 5];
>> pie(q)
```

برای اینکه قسمت‌های مختلف از هم جدا شوند، از متغیر دوم استفاده می‌کنیم. اندازه متغیر دوم با متغیر اول برابر بوده و هر قسمت را که بخواهیم از بقیه جدا نشان داده شود، درایه‌اش را غیرصفر قرار می‌دهیم:

```
>> q = [3 9 15 2 5];
>> h = [0 1 0 0 1];
>> pie(q,h)
```

برای رسم نمودار دایره‌ای سه‌بعدی از دستور pie3 استفاده می‌شود:

```
>> q = [3 9 15 2 5];
>> h = [0 1 0 0 1];
>> pie3(q,h)
```

– رسم کره و بیضی

برای رسم کره از دستور sphere یا sphere(n) استفاده می‌شود که در آن n تعداد قسمت‌های تشکیل‌دهنده شکل است. برای رسم بیضی از دستور ellipsoid استفاده می‌گردد. معادله بیضی بصورت زیر است:

$$\frac{(x-x_c)^2}{x_r^2}+\frac{(y-y_c)^2}{y_r^2}+\frac{(z-z_c)^2}{z_r^2}=0$$

که $[x_c,y_c,z_c]$ مرکز بیضی و $[x_r,y_r,z_r]$ شعاع بیضی در راستای محورهای مختلف است. دستور ellipsoid(xc,yc,zc,xr,yr,zr,n) یک بیضی با مرکز و شعاع مشخص‌شده و متشکل از n قسمت رسم می‌کند. برای مثال:

```
[x, y, z] = ellipsoid(0,0,0,6,4,4,30);
surfl(x, y, z)
colormap winter
axis equal
```

– آشنایی مقدماتی با دستورات مهندسی کنترل

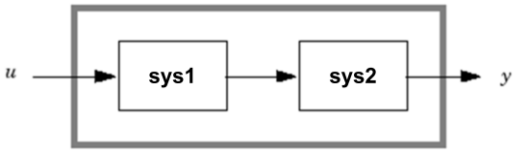
برای ساختن تابع تبدیل از مفهوم چندجمله‌ای‌ها استفاده کرده و از دستور tf کمک می‌گیریم. برای مثال برای ساختن تابع تبدیل زیر بصورت نشان داده شده عمل می‌کنیم:

$$G(s) = \frac{S^2 + 0.1S + 7.5}{S^4 + 0.12S^3 + 9S^2}$$

```
>> G = tf([1 0.1 7.5],[1 0.12 9 0 0]);
```

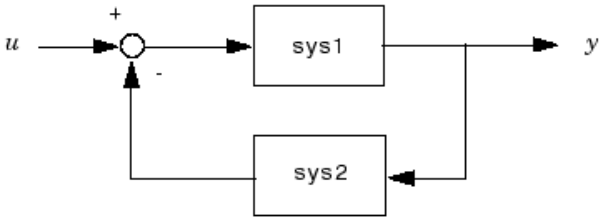
برای رسم نمودار بود از دستور bode(G)، نمودار نایکوئیست از دستور nyquist(G) و برای رسم مکان هندسی از دستور rlocus(G) استفاده می‌شود.

برای ایجاد ترکیب سری توابع تبدیل، از دستور series استفاده می‌شود:




```
>> sys = series(sys1,sys2)
```

برای ایجاد فیدبک از دستور feedback استفاده می‌گردد:




```
>> sys = feedback(sys1,sys2)
>> sys = feedback(sys1,sys2,+1)
```

از 1+ در خط دوم برای ایجاد فیدبک مثبت استفاده شده است.

برای رسم پاسخ پله از دستور step و برای رسم پاسخ ضربه از دستور impulse استفاده می‌شود:

```
>> G = tf([1 0.1 7.5],[1 0.12 9 0 0]);
>> sys=feedback(G,1);
>> subplot(2,1,1)
>> step(sys)
>> subplot(2,1,2)
>> impulse(sys)
```

– آشنایی با متغیرها و توابع پیشرفته ریاضی

برای تعریف متغیر از تابع sym به دو صورت زیر استفاده می‌نماییم:

- روش اول:

```
>> syms x y z a
```

- روش دوم:

```
>> x = sym('x');
>> y = sym('y');
```

در مبحث اعداد مختلط، اگر متغیرها بصورت بالا تعریف شوند، متلب آنها را به عنوان اعداد مختلط درنظر گرفته و درنهایت عدد مختلط تشکیل‌شده از آنها را به عنوان عدد مختلط نمی‌شناسد. برای رفع این مشکل به یکی از دو روش زیر عمل می‌کنیم:

- روش اول:

```
>> syms  x y real
```

- روش دوم:

```
>> x = sym('x','real')
>> y = sym('y','real')
```

- چند مثال:

```
>> syms x y
>> f = x+i*y;
>> conj(f)
>> syms x y real
>> f = x+i*y;
>> conj(f)

>> syms x
>> f = (cos(x))^2+(sin(x))^2;
>> simplify(f)
```

دستور simplify یک عبارت ریاضی را در صورت امکان ساده می‌کند.

```
>> syms x
>> besselj(1/2,x)
>> pretty(ans)
```

دستور pretty خروجی را بصورت استاندارد ریاضی نمایش داده و تابع besselj (بسل) تنها به عنوان یک مثال برای نشان‌دادن کاربرد دستور pretty استفاده شده است.

– مشتق

شکل کلی این دستور بصورت زیر می‌باشد:

$$\frac{d^n f}{du^n} \Rightarrow diff(f,u,n)$$

```
>> syms x a
>> f = sin(a*x);
>> diff(f)
```

همانگونه که در مثال بالا مشاهده می‌شود، مشتق‌گیری بر حسب متغیر x صورت گرفته است. مبنای تعیین اولویت متغیرها در متلب بدین صورت است که ابتدا متغیر x درنظر گرفته می‌شود. اگر x وجود نداشت، متغیری که به x نزدیکتر است، لحاظ شده و اگر فاصله از دو طرف x یکسان بود، مبنا ترتیب حروف الفباست. این قانون در متلب کلی است.

برای تعیین متغیری که می‌خواهیم بر حسب آن مشتق گرفته شود، از آرگومان دوم و برای تعیین مرتبه مشتق از آرگومان سوم استفاده می‌شود. برای مثال برای محاسبه مشتق سوم تابع بالا بر حسب a از دستور زیر استفاده می‌نماییم:

```
>> diff(f,a,3)
```

– حد

شکل کلی این دستور بصورت زیر می‌باشد:

$$\lim_{x \to a} \quad f(x) \Rightarrow limit(f,x,a,'right')$$

```
>> syms x h
>> limit(sin(x),x,pi/2)
>> limit((1+h/x)^x,x,inf)
>> limit(1/x,x,0)
>> limit(1/x,x,0,'right')
>> limit(1/x,x,0,'left')
```

بایستی خاطر نشان کرد که در متلب inf نماد ((بینهایت)) و NaN نماد ((تعریف نشده)) است. از عبارت‌های right و left برای محاسبه حد راست و چپ استفاده می‌شود.

– انتگرال

شکل کلی این دستور بصورت زیر می‌باشد:

$$\int_a^b f(x)dx \Rightarrow \text{int}(f, x, a, b)$$

```
>> syms x n a b
>> int(sin(x))
>> int(cos(x),x,0,2*pi)
>> int(x^n,n)
>> int(sin(x),x,a,b)
```

– سیگما

شکل کلی این دستور بصورت زیر می‌باشد:

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

```
>> syms x
>> symsum(1/x^2,x,1,inf)
```

– بسط و تجزیه یک تابع

برای بسط یک تابع از دستور expand و برای تجزیه یک تابع به عبارت‌های با درجه کمتر از دستور factor استفاده می‌نماییم:

```
>> syms x y
>> expand(sin(x+y))
>> factor(x^2-1)
```

– حل معادلات یک مجهولی

برای حل معادلات یک مجهولی از دستور solve استفاده می‌کنیم.

```
>> syms a b c x y
>> s = a*x^2+b*y+c;
>> solve(s)
```

برای تعیین متغیری که می‌خواهیم براساس آن حل صورت پذیرد، از آرگومان دوم استفاده می‌کنیم:

```
>> solve(s,c)
>> solve(s,b)
```

اگر معادله ما بصورت f(x)=g(x) بود، بصورت زیر عمل می‌کنیم:

```
>> syms x
>> solve('cos(2*x)+sin(x)=1')
>> solve('6*x^3+2*x^2=2*x+1')
```

همانگونه که در خروجی متلب برای دستور آخر مشاهده می‌شود، جواب بصورت واضح نمایش داده نشده است. در اینچنین مواقع از دستور double استفاده نمایید:

```
>> solve('6*x^3+2*x^2=2*x+1')
>> double(ans)
```

برای نمایش یک عدد با تعداد رقم دلخواه از دستور vpa استفاده می‌نماییم. مثلاً برای نمایش عدد π با ۵۰ رقم اعشار بدین صورت عمل می‌کنیم:

```
>> vpa(pi,50)
```

– حل معادلات چند مجهولی

برای حل دستگاه معادلات بصورت زیر عمل می‌نماییم. فرض کنید دستگاه معادلات زیر را داریم:

$$\begin{cases} xy^2 = 0 \\ x - \frac{y}{2} = a \end{cases}$$

برای حل بصورت زیر عمل می‌کنیم:

```
>> syms x y a
>> [x y] = solve('x*y^2=0','x-y/2=a')
```

– حل معادلات دیفرانسیل

برای حل معادلات دیفرانسیل از دستور dsolve و برای تعریف اپراتور مشتق از عبارت D استفاده می‌کنیم.

$y' = 1 + y^2$

```
>> syms y
>> dsolve('Dy=1+y^2')
```

برای تعیین شرایط اولیه و بدست‌آوردن جواب خصوصی معادله دیفرانسیل از آرگومان دوم استفاده می‌کنیم:

```
>> dsolve('Dy=1+y^2','y(0)=1')
```

- مثال

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

```
>> syms x y
>> dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0')
```

همانطور که مشاهده می‌کنید چون متغیری را تعیین نکردیم، متلب فرض می‌کند که y تابعی از t بوده و معادله را بر این اساس حل می‌کند در صورتی که در این مثال مشتق بر حسب x بوده و این جواب اشتباه است. برای رفع این مشکل متغیر x را به عنوان آرگومان آخر وارد می‌کنیم:

```
>> dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0','x')
```

برای حل دستگاه معادلات دیفرانسیل نیز بصورت زیر عمل می‌کنیم:

```
>> syms f g
>> [f g] = dsolve('Df=3*f+4*g','Dg=-4*f+3*g')
```

– تبدیل فوریه و عکس آن

برای تبدیل فوریه از دستور fourier و برای عکس تبدیل فوریه از دستور ifourier استفاده می‌نماییم.

```
>> syms x
>> f = exp(-x^2);
>> fourier(f)
```

متغیر w در خروجی این دستور نماد ω در تبدیل فوریه است.

```
>> syms w
>> f = sin(w)/w;
>> ifourier(f)
```

– تبدیل لاپلاس و عکس آن

برای تبدیل لاپلاس از دستور laplace و برای عکس تبدیل لاپلاس از دستور ilaplace استفاده می‌نماییم.

```
>> syms t
>> f = t^4;
>> laplace(f)
```

متغیر s در خروجی این دستور نماد متغیر تبدیل لاپلاس می‌باشد.

```
>> syms s a
>> f = 1/(s-a)^2;
>> ilaplace(f)
```

– رسم نمودار با استفاده از مفهوم متغیرها

برای رسم نمودارهای دو بعدی از دستور ezplot استفاده می‌نماییم:

```
>> syms x
>> ezplot(sin(x))
```

برای تعیین محدوده رسم شکل، از آرگومان دوم بصورت زیر استفاده می‌کنیم:

```
>> ezplot(sin(x),[-5*pi 10*pi])
```

از این دستور می‌توان برای رسم نمودارهای f(x)=f(y) نیز به دو صورت ezplot(f(x)-f(y)) و ezplot('f(x)=f(y)') استفاده نمود:

$$\sin(x^2) = \cos(y)$$

```
>> syms x y
>> ezplot(sin(x^2)-cos(y))
>> ezplot('sin(x^2)=cos(y)')
```

برای رسم نمودارهای سه بعدی نیز از دستور ezsurf و یا ezmesh استفاده می‌کنیم:

$z = x^2 + y^2$

```
>> syms x y
>> ezsurf(x^2+y^2)
```

– برنامه‌نویسی با متلب

برای نوشتن برنامه‌ها از محیط M فایل استفاده می‌کنیم. این محیط از طریق تب Home گزینه New Script (منوی File > New > M File) قابل دسترسی است.

– حلقه for

این حلقه باعث اجرای قسمتی از برنامه به تعداد معین می‌شود. هنگامی از این دستور استفاده می‌کنیم که بدانیم حلقه چند بار می‌خواهد تکرار شود.

شکل کلی این دستور به صورت زیر می‌باشد:

```
for        شمارنده
           دستورات
end
```

مثال‌های زیر را در محیط M فایل نوشته و اجرا نمایید تا با مفهوم این دستور بیشتر آشنا گردید.

```
clc
clear all
a=zeros(5);
for i=1:5
    a(i,1)=1;
    a(i,3)=3;
    a(i,5)=5;
end
a
```

```
clc
clear all
v=rand(10,1)
n=length(v);
for i=1:n/2
    v(i)=100;
end
for i=n/2+1:n
    v(i)=0;
end
v'
```

```
clc
a=zeros(5);
for i=1:5
    for j=1:5
        a(i,j)=j;
    end
end
a
```

```
tic
for i=0:pi/100:2*pi
    plot(i,sin(i),'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
    hold on
    pause(0.1)
end
toc
```

نکته: از دستور pause برای ایجاد وقفه در اجرای دستور استفاده می‌شود. از عبارت‌های tic و toc برای تعیین مدت زمان اجرای دستورات استفاده می‌گردد. کافی است عبارت tic را در ابتدا و عبارت toc را در انتهای قسمتی که می‌خواهید مدت زمان اجرای آن قسمت را مشاهده نمایید، قرار دهید.

– حلقه while

اگر قبل از اجرای برنامه ندانید که حلقه باید چند بار تکرار شود، از حلقه while استفاده می‌کنیم.

شکل کلی این دستور به صورت زیر می‌باشد:

```
while    عبارت
         دستورات
end
```

حلقه while تا زمانی تکرار می‌گردد که عبارت مربوط به آن نقض شود. عبارت‌های مقایسه‌ای عبارتند از:

عبارت	نماد
مساوی	==
نامساوی	~=
کوچکتر	<
بزرگتر	>
کوچکتر مساوی	<=
بزرگتر مساوی	>=

مثال‌های زیر را در محیط M فایل نوشته و اجرا نمایید تا با مفهوم این دستور بیشتر آشنا گردید.

```
clc
x=0;
while x~=5
    x=input('Enter a Number=');
end
```

```
clc
q='y';
while q~='n'
    i=input('Enter a Number=');
    x=i*i;
    q=input('do you want to continue y/n:','s');
end
disp('program finished')
```

– دستور if

مهمترین دستور شرطی در متلب، دستور if است. شکل کلی این دستور به دو صورت زیر می‌باشد:

```
if      شرط
        دستورات
end
```

```
if      شرط
        دستورات
elseif  شرط
        دستورات
else
        دستورات
end
```

مثال‌های زیر را در محیط M فایل نوشته و اجرا نمایید تا با مفهوم این دستور بیشتر آشنا گردید.

```
clc
clear all
for i=1:10
    if i<=5
        v(i)=0;
    else
        v(i)=1;
    end
end
v'
```

```
clc
clear all
for i=1:10
    a(i)=i;
    if i<=5
        b(i)=i*2;
    else
        b(i)=i*5;
    end
end
[a' b']
```

```
clc
clear all
x=input('please enter a number:');
if x>100
    disp('the number is greater than 100')
else
    disp('the number is less than 100')
end
```

برنامه‌ای بنویسید که حروف n، y و d را تشخیص دهد:

```
ch=input('please enter a character','s');
if ch=='n'
    disp('character is n')
elseif ch=='y'
    disp('character is y')
elseif ch=='d'
    disp('character is d')
else
    disp('character is not n,y,d');
end
```

نکته: دستور max دو خروجی دارد که خروجی اول ماکزیمم عدد موجود در بردار ورودی دستور و خروجی دوم آن شماره درایه متناظر با این مقدار ماکزیمم می‌باشد:

```
clc
clear all
v=rand(15,1)
n=length(v)
[a b]=max(v)
```

– تعریف تابع در متلب

این تکنیک را با دو مثال توضیح می‌دهیم. فرض کنید می‌خواهیم تابعی تعریف کنیم که دو ورودی و دو خروجی دارد. بعنوان مثال مقدار دو مقاومت را به عنوان ورودی دریافت و حاصل سری و موازی آنها را بعنوان خروجی برگرداند.

```
function [RS RP]=calc(R1,R2);
RS=R1+R2;
G=1/R1+1/R2;
RP=1/G;
```

همانگونه که مشاهده می‌کنید RS و RP خروجی‌ها و R1 و R2 ورودی‌های تابعی به نام calc (اسم اختیاری) هستند. در خطوط بعد ارتباط بین ورودی‌ها و خروجی‌ها را تعریف می‌نماییم. سپس باید M فایل را با نام تابع تعریف‌شده ذخیره نمایید. حال می‌توانید از این دستور استفاده کنید. یک M فایل جدید باز کرده و دستور زیر را در آن بنویسید (توجه نمایید که محل ذخیره‌سازی هر دو M فایل در یک پوشه باشد):

```
clc
a=input('R1=');
b=input('R2=');
[series parallel]=calc(a,b)
```

بعنوان مثالی دیگر فرض کنید می‌خواهیم تابعی تعریف کنیم که لگاریتم را در هر پایه‌ای حساب نماید. واضح است که این تابع به دو ورودی برای خود عدد و پایه آن و یک خروجی برای مقدار لگاریتم نیاز دارد:

```
function A=logarithm(a,b);
A=log(a)/log(b);
```

حال برنامه زیر را در یک M فایل جدید نوشته و اجرا نمایید:

```
clc
d=input('Number=');
e=input('Base=');
Result=logarithm(d,e)
```

پیوست:

فهرست نمادها و نحوه واردکردن آنها در شکل‌های متلب

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	Φ	\leq	≤
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamondsuit	♦
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	Θ	\Theta	Θ	\leftrightarrow	↔
\vartheta	ϑ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→
\lambda	λ	\Sigma	Σ	\downarrow	↓
\mu	μ	\Upsilon	Υ	\circ	°
\nu	ν	\Phi	Φ	\pm	±
\xi	ξ	\Psi	Ψ	\geq	≥
\pi	π	\Omega	Ω	\propto	∝
\rho	ρ	\forall	∀	\partial	∂
\sigma	σ	\exists	∃	\bullet	•
\varsigma	ς	\ni	∋	\div	÷
\tau	τ	\cong	≅	\neq	≠
\equiv	≡	\approx	≈	\aleph	ℵ
\Im	ℑ	\Re	ℜ	\wp	℘
\otimes	⊗	\oplus	⊕	\oslash	⊘
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\int	∫	\in	∈	\o	o
\rfloor	⌋	\lceil	⌈	\nabla	∇
\lfloor	⌊	\cdot	·	\ldots	...
\perp	⊥	\neg	¬	\prime	′
\wedge	∧	\times	x	\0	∅
\rceil	⌉	\surd	√	\mid	|
\vee	∨	\varpi	ϖ	\copyright	©
\langle	∠	\rangle	∠		

منابع:

[۱] رضا داژ و محمد آیت، نرم افزار MATLAB 6.1، مؤسسه فرهنگی هنری دیباگران تهران، چاپ دوم، ۱۳۸۳.

[2] The Mathwork Inc., Getting Started with MATLAB 6.