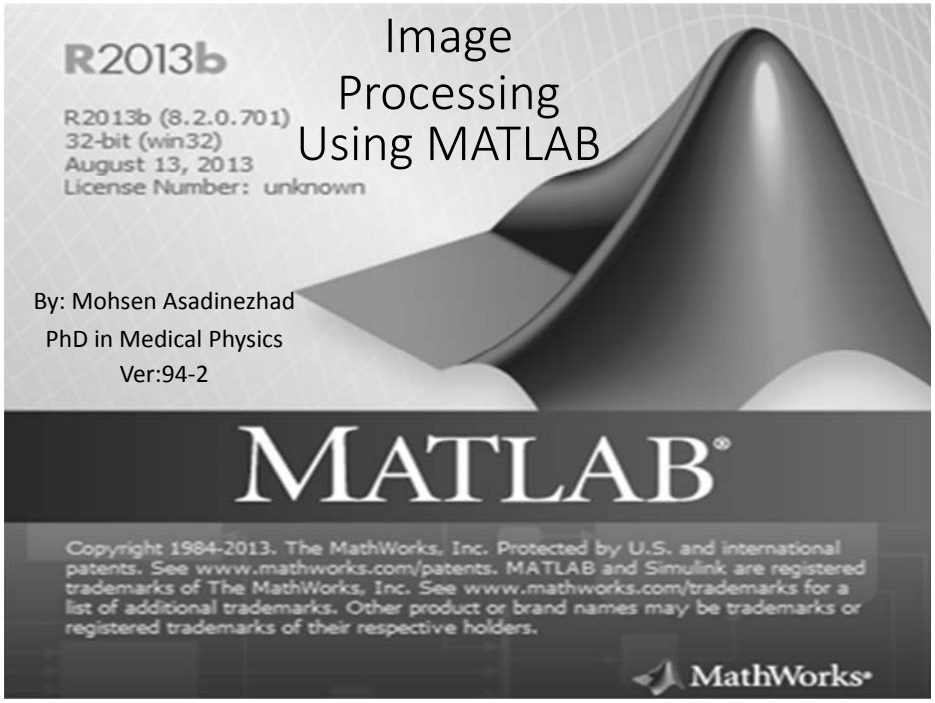


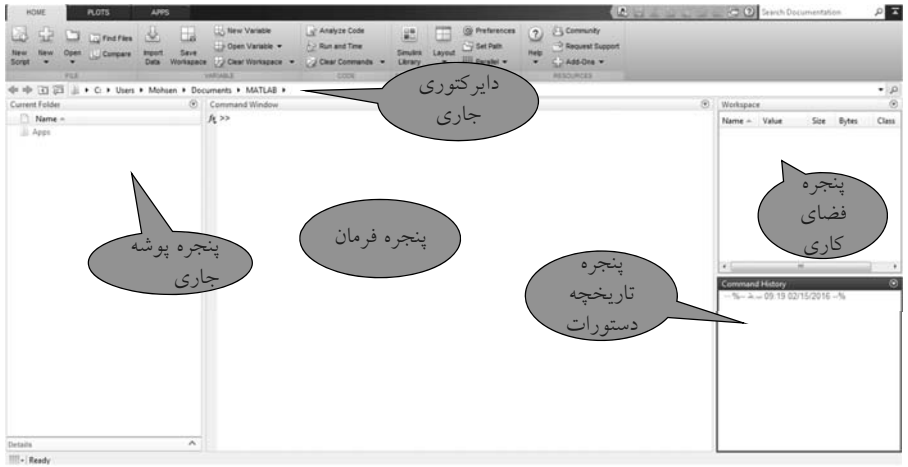
فهرست

فصل اول
ویژگیهای اصلی MATLAB

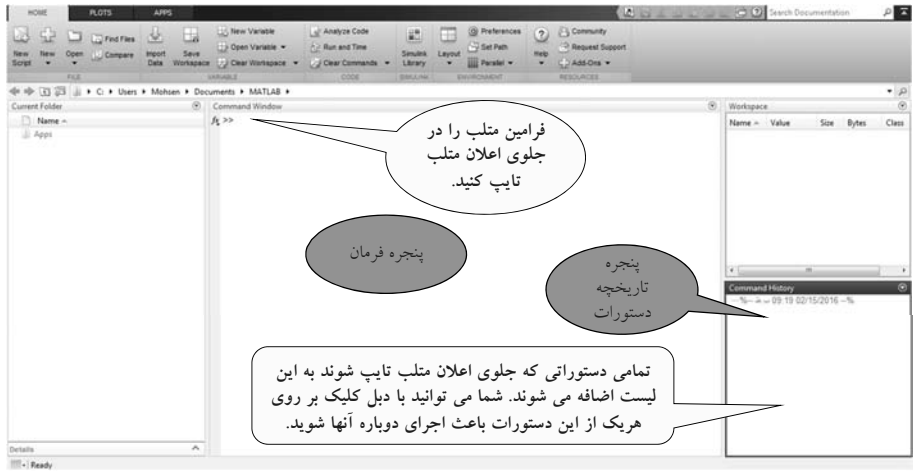
MATrix LABoratory

فصل اول: ویژگیهای اصلی MATLAB

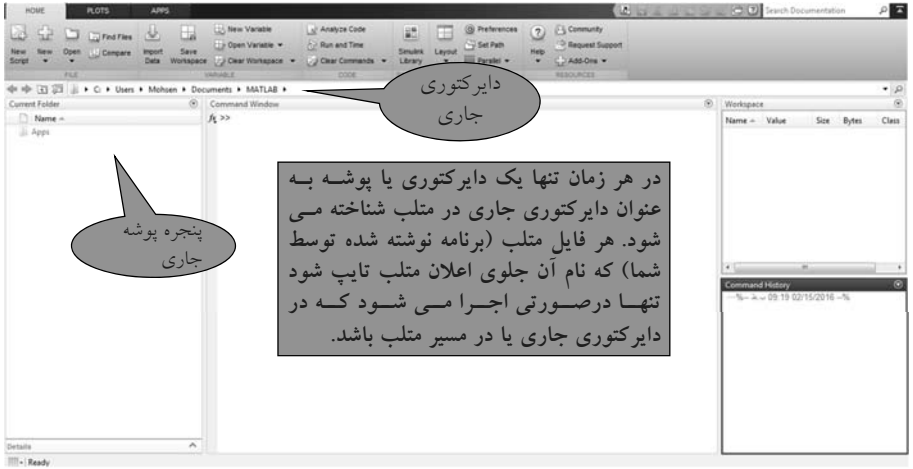
۱-۱- آشنایی با محیط متلب



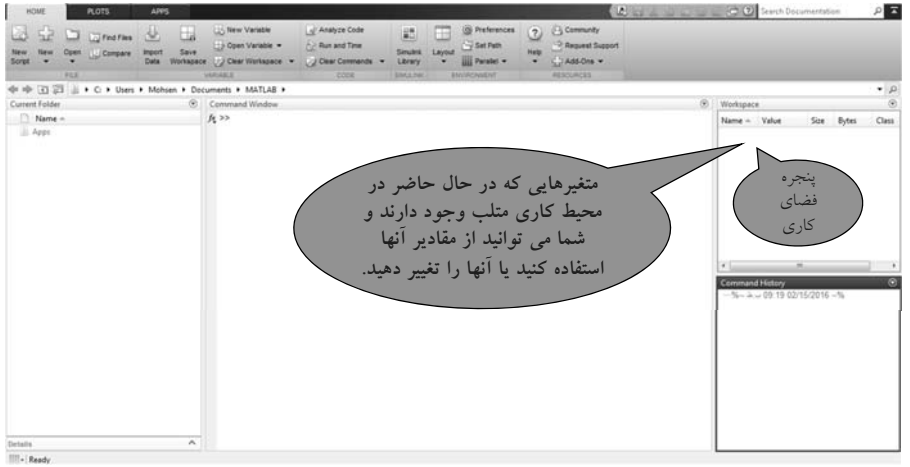
۱-۱- آشنایی با محیط متلب



۱-۱- آشنایی با محیط متلب



۱-۱- آشنایی با محیط متلب



۱-۲- عملیات ریاضی ساده

مثال: محاسبه یک عبارت:

راه اول:

```
>> 4*25 + 6*22 + 2*99
 ans=
          430
```


۱-۲- عملیات ریاضی ساده

مثال: محاسبه یک عبارت:

راه دوم:

```
>> a=25;
>> b=22; c=99;
>> d=4*a+6*b+2*c
 d=
     430
```

نکته ۱: علائم ; و ,

Ctrl-c Escape Tab → ← ↓ ↑

نکته ۲: تعریف متغیرها

نکته ۳: متغیرهای ویژه

۱-۳- عملگرهای ریاضی متلب:

^ \ / * - +

مثال:

```
>>5^2
 ans=
     25
```

/ و \ هر دو عملگر تقسیم میباشند. / تقسیم از چپ و \ تقسیم از راست است. مثلا حاصل 56/8 و 8\56 یکسان است.

- ترتیب حق تقدم: - + < * \ / < ^

۱-٤- فضای کاری متلب

- زمان اعتبار متغیرها:
- دستور who و whos
- ذخیره و بازیابی متغیرها: دستورات save و load

۱-٤-۱- زمان اعتبار متغیرها

متغیرهایی که در فضای کاری تعریف می شوند تنها در دو حالت زیر از حافظه پاک خواهند شد:

- خروج متلب
- استفاده از دستور clear :

>> clear تمامی متغیرها از حافظه پاک می شوند.

>> clear a b c تنها متغیرهای نامبرده شده از حافظه پاک می شوند.

۱-٤-۲- دستورات who و whos

با استفاده از این دو دستور می توان اسامی(و مشخصات) متغیرهای موجود در فضای کاری را بدست آورد.

```
>> who
Your variables are:
  a b c
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array
c	1x1	8	double array

یادآوری: پنجره workspace نیز مشخصات متغیرهای موجود در فضای کاری را مانند دستور whos نشان می دهد.

۱-٤-۳- ذخیره و بازیابی متغیرها: دستورات save و load:

در صورتیکه بخواهیم پس از خروج از محیط متلب همه یا بعضی از متغیرهای موجود در فضای کاری برای استفاده های بعدی ذخیره گردند از دستور save استفاده می کنیم. با دستور load می توان متغیرهای ذخیره شده را به فضای کاری بازگرداند.

```
>> a=5; b=4; c=7;
>> save c:\myfile.mat a c;
>> clear  همه متغیرها پاک می شوند
>> a
??? Undefined function or variable 'a'
>> load c:\myfile.mat
>> a
  a=
       5
>> b
??? Undefined function or variable 'a'
```


۱-٤-۳- ذخیره و بازیابی متغیرها: دستورات save و load:

فرم کلی کاربرد دستورات save و load بصورت زیر است:

save [filename] [variables]

Load [filename] [variables]

درصورتیکه اسم فایل نوشته نشود، فایل پیش فرض matlab.mat مورد استفاده قرار خواهد گرفت و درصورتیکه نام متغیرها نوشته نشود تمامی متغیرهای موجود در فضای کاری ذخیره و یا تمامی متغیرهای ذخیره شده در فایل بازیابی میشوند.

۱-۵-فرمت نمایش اعداد (دستور Format)

با استفاده از این دستور می توان نحوه نمایش اعداد در پنجره فرمان متلب را تغییر داد.

>> Format [option]

Option: short, long, short e, long e, short g, long g, bank, hex, + , …

دقت کنید که این دستور دقت محاسبات را تغییر نمی دهد و تنها بر نحوه نمایش اعداد تاثیر خواهد گذاشت.

۱-۵-فرمت نمایش اعداد (دستور Format)

>> a=100/3

>> format [option]

Option	example
short	33.3333
long	33.333333333333336
short e	3.3333e+01
Long e	3.333333333333334e+01
Short g	33.333
Long g	33.3333333333333
bank	33.33
hex	4040aaaaaaaaaab

۱-٦- انواع متغیرها

بعضی از مهمترین انواع متغیر در متلب:

double : نقطه اعشار با دقت مضاعف (۸ بایت)

struct : نوع تعریف شده توسط کاربر

single : نقطه اعشار (٤ بایت)

uint8 : عدد صحیح بی علامت ۸ بیتی (uint: unsigned integer)

uint16 : عدد صحیح بی علامت ١٦ بیتی

uint32 : عدد صحیحی بی علامت ۳۲ بیتی

uint64 : عدد صحیحی بی علامت ٦٤ بیتی

int8 : عدد صحیح ۸ بیتی

int16 : عدد صحیح ١٦ بیتی

int32 : عدد صحیح ۳۲ بیتی

int64 : عدد صحیح ٦٤ بیتی

برای دیدن لیست کامل انواع متغیر در متلب در پنجره فرمان از دستور help datatypes استفاده کنید

۱-٦- انواع متغیرها

• باید دقت کرد که اگرچه متلب انواع مختلفی از متغیرها را پشتیبانی می کند اما نوع پیش فرض، نوع "دقت مضاعف" است و برای تبدیل نوع یک متغیر باید دستور کلی زیر را بکار برد:

a=TypeName(a);

در اینجا نوع متغیر به صحیح بی علامت ۸ بیتی تغییر می کند. ;>> a=uint8(a)

در اینجا یک متغیر از ابتدا از نوع صحیح بی علامت ۳۲ بیتی تعریف شده است ;>> b = uint32(345)

• دقت: در هنگام تبدیل یا ایجاد یک متغیر باید دقت کنید که مقدار انتساب داده شده خارج از دامنه مقادیر آن نوع خاص نباشد. برای انواع صحیح می‌توانید از دستور زیر برای تعیین دامنه استفاده کنید:

>> intmin('int16')

>> intmax('int16')

• استثناء: در مورد جعبه ابزار پردازش تصویر نوع پیش فرض نوع uint8 است.

۱-۷- نامگذاری متغیرها

- فقط یک کلمه مثل average_cost
- اختلاف حروف کوچک و بزرگ
- با حرف الفبا باید شروع شود.
- کاراکترهای مجاز: حروف الفبا، اعداد و _
- حداکثر طول نام: با استفاده از تابع namelengthmax در هر نسخه از MATLAB می‌تواند تعیین شود. در نسخه ۲۰۱۳، حداکثر ۶۳ کاراکتر است.
- مراقب باشید متغیر شما با یک تابع درونی MATLAB یا تابعی که توسط خود شما نوشته شده است هم نام نباشد.
- مثال:

```
>>This_Is_a_Variable=5;
```


۱-۸- متغیرهای ویژه

- متغیرهای زیر در محیط متلب بصورت پیش‌فرض وجود دارند.

ans NaN nargin beep pi i nargout
eps j inf

- اگر مقدار متغیر ویژه تغییر کند با دستور clear به مقدار اولیه برمی گردد.
- برخی کلمات رزرو شده: for end if while function return elseif case otherwise switch continue else try catch global persistent break
- می توان کلمات رزرو را با حروف بزرگ نوشت.

۱-۹- علائم نقطه‌گذاری و جملات توضیحی

- برای درج یک متن توضیحی در برنامه‌های متلب باید از کاراکتر % استفاده شود.

```
>> a=5; %"a" is a variable
```

- برای نوشتن ادامه یک جمله در سطر بعد باید از ... استفاده کرد:

```
>> b=a+a^2+...
     3*a^3;
```


۱-۱۰- راهنمای متلب

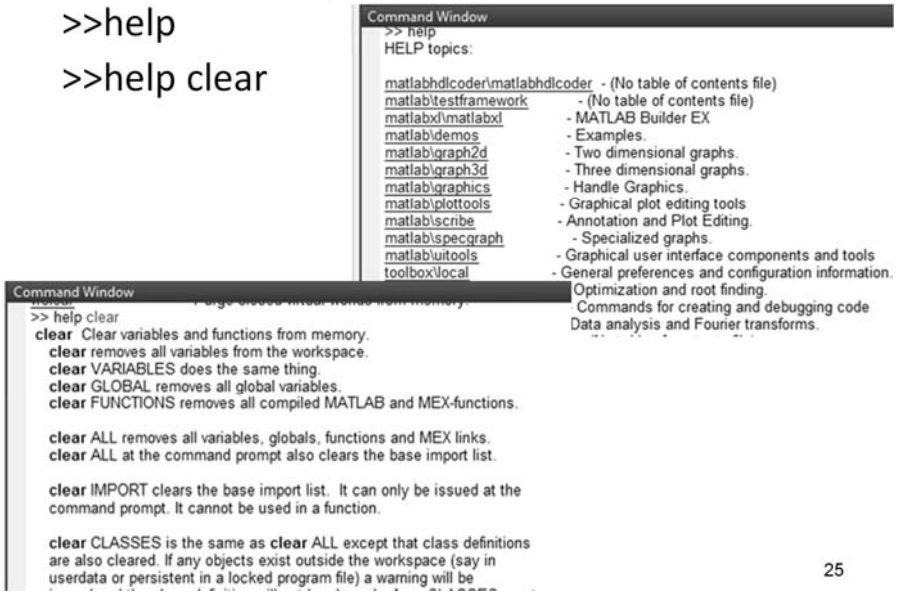
متلب دارای دستورات راهنمای متفاوتی است که هم از طریق منوی start و هم از طریق اعلان متلب قابل دسترسند.

```
>>demo
```

۱-۱۰- راهنمای متلب

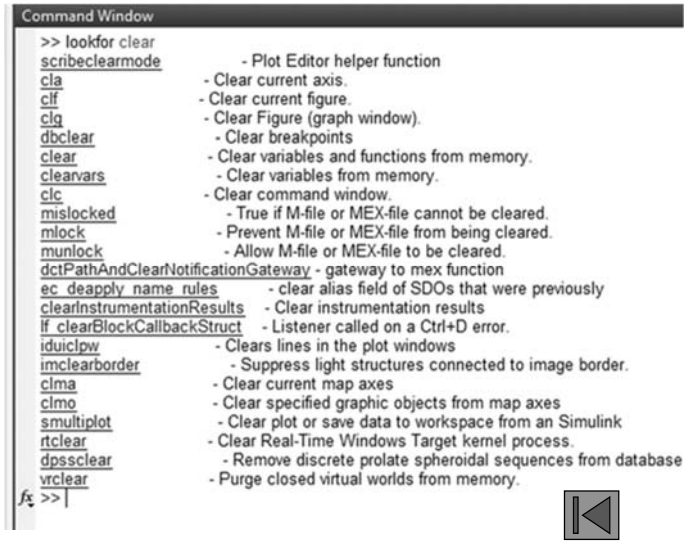
>>help

>>help clear



۱-۱۰- راهنمای متلب

>>lookfor …



۱-۱۱- بعضی از توابع ریاضی در متلب

مختلط	مثلثاتی
abs	sin(x)
angle	cos(x)
imag	tan(x)
real	cot(x)
	asin(x)
	acos(x)
	atan(x)
	acot(x)
	sind(x)
	asind(x)

- تمرین: توابع روبرو را در متلب استفاده کنید و با استفاده از help متلب عملکرد آنها را بررسی کنید.

۱-۱۱- بعضی از توابع ریاضی در متلب

تابع نمایی	عملکرد تابع	مثال
exp	e به توان X	exp(1)= 2.7183
log	لگاریتم نپرین	log(2)= 0.6931
log10	لگاریتم پایه ده	log10(100)= 2
log2	لگاریتم پایه دو	log2(8)= 3
pow2	دو به توان X	pow2(4)= 16
sqrt	جذر X	sqrt(64)= 8

۱-۱۱- بعضی از توابع ریاضی در متلب

تابع گردکردن و باقیمانده	عملکرد تابع	مثال
fix	گرد کردن به سمت صفر	fix(C)= 6 fix(B)= -4
round	گردکردن به سمت نزدیکترین عدد صحیح	round(D)= -9 round(A)= 2
floor	گردکردن به سمت منفی بینهایت	floor(D)= -9 floor(A)= 2
ceil	گردکردن به سمت مثبت بینهایت	ceil(A)= 3 ceil(D)= -8
rem	باقیمانده تقسیم بدون علامت	rem(C,B)= 2.2200
mod	باقیمانده تقسیم با علامت	mod(C,B)= -2.2200
sign	تابع علامت	sign(C)= 1 sign(B)= -1

A=2.22
B=-4.44
C=6.66
D=-8.88

۱-۱۲- فایلهای متنی(Script) یا فایلهای m

بمنظور اجرای چند دستور بطور همزمان و بدون نیاز به تایپ مجدد، از فایلهای متنی استفاده می‌شود.

این فایلها باید دارای پسوند m باشند.

۱-۱۲-۱- مراحل ایجاد فایلهای متنی

1. باز کردن یک فایل جدید در ویرایشگر متلب:

Home>New>Script

یا Ctrl-n

2. تایپ کردن دستورات متلب در فایل مذکور
3. ذخیره کردن فایل با نامی مشخص:

Editor>Save As…

۱-۱۲-۲- روش اجرای یک فایل متنی

برای اجرای یک فایل متنی کافی است نام آنرا در جلوی اعلان متلب تایپ کرده کلید Enter را بزنیم.

مثال: برنامه sample1.m

```
% SAMPLE1: A Simple m-file
n=10;a=2;b=4;
c=n*a^3/b
```

```
>> sample1
 c=
     20
```

۱-۱۲-۳- توابع و دستورات مفید در فایلهای m

1. تابع disp(x): این تابع مقدار یک متغیر یا یک رشته متنی را نمایش می‌دهد.

مثال:

```
>> n=10;
>> disp(n)
    10
>> disp('This is a string')
    This is a string
```


۱-۱۲-۳- توابع و دستورات مفید در فایلهای m

2. تابع x=input(s): برای گرفتن مقدار یک متغیر از ورودی.

مثال:

```
n=input('Please tell me "n" value: ')
-----------------------------------------------
Please tell me "n" value: 10
n=
    10
```


۱-۱۲-۳- توابع و دستورات مفید در فایلهای m

3. دستور pause: توقف موقت در حین اجرا.

```
pause
pause(n)  % n seconds
```

مثال:

```
%SAMPLE2: Enhanced Sample1
n=10;
a=input(' "a" value= ');
b=input(' "b" value= ');
c=n*a^3/b;
disp('Please wait 5 seconds only!');pause(5);
disp('Press any key to see answer.'); pause;
disp(' "C" Value is= '); disp(c)
```


تکلیف ۱-۱

برنامه‌ای بنویسید که با گرفتن وزن (بر حسب کیلوگرم) و قد (بر حسب متر) از کاربر، شاخص توده بدن BMI(Body Mass Index) را محاسبه کند و مقدار آن را با پیغام مناسب نمایش دهد.

- با تایپ نام برنامه در جلوی اعلان MATLAB، آنرا اجرا کنید.
- با استفاده از ویرایشگر MATLAB، برنامه خود را اجرا و trace کنید.

۱-۱۳- مدیریت فایل: کار کردن با فایلها و شاخه‌ها

بعضی از دستورات مفید:

- دستور cd: تغییر و یا نمایش شاخه جاری :

```
>>cd
C:\Matlab\Work
>>cd C:\MyDir
>>cd
C:\MyDir
```

- دستور dir: نمایش نام فایلها و زیرشاخه‌های دایرکتوری جاری
- دستور delete: حذف(پاک کردن) فایل:

```
>>delete sample1
```

فصل دوم
آرایه‌ها

فصل دوم: آرایه ها

1. ایجاد آرایه
2. عملیات ریاضی بر روی آرایه‌ها
3. ترانهاده یک ماتریس
4. بکاربردن توابع ریاضی بر روی آرایه‌ها
5. استخراج بخشی از آرایه
6. حذف بخشی از آرایه
7. جستجوی زیرآرایه
8. اندازه آرایه
9. چند تابع برای دستکاری آرایه‌ها

۲-۱- ایجاد آرایه

چند تعریف:

اسکالر Scalar (آرایه دو بعدی ۱*۱)

بردار Vector (آرایه دو بعدی 1*n یا m*1، سطری یا ستونی)

ماتریس Matrix (آرایه دو بعدی m*n)

روشهای ایجاد آرایه:

1. با استفاده از علائم [ ] , ;
2. با استفاده از علامت :
3. با استفاده از توابع linspace و logspace
4. با استفاده از ترکیبی از روشهای فوق
5. ماتریس های ویژه

۲-۱-۱- با استفاده از علائم ; , [ ]

- از علامت ; برای تعیین سطر جدید و از علامت , برای تعیین ستون جدید استفاده می‌شود.
- مثال:

```
>> a=[1,2,3;4,5,6]
a=
     1 2 3
     4 5 6
>> b=[1,2,3,4,5,6]
b=
     1 2 3 4 5 6
```


۲-۱-۱- با استفاده از علائم ; , [ ]

نکته: بجای علامت ; از enter و بجای علامت , از فاصله خالی نیز می‌توان استفاده کرد

مثال:

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


۲-۱-۲- ایجاد آرایه با استفاده از علامت :

- در مواقعی که عناصر یک آرایه رابطه خطی با یکدیگر داشته باشند از این روش می‌توان استفاده کرد.
- شکل کلی دستور بصورت زیر است:

```
ArrayName=first : step : last
```

- اگر step حذف شود، مقدار ۱ بجای آن بکار خواهد رفت.
- اگر last کوچکتر از first باشد، باید step منفی باشد. در غیر اینصورت مقدار آرایه تهی خواهد شد.

۲-۱-۲- ایجاد آرایه با استفاده از علامت :

- مثال:

```
>> x=(0 : 0.1 : 1) * pi;
>> y=sin(x);

>>z=1:5
z=
     1 2 3 4 5
>>t=5:1
t =
  Empty matrix: 1-by-0
```

۲-۱-۳- ایجاد آرایه با استفاده از توابع linspace و logspace

- با ارائه عناصر اول و آخر و طول آرایه به این توابع می‌توان آرایه‌هایی خطی و یا لگاریتمی بدست آورد.

```
ArrayName=linspace(first,last,length)
```

- مثال:

```
>>x=linspace(0,1,11)*pi;
>>y=logspace(1,3,3)
y=
     10  100  1000
```


۲-۱-٤- ایجاد آرایه با استفاده از ترکیبی از علائم فوق

- مثال:

```
>>x=[0,1,2, 4:2:12 ,18,19]
x=
     0 1 2  4 6 8 10 12 18 19
>> y=[10,1,7,4,6,-1 ; linspace(0,10,6) ; 5:-1:0]
y=
     10  1  7  4  6  -1
     0   2  4  6  8  10
     5   4  3  2  1   0
```


۲-۱-۵- ماتریس های ویژه

- [ ] : ماتریس تهی
- eye : یک ماتریس یکه با ابعاد داده شده ایجاد می‌کند.
- ones : یک ماتریس که تمامی عناصر آن یک می‌باشند با ابعاد داده شده ایجاد می‌کند.
- zeros : یک ماتریس صفر با ابعاد داده شده ایجاد می‌کند.
- rand : یک ماتریس با عناصر راندوم با توزیع یکنواخت به ابعاد داده شده ایجاد می‌کند.
- randn : یک ماتریس با عناصر راندوم با توزیع نرمال به ابعاد داده شده ایجاد می‌کند.

۲-۱-۵- ماتریس های ویژه

مثال:

```
>> ones(2,3)
ans =

   1   1   1
   1   1   1
>> ones(2)
ans =

   1   1
   1   1
```

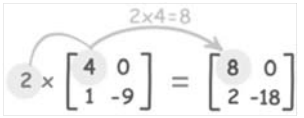
تمرین: سایر توابع فوق را خودتان آزمایش کنید.

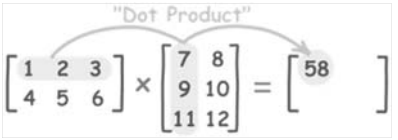
۲-۲- عملیات ریاضی بر روی آرایه‌ها

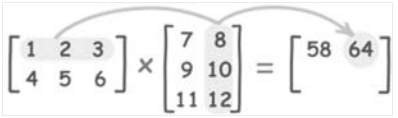
عملگر	نام	تابع	توضیحات و مثال
+	جمع ماتریسی و آرایه ای	plus(A,B)	A+b, A+B, a+A
-	تفریق ماتریسی و آرایه ای	minus(A,B)	a-b, A-B, a-A
.*	ضرب آرایه ای	times(A,B)	C=A.*B, C(i,j)=A(i,j)*B(i,j)
*	ضرب ماتریسی	mtimes(A,B)	A*B, a*A
./	تقسیم آرایه ای راست	rdivide(A,B)	C=A./B, C(i,j)=A(i,j)/B(i,j)
.\	تقسیم آرایه ای چپ	ldivide(A,B)	C=A.\B, C(i,j)=B(i,j)/A(i,j)
/	تقسیم ماتریسی راست	mrdivide(A,B)	A/B, A*inv(B)
\	تقسیم ماتریسی چپ	mldivide(A,B)	A\B, inv(A)*B
.^	توان آرایه ای	power(A,B)	C=A.^B, C(i,j)=A(i,j)^B(i,j)
^	توان ماتریسی	mpower(A,B)	مراجعه به راهنمای MATLAB
.'	ترانهاده بردار و ماتریس	transpose(A)	ترانهاده استاندارد بردار و ماتریس
'	ترانهاده مزدوج بردار و ماتریس	ctranspose(A)	ترانهاده مزدوج استاندارد بردار و ماتریس

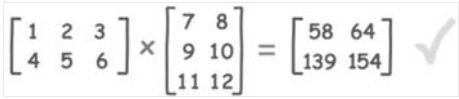
توجه: A,B ماتریس
a,b مقادیر عددی

How to Multiply Matrices?









۲-۲-۱- عملیات ریاضی اسکالر-آرایه

- با استفاده از عملگرهای ریاضی متلب براحتی می‌توان عملیات ریاضی اسکالر-آرایه را انجام داد.
- مثال:

```
>> x=[1 2 3;4 5 6; 7 8 9];
>> y=2*x + 4
y=
     6     8    10
    12    14    16
    18    20    22
```


۲-۲-۲- عملیات ریاضی عنصری بین دو آرایه

- بدین منظور باید دو آرایه حتما هم بعد باشند.
- مثال:

```
>> a=[2 4 6; 3 5 6; 10 -1 0];
>> b=[-1 0 0; 2 1 1; 0 0 3];
>> c= (2*a ./ (b+1)) .^ 2
c =
   Inf    64   144
     4    25    36
   400     4     0
```

۲-۳- ترانهاده یک ماتریس

برای محاسبه ترانهاده یک ماتریس از علامت تک کوتیشن ' استفاده می‌شود.

مثال:

```
>> a=[2 1 7
     4 5 -1
     6, 6, 0];
>> b=a'
   2    4    6
   1    5    6
   7   -1    0
```


۲-٤- بکاربردن توابع ریاضی بر روی آرایه‌ها

- توابع متلب بصورت ماتریسی عمل می‌کنند. یعنی لازم نیست تابعی مانند sin را یک به یک بر روی عناصر یک آرایه اعمال کرد. بلکه براحتی می‌توان با یک دستور مقدار سینوس کل عناصر آرایه را محاسبه نمود.
- مثال:

```
>> a=[2 4 6; 3 5 6; 10 -1 0];
>> SinA=sin(abs(a) / 10)
SinA =
   0.1987   0.3894   0.5646
   0.2955   0.4794   0.5646
   0.8415   0.0998        0
```


تمرین ۲-۱

1. برنامه ای بنویسید که عدد صحیح n را از کاربر بگیرد و برداری ۱۰۰ عنصری بین ۰ و $2n\pi$ ایجاد نموده در متغیر x قرار دهد. سپس مقادیر y را از رابطه زیر محاسبه کرده نمایش دهد:

$y=|\sin(x)|*x^2$

2. برنامه فوق را طوری تغییر دهید که علاوه بر مقدار n، عددی بین ۰ و ۱ را نیز از کاربر بگیرد و در متغیر جدید d قرار دهد. سپس بردار x را بین ۰ و $2n\pi$ اما با گامهایی برابر با d محاسبه نماید.

۲-۵- استخراج بخشی از آرایه

m2=m1 (آرایه‌ای از اندیس‌ها,آرایه‌ای از اندیس‌ها)

- مثال:

```
>> a=[1 2 3
     4 5 6
     7 8 9];
>> k1=[1,2];k2=[2,3];
>> b=a(k1,k2)
b=
     2    3
     5    6
```

۲-۵- استخراج بخشی از آرایه

```
>>c=a([1 2 3],[1,3])
c=
     1     3
     4     6
     7     9
>>d=a([3,2],[3,1])
d=
     9     7
     6     4
```


۲-۵- استخراج بخشی از آرایه

```
>>e=a([1,2,3],2)
e=
     2
     5
     8
>>f=a(1:2:3 , 3:-2:1)
f=
     3     1
     9     7
```


۲-۵- استخراج بخشی از آرایه

```
>>g=a(1:3 , 1:2)
g=
     1     2
     4     5
     7     8
>>h=a(1:2:3, : )
h=
     1     2     3
     7     8     9
```


۲-۵- استخراج بخشی از آرایه

```
>> k=a( : , : )
k=
     1     2     3
     4     5     6
     7     8     9
>>l=a(1:end,end)
l=
     3
     6
     9
```

۲-۵- استخراج بخشی از آرایه

نکته:

```
>>n=a([1 1 1] , :)
n=
     1    2    3
     1    2    3
     1    2    3
>>m=a( : , [3 3 3 3])
m=
     3    3    3    3
     6    6    6    6
     9    9    9    9
```


۲-۵- استخراج بخشی از آرایه

نکته:

```
>>p=a( : )
p=
     1
     4
     7
     2
     5
     8
     3
     6
     9
```


تمرین ۲-۲

1. ماتریس سمت راست را بدون وارد کردن مستقیم عناصر ایجاد کنید.

2. ماتریسی شامل ستونهای سوم تا هشتم و سطرهای چهارم تا نهم ماتریس فوق ایجاد کنید.

۲-۶- حذف بخشی از آرایه

- به منظور حذف بخشی از یک آرایه می‌توان ماتریس تهی را به آن بخش نسبت داد:

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

>>a(1 : 2 , : ) = [ ]
a=
     7    8    9
```

۲-۷- جستجوی زیرآرایه

- به منظور یافتن عناصری از آرایه که در شرط خاصی صدق می‌کنند می‌توان از دستور find استفاده کرد (این دستور عناصر را بصورت ستونی شمارش می‌کند):

```
>> a=[   1   2   3
         4   5   6
         7   8   9];
>> k=find( a > 5 )
k=
     3
     6
     8
     9
```


۲-۷- جستجوی زیرآرایه

```
>>b=a(k)
b=
     7
     8
     6
     9
```


۲-۷- جستجوی زیرآرایه

دستور findدر صورتیکه با دو آرگومان خروجی بکار برده شود، شماره سطر و ستون عناصر را باز می‌گرداند:

```
>>[k1,k2]=find( a > 5)
k1=          k2=
     3            1
     3            2
     2            3
     3            3
```


۲-۸- اندازه آرایه

- با استفاده از دستورات length و size می‌توان ابعاد یک آرایه را بدست آورد.
- دستور length اگر بر روی یک بردار بکار برده شود، تعداد عناصر آنرا باز می‌گرداند و اگر بر روی یک ماتریس بکار رود، بزرگترین بعد آنرا باز می‌گرداند.
- دستور size انعطاف‌پذیرتر بوده و می‌تواند به روشهای زیر بکار برده شود:
  - اگر با یک آرگومان ورودی بکار برده شود، طول و عرض ماتریس را بازمی‌گرداند.
  - اگر با دو آرگومان ورودی بکار برده شود، بطوریکه آرگومان دوم ۱ یا ۲ باشد، بترتیب تعداد سطرها یا ستونهای ماتریس را بازمی‌گرداند.
  - اگر با یک آرگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را در یک بردار سطری دو عنصری بازمی‌گرداند.
  - اگر با دو ارگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را بترتیب در آرگومان اول و دوم باز می‌گرداند.

۲-۸- اندازه آرایه

• مثال:

```
>> a=[1    2     3     4
      5    6     7     8];
>> size(a)
ans=
     2     4
>> [r , c] = size(a)
r =
     2
c=
     4
```

```
>> r=size(a , 1)
r=
     2
```

```
>> c=size(a,2)
c=
     4
```


۲-۸- اندازه آرایه

• مثال:

```
>> b=[1    2     3     4];
>> l=length(b)
l=
     4
>> a=[1    2     3     4
      5  6     7     8];
>> la=length(a)
la=
     4
```


۲-۹- چند تابع برای دستکاری آرایه‌ها

flipud : ماتریس را حول محور افقی ۱۸۰ درجه می‌چرخاند.

fliplr : ماتریس را حول محور عمودی ۱۸۰ درجه می‌چرخاند.

rot90 : ماتریس را در جهت مثلثاتی ۹۰ درجه می‌چرخاند

Diag : در صورتیکه بر روی یک ماتریس بکاربرده شود، قطر اصلی ماتریس را استخراج می‌کند. اما اگر بر روی یک بردار بکار رود، ماتریسی قطری با عناصر آن بردار می‌سازد

تمرین ۲-۳

1. برنامه ای بنویسید که ماتریسی دو ستونی را که مقادیر ستون اول آن نمرات دروس مختلف یک ترم یک دانشجو و مقادیر ستون دوم آن تعداد واحد مربوط هر یک از آن دروس می باشد را از کاربر بگیرد و عملیات زیر را بر روی انجام دهد:
   - محاسبه تعداد واحدها
   - محاسبه معدل ترم
   - نمایش نتایج با پیغام مناسب

فصل سوم
توابع و عملیات ماتریسی

1. حل دستگاه معادلات خطی
2. تعدادی از توابع ماتریسی

۳-۱- حل دستگاه معادلات خطی

با استفاده از عملیات ضرب و تقسیم ماتریسی در متلب براحتی می‌توان دستگاههای معادلات خطی را حتی در مواردی که تعداد معادلات با تعداد متغیرها مساوی نباشند، حل کرد. بدین‌منظور باید بردار سمت راست معادلات را بر ماتریس ضرایب متغیرها تقسیم کرد.

۳-۱- حل دستگاه معادلات خطی

مثال:

$$\begin{cases} x+2y+3z=366 \\ 4x+5y+6z=804 \\ 7x+8y=351 \end{cases}$$

```
>>a=[1  2  3          >>b=[366 ; 804 ; 351];
     4  5  6
     7  8  0];
>> x= a \ b   یا   >>x=a ^ (-1) * b   یا   >>x=inv(a) * b
x=
  25
  22
  99
```

۳-۲- تعدادی از توابع ماتریسی

- det : دترمینان ماتریس را محاسبه می‌کند
- inv : معکوس ماتریس را محاسبه می‌کند
- pinv: شبه معکوس یک ماتریس غیرمربعی را محاسبه می‌کند
- trace: مجموع عناصر قطر اصلی یک ماتریس را بازمی‌گرداند

تکلیف۳-۱: دستگاه معادلات خطی زیر را حل کنید و بهترین جواب را بدست آورید:

$$\begin{cases} x+2y+3z+7t=4 \\ 6x+7y+22z+32t=5 \\ 98x+5y-23z+t=7 \\ 32x+5y-75z+23t=1 \\ 22x+2y+3z+t=0 \end{cases}$$

تکلیف ۳-۲: برنامه‌ای بنویسید که ماتریس ضرایب و مقادیر سمت راست یک دستگاه معادلات خطی را از کاربر بگیرد و پاسخ دستگاه را با پیغام مناسب نمایش دهد.

فصل چهارم
عملیات منطقی و رابطه‌ای

فصل چهارم: عملیات منطقی و رابطه‌ای

- تعریف عملیات منطقی

1. عملگرهای رابطه‌ای
   1. مقایسه دو آرایه
   2. مقایسه یک آرایه با یک عدد
2. عملگرهای منطقی
3. توابع رابطه‌ای و منطقی

تعریف عملیات منطقی

عملیاتی که بر اساس مقادیر منطقی true و false (یا ۰ و ۱) استوار باشد را عملیات منطقی می‌گویند.

٤-١- عملگرهای رابطه‌ای

عملگر	نام
<	کوچکتر از
<=	کوچکتر مساوی با
>	بزرگتر از
>=	بزرگتر مساوی با
==	مساوی با
~=	نامساوی با

٤-١-١- مقایسه دو آرایه

مقایسه عنصر به عنصر دو آرایه با یکدیگر:

به ازای نقاطی که در شرط ذکر شده صدق می‌کنند، مقدار ۱ و به ازای سایر نقاط مقدار ۰ باز گردانده می‌شود.

```
>> a= [1 , 2, 3 , 4 , 5];
>>b=[10 , 2 , 13 , 4 , 8];
>>c=a == b
c=
  0    1      0      1      0
```

متغیر c یک متغیر از نوع منطقی(logical) خواهد بود. یعنی تنها می‌تواند مقادیر ۰ و ۱ را در خود نگهدارد.

به عنوان تمرین سعی کنید عنصر سوم c را با ۵۰ جایگزین کنید.

٤-١-٢- مقایسه یک آرایه با یک عدد

در این حالت تمامی عناصر آرایه با یک عدد مقایسه می‌شوند:

```
>> a = [1 , 2 , 3 ; 4 , 2 , 2 ; 1 , 10 , 0];
>> t= a >=2
t=
     0   1  1
     1   1  1
     0   1  0
```

مثال: استخراج عناصری از یک ماتریس که در شرط خاصی صدق می‌کنند

```
>> a = [1 , 2 , 3 ; 4 , 2 , 2 ; 1 , 10 , 0];
>> b = a .* (a >= 3)
b=
     0    0    3
     4    0    0
     0   10    0
```


تکلیف ٤-١

برنامه‌ای بنویسید که نمرات دروس ریاضی (٢ واحد)، فارسی (٣ واحد) و معارف اسلامی (٢ واحد) چند دانشجو را بصورت یک ماتریس(n*3) از کاربر بگیرد و موارد زیر را محاسبه و با پیغام مناسب نمایش دهد:

- تعداد دانشجویان
- معدل هر دانشجو
- معدل هر درس
- معدل کل دروس برای تمامی دانشجویان (یک عدد)
- میانگین نمرات زیر ١٠ بدون احتساب واحد هر درس

٤-٢- عملگرهای منطقی

عملگر	نام
&	AND
|	OR
~	NOT

٤-٢-عملگرهای منطقی

مثال:

```
>> a= 1 : 9;
>> t = a > 3
  0     0     0     1     1     1     1     1     1

>> f = ~ ( a > 3)
  1     1     1     0     0     0     0     0     0

>> tf = ( a > 3) & (a <=7)
  0     0     0     1     1     1     1     0     0
```

٤-٣- توابع رابطه‌ای و منطقی

علاوه بر عملگرهای رابطه‌ای و منطقی در متلب توابعی نیز بدین منظور وجود دارد که عبارتند از:

all (x) درصورتیکه تمامی عناصر یک بردار غیر صفر باشد مقدار ۱ و در غیر اینصورت ۰ باز می‌گرداند.

any(x) درصورتیکه حداقل یکی از عناصر یک بردار غیر صفر باشد مقدار ۱ و در غیر اینصورت ۰ باز می‌گرداند.

xor(x,y) یای انحصاری، در صورت متفاوت بودن دو عملوند منطقی، ۱ و در غیر این صورت ۰ باز می گرداند.

٤-٣- توابع رابطه‌ای و منطقی

مثال:

```
>>x=[1 1 0];
>>y=[0 1 0];
>>tor= x | y
tor=
  1   1   0
```

```
>>txor=xor(x , y)
txor=
  1   0   0
```


فصل پنجم
متن: کار با رشته‌های کاراکتری

٤-٣- توابع رابطه‌ای و منطقی

مثال:

```
>>a= [1 1 1 0];
>>t=any(a)
 t=
     1
```

```
>>t=all(a)
 t=
     0
```

```
>>a=[3 2 4];
>>t=any(a==2)
t=
 1
```

فصل پنجم :متن: کار با رشته‌های کاراکتری

1. رشته‌های کاراکتری
2. نمایش کد اسکی کاراکترها: تابع abs
3. تبدیل کد اسکی به کاراکتر
4. رفتار ماتریسی رشته‌ها
5. ایجاد ماتریسهای کاراکتری
6. گرفتن رشته در حین اجرای برنامه
7. سایر توابع کار با رشته‌ها

۵-۱-رشته‌های کاراکتری

برای تعریف رشته‌های کاراکتری در متلب از علامت ‘ ’ استفاده می‌شود:
مثال:

```
>> s='This is a character string';
>> size(s)
ans=
     1    26
```

نکته: در متلب رشته‌های کاراکتری نیز بعنوان ماتریس شناخته می‌شوند بطوریکه هر کاراکتر یک عنصر ماتریس محسوب می‌شود.

۵-۲-نمایش کد اسکی کاراکترها: تابع abs

برای نمایش کد اسکی یک رشته می‌توان از تابع abs متلب استفاده کرد:

```
>> s= 'Hello'
>> u=abs(s)
  u=
       72  101   108   108  111
```


۵-۳-تبدیل کد اسکی به کاراکتر

برای تبدیل کد اسکی به کاراکتر از تابع char استفاده کنید.

```
>> s= 'Hello'
>> u=abs(s)
  u=
       72  101   108   108  111
>> sNew=char(u)
sNew=
  Hello
```

۵-٤-رفتار ماتریسی رشته‌ها

با رشته‌های کاراکتری متلب دقیقا می‌توان مانند ماتریسهای عددی رفتار کرد. مثلا می‌توان عملیات ریاضی را بر آنها اعمال کرد. در اینصورت متلب کد اسکی رشته را مورد استفاده قرار می‌دهد.

مثال: نمایش رشته از آخر به اول

```
>> s= 'Hello'
>> sInv=s( end : -1 : 1);
>>disp(sInv)
  olleH
```


۵-۵-ایجاد ماتریسهای کاراکتری(روش اول)

برای ایجاد یک ماتریس کاراکتری می‌توان از علائم [ ] و ; مانند ایجاد ماتریس‌های عددی استفاده کرد. اما باید دقت شود که تعداد ستونهای هر سطر مساوی باشند:

```
>> sm=['This is first line' ; 'This is second line']
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
>> sm=['This is first line '   یک فاصله خالی در انتهای خط
      'This is second line'];
```


۵-۵-ایجاد ماتریسهای کاراکتری(روش دوم)

روش بهتر برای ایجاد یک ماتریس کاراکتری استفاده از تابع **char** می‌باشد:

```
>> line1='This is first line' ;
>> line2= 'This is second line';
>>sm=char(line1,line2)
sm=
   This is first line
   This is second line
```


۵-٦-گرفتن رشته در حین اجرای برنامه

برای گرفتن یک رشته از ورودی با استفاده از تابع **input** در حین اجرای برنامه دو روش را می‌توان بکار برد:

روش اول روش معمول استفاده از این تابع است. یعنی تابع مذکور را تنها با یک آرگومان ورودی بکار می‌بریم. در اینصورت در حین اجرا، باید رشته را در داخل ' ' قرار داد.

روش بهتر استفاده از تابع **input** با یک آرگومان دوم 's' می‌باشد که در اینصورت متلب ورودی کاربر را بعنوان رشته تلقی می‌کند حتی اگر یک عدد یا نام یک متغیر باشد.

۵-۶-گرفتن رشته در حین اجرای برنامه

مثال:

```
>>s=input('Please answer Yes or No: ')
Please answer Yes or No: 'No'
s=
  No
----------------------
>>s=input('Please answer Yes or No: ','s')
Please answer Yes or No: No
s=
  No
```


۵-۷- سایر توابع کار با رشته‌ها

strcmp(s1,s2)	درصورتیکه دو رشته یکسان باشند ۱ و درغیر اینصورت ۰ باز می‌گرداند :
upper	تمای حروف یک رشته را به حروف بزرگ تبدیل می‌کند :
lower	تمامی حروف یک رشته را به حروف کوچک تبدیل می‌کند:
num2str	تبدیل عدد به رشته عددی:
str2num	تبدیل رشته عددی به عدد:
mat2str	تبدیل ماتریسی از اعداد به رشته:
eval	اجرای فرمانی از متلب که بصورت رشته وارد شده باشد:

- نکته: تفاوت تابع num2str با تابع mat2str در این است که در تابع دوم رشته بازگردانده شده قابل اجرا توسط تابع eval است.

۵-۷- سایر توابع کار با رشته‌ها

مثال:

```
>> a=input('Enter <a> value= ');
  enter <a> value= 12

>> disp(['You number is', num2str(a) , ' . Thank
                                you!']);

Your number is 12 . Thank you!
```


تکلیف

تکلیف ۵-۱: برنامه‌ای بنویسید که دو ماتریس عددی را از کاربر بگیرد و در متغیرهای x و y قرار دهد. سپس یک رشته کاراکتری شامل عبارتی ریاضی از متغیرهای x و y را از کاربر بگیرد و نتیجه آنرا بر اساس مقادیر متغیرهای ورودی تعیین کند.

تکلیف ۵-۲: برنامه‌ای بنویسید که یک رشته کاراکتری را از کاربر بگیرد و با تغییر کد اسکی آن، آنرا بصورت رمز در آورده نمایش دهد.

تکلیف ۵-۳: برنامه‌ای بنویسید که نتایج تمرین ۵-۲ را از حالت رمز خارج کرده و نمایش دهد.

فصل ششم:
تصمیم‌گیری و کنترل روند،
استفاده از حلقه‌ها و دستورات شرطی

فصل ششم: تصمیم‌گیری و کنترل روند،
استفاده از حلقه‌ها و دستورات شرطی

1. حلقه for
2. حلقه while
3. ساختار if-else-end
4. ساختار switch-case-end

٦-١- حلقه for

- شکل کلی حلقه for در متلب بصورت زیر است:

```
for  x= آرایه
 دستورات
end
```

- جهت تکرار دستورات بین for و end به تعداد معین
- حلقه فوق به تعداد ستونهای آرایه مشخص‌شده تکرار خواهد شد و در هر تکرار یکی از ستونهای این آرایه در متغیر x قرار گرفته و در بدنه حلقه قابل استفاده است.
- در صورتیکه آرایه یک بردار باشد، هر بار یک عنصر از آن در متغیر x قرار خواهد گرفت.
- تذکر: با توجه به تواناییهای ماتریسی متلب از کاربرد حلقه‌ها در متلب تا حد ممکن باید پرهیز گردد زیرا اینکار باعث کند شدن شدید برنامه می‌شود و نیاز به کد نویسی بسیار بیشتری دارد.

٦-١- حلقه for

مثال: برنامه ای بنویسید که مجموع اعداد بین ٥٠ تا ٣٠٠٠ را حساب کند.

```
s=0;
for i=50:3000
  s=s+i;
end
disp(s)
--------------
for k=[1,2,3,7]
  x(k) = k+1;
end;
>>x
x=
  2  3  4  0  0  0  8
```

۶-۱- حلقه for

تمرین: برنامه ای بنویسید که مجموع اعداد زوج بین ۱۰۰ تا ۴۰۰۰ را حساب کند.

۶-۲- حلقه while

- جهت تکرار دستورات بین **while** و **end** به تعداد نامعین تا برقراری شرطی خاص
- شکل کلی حلقه **while** بصورت زیر است:

```
while  شرط
  دستورات
end
```

حلقه فوق تا زمانیکه شرط ذکر شده برقرار باشد تکرار خواهد شد.

۶-۲- حلقه while

مثال: محاسبه ده فاکتوریل

```
x=0;y=1;
while  x<10
   x=x+1;
   y=y*x;
end
>>y
y=
  3628800
```

```
>> f=1;
>> for n=2:10
       f=f*n;
   end
>> f
```


۶-۳- ساختار if-else-end

هرگاه بخواهیم یک یا چند جمله در صورت برقرار بودن شرط خاصی (یکبار) اجرا شود، از بلوک **if** استفاده می‌کنیم. شکل کلی استفاده از این دستور بصورت زیر است:

```
if شرط ۱
  دستورات
elseif شرط ۲
  دستورات
elseif …
  …
else
  دستورات
end;
```

۶-۳- ساختار if-else-end

مثال: برنامه ای بنویسید که دو عدد را از کاربر دریافت کند و عدد بزرگتر را به نمایش بگذارد. چنانچه دو عدد با هم برابر باشند برنامه دو عدد دیگر دریافت کند.

```
clc; clear all
a=input('please enter the first number');
b=input('please enter the second number');
while a==b
    clear a b
    a=input('please enter the first number');
    b=input('please enter the second number');
end
if a>b;
    disp([num2str(a),' is bigger than ', num2str(b)])
else
    disp([num2str(b),' is bigger than ', num2str(a)])
end
```


۶-۳- ساختار if-else-end

- نکته: continue و break
- این دو دستور داخل حلقه های for یا while قرار می گیرند و باعث جهش هایی می شوند.
- continue عمل جهش به ابتدای حلقه را بدون احرای سطرهای باقیمانده انجام می دهد.
- break باعث خروج از حلقه در حال اجرا می شود به عبارت دیگر برنامه به اولین خط پس از دستور end می جهد.
- اغلب این دو دستور تحت یک عبارت شرطی با if آورده می شوند که اگر شرط if برقرار باشد این دستورات به عنوان محتویات زیر if اجرا گردیده و موجب جهش گردند.

۶-4- ساختار switch-case-end

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

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

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


تکلیف

تکلیف ۶-۱: حقوق قابل پرداخت به یک کارگر با ۴۰ ساعت کار در هفته، حقوق اصلی است. به ازای هر ساعت اضافه کاری، ۵۰٪ بیشتر از ساعات عادی مزد می گیرد. برنامه ای بنویسید که از کاربر تعداد ساعات کاری و حق الزحمه ساعتی را دریافت و حقوق کاربر را حساب کند.

تکلیف ۶-۲: برنامه ای بنویسید که سه عدد را از کاربر دریافت کند و عدد بزرگتر را نمایش دهد. (با فرض اینکه سه عدد با هم برابر نیستند)

تکلیف

تکلیف ٦-٣: برنامه‌ای بنویسید که نمرات چند دانشجو را به صورت یک بردار بگیرد و عملیات زیر را انجام دهد:

- در صورتیکه ورودی کاربر بردار نباشد(ماتریس یا اسکالر باشد) پیام خطا دهد.(راهنمایی برای دادن پیام خطا می‌توانید از تابع error به جای disp استفاده کنید)
- با استفاده از حلقه for و دستورات شرطی if-else-end تک تک نمرات را چک کند و به صورت زیر آنها را تغییر دهد:
  - نمرات کمتر از ۵ را به ۹ تغییر دهد
  - نمرات بین ۵ و ۸ را به ۹,۵ تغییر دهد.
  - نمرات بین ۸ و ۱۰ را به ۱۰ تغییر دهد.
  - نمرات بین ۱۰ و ۱۵ را ۱ نمره افزایش دهد
  - نمرات بیشتر از ۱۵ و کمتر از ۲۰ را ۰,۵ نمره افزایش دهد.

تکلیف ٦-٤: برنامه دیگری بنویسید که همان کارهای برنامه ٦-٣ را بدون استفاده از حلقه انجام دهد.

فصل هفتم:
ایجاد توابع در متلب
Functions

۷-۱-مزایای استفاده از توابع به جای فایلهای اسکریپت

1. سرعت بالاتر
2. صرفه‌جویی در حافظه کامپیوتر
3. توسعه توانایی‌های متلب

توابع بر خلاف فایلهای اسکریپت در هنگام اجرا یکبار کامپایل شده و اجرا می‌شوند. در حالیکه فایلهای اسکریپت سطر به سطر کامپایل و اجرا می‌گردند. این امر باعث افزایش سرعت اجرای توابع در مقایسه با فایلهای اسکریپت می‌شود.

متغیرهای تعریف شده در توابع پس از پایان اجرای آن از حافظه پاک می‌شوند و بطور کلی فضای کاری توابع مستقل از فضای کاری متلب است. خصوصا در مواقعی که برنامه با ماتریسهای بزرگ (مانند تصاویر) کار می‌کند بهتر است از توابع استفاده شود.

۷-۱-مزایای استفاده از توابع به جای فایلهای اسکریپت

اکثر دستورات اصلی متلب و جعبه‌ابزارهای آن با استفاده از توابع نوشته شده است. به بیان دیگر به راحتی می‌توان قابلیتهایی که در حال حاضر در متلب وجود ندارد را با نوشتن یک مجموعه از توابع به آن افزود. همین امر باعث شده است که در دهه گذشته قابلیتهای متلب در رشته‌های مختلف علمی و فنی با سرعت چشمگیری توسعه یابد.

نکته : بهتر است در هنگام نوشتن یک برنامه آنرا بصورت اسکریپت بنویسیم تا اشکال زدایی آن آسانتر باشد اما پس از کامل شده برنامه آنرا به فانکشن تبدیل کنیم تا سرعت و کیفیت آن افزایش یابد.

۷-۲- تفاوتهای توابع و فایلهای متنی

1. فایلهای متنی سطر به سطر ترجمه و اجرا می‌شوند اما توابع یکبار بطور کامل ترجمه و سپس اجرا می‌گردند.
2. محیط کاری فایلهای متنی همان محیط کاری متلب است اما محیط کاری هر تابعی مختص خود اوست یعنی اگر متغیری در یک تابع تعریف شود تنها در آن تابع قابل دسترسی است و برعکس متغیرهای تعریف شده در محیط کاری متلب در داخل توابع تعریف شده نیستند.(مگر اینکه بصورت عمومی تعریف شده باشند.)
3. توابع تنها از طریق آرگومانهایشان با محیط خارج در ارتباطند.

۷-۳- نحوه ایجاد توابع

تنها تفاوت ظاهری یک تابع و یک فایل متنی آن است که سطر اول یک تابع با کلمه کلیدی function شروع می‌شود که شکل کلی آن بصورت زیر است:

```
function [argout1 , argout2, ... ] = funcname(argin1,argin2,...)
معرفی فانکشن در یک سطر%
راهنمای استفاده%
از این فانکشن%

نویسنده فانکشن ، نسخه و سال ساخت %
  بدنه تابع
  ...
```


۷-۳- نحوه ایجاد توابع

نکات:

1. کلمه function یک کلمه کلیدی متلب است و همیشه برای تعریف توابع از آن استفاده می شود.
2. همیشه آرگومانهای خروجی در کروشه و آرگومانهای ورودی در پرانتز قرار می گیرند.
3. اگر تابع دارای یک خروجی باشد می توان کروشه ها را حذف کرد.
4. تابع ممکن است هیچ آرگومان ورودی یا خروجی نداشته باشد.
5. اگر تابع خروجی نداشته باشد تنها از کلمه function قبل از نام تابع استفاده می شود و کروشه ها و علامت مساوی حذف می شوند.

۷-۳- نحوه ایجاد توابع

ادامه نکات:

1. نام تابع با یک حرف شروع و شامل ترکیب حروف، اعداد و علائم و بدون فاصله است تا ٦٤ کاراکتر.
2. اولین سطر بعد از اعلان تابع، یک جمله توضیحی است که در هنگام استفاده از دستور lookfor در متلب مورد جستجو قرار می‌گیرد.
3. تمامی سطرهای توضیحی تا نخستین سطر غیر توضیحی در هنگام استفاده از دستور help نمایش داده میشود.

نکته :بهتر است هنگام نوشتن یک تابع حتما یکی دو سطر در مورد نحوه استفاده از آن و عملکرد آن توضیح داده شود تا کاربر بتواند با استفاده از دستور help متلب با روش استفاده از آن تابع و قابلیتهای آن آشنا شود.

۷-٤- فرمانهای return و error

با استفاده از این دو دستور می‌توان اجرای یک تابع را پیش از رسیدن به انتهای آن متوقف کرد. تفاوت دستور error با دستور return آن است که دستور error می تواند یک پیغام خطا نیز بمنظور آگاه سازی کاربر نمایش دهد.

مثال:

```
s= input( 'Please enter a scalar value= ');
if  length (s) > 1
  error('Error! Your input isn''t a scalar!');
end
a= linspace( 0 , abs(s) , 100);
```


۷-۵- تعیین تعداد آرگومانهای بکار رفته در حین اجرا

در متلب می‌توان توابع را با تعداد آرگومان کمتر از تعداد آرگومان موجود در تعریف تابع نیز فراخوانی کرد. مثلا تابع size در متلب با دو آرگومان نوشته شده است اما با یک آرگومان نیز قابل اجراست که البته مقدار بازگشتی به تعداد آرگومانهای مورد استفاده بستگی خواهد داشت.

در صورتیکه بخواهیم از تعداد آرگومانها در حین اجرا مطلع شویم باید از توابع nargin و nargout به ترتیب برای تعداد آرگومانهای ورودی و تعداد آرگومانهای خروجی استفاده کنیم.

همچنین توابع nargchk و nargoutchk تعداد آرگومانهای ورودی و خروجی را چک می‌کنند و درصورتیکه با تعداد درخواست شده برابر نباشند پیام خطای مناسب را نشان می‌دهند.

۷-٦- نکاتی در مورد توابع

- در یک فایل می‌توان بیش از یک تابع تعریف کرد. در اینصورت تمامی این توابع می‌توانند یکدیگر را فراخوانی کنند اما تنها نخستین تابع از خارج از این فایل قابل فراخوانی است.
- نام فایل با نام نخستین تابع آن باید یکسان باشد. در غیر اینصورت بمنظور اجرای تابع باید از نام فایل به جای نام تابع استفاده گردد که البته کار درستی نیست.

مثال

مثال ۷-۱- تابعی بنویسید که یک بردار(آرایه سطری یا ستونی) را از کاربر بگیرد و مراحل زیر را انجام دهد:

- تعداد آرگومان ورودی و خروجی که توسط کاربر وارد شده است را چک کند و در صورتیکه تعداد آرگومان ورودی بیشتر یا کمتر از یک و تعداد آرگومان خروجی بیشتر از یک باشد، پیام خطا نمایش داده از تابع خارج شود.
- ابعاد آرگومان ورودی را چک کند و درصورتیکه آرایه‌ای غیر سطری یا غیر ستونی باشد (یعنی در صورتیکه به جای بردار، ماتریس باشد)، با پیام خطا از تابع خارج شود.
- عبارت زیر را بر روی مقادیر ورودی اعمال نموده به عنوان خروجی بازگرداند.

$$y=2\exp(4x^2)+3\sin(2\pi x)+10$$

- تعداد آرگومان خروجی را چک کند و در صورتیکه برابر با صفر باشد، نمودار تغییرات y در مقابل x را رسم کند. (راهنمایی: برای رسم نمودار از تابع plot(x,y) استفاده کنید.

تکلیف

تکلیف ۷-۱-تابعی بنویسید که یک عبارت ریاضی دلخواه را از کاربر (به صورت یک رشته کاراکتری) به عنوان آرگومان اول و یک آرایه را به عنوان آرگومان دوم بگیرد و:

- چک کند که تعداد آرگومان ورودی دقیقا دو عدد باشد (با استفاده از تابع nargchk)
- چک کند که تعداد آرگومان خروجی دقیقا یک عدد باشد. (با استفاده از تابع nargoutchk)
- چک کند که آرگومان اول حتما یک رشته کاراکتری باشد و آرگومان دوم حتما یک متغیر عددی. (از توابع isstr و isnumeric استفاده کنید)
- با استفاده از تابع eval عبارات ریاضی وارد شده توسط کاربر را بر روی تمامی عناصر آرایه ورودی اعمال نموده، باز گرداند.

فصل هشتم:
تجزیه و تحلیل فوریه

فصل هشتم: تجزیه و تحلیل فوریه

1. تبدیل سریع فوریه
2. مثالی از کاربرد تبدیل فوریه

۸-۱-تبدیل سریع فوریه

کاربرد: استخراج سیگنالی خاص از سیگنالی مرکب از چندین سیگنال.

توابع پرکاربرد: fft , ifft , fft2 , ifft2

```
>> fx = fft(x)          تبدیل فوریه
>> fx = fft(x,n)        تبدیل فوریه در n نقطه
>> fsx = abs( fft(x) )  طیف فوریه
>> psx = ( fft(x) ) .^ 2  طیف توان
>> x = ifft( fx )       عکس تبدیل فوریه
>> x = ifft( fx , n )   عکس تبدیل فوریه در n نقطه
```


۸-۲-مثالی از کاربرد تبدیل فوریه

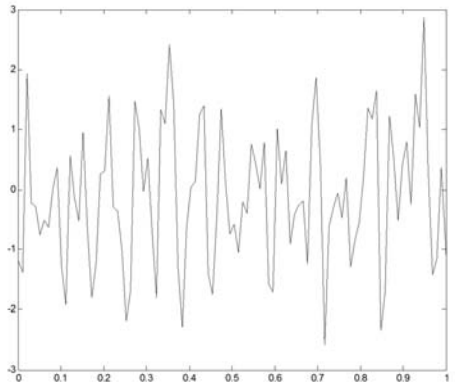
ابتدا سیگنالی مرکب از دو سیگنال متناوب و راندوم (نویز) ایجاد می‌کنیم(واضح است که در شرایط واقعی این سیگنال از طریق آزمایش بدست می‌آید)

```
>> t= 0 : 1/99 : 1;   بردار زمان
```

```
>> y= sin ( 2*15 * pi * t) + randn(size(t));   سیگنالی با فرکانس ۱۵ هرتز که با یک سیگنال نویز ترکیب شده است
```


۸-۲-مثالی از کاربرد تبدیل فوریه-ادامه

```
>> plot(t , y);   رسم نمودار تغییرات سیگنال در حوزه زمان
```



۸-۲-مثالی از کاربرد تبدیل فوریه-ادامه

اکنون فرض می‌کنیم که سیگنال فوق را در اختیار داشتیم و می‌خواستیم بخش متناوب آنرا استخراج کنیم:

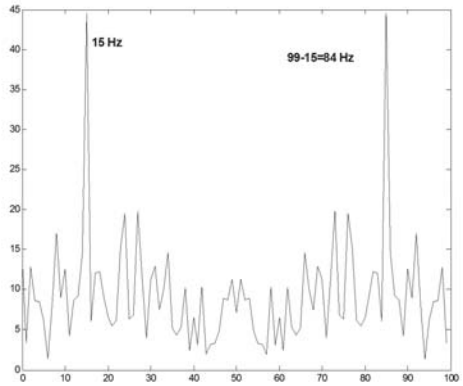
```
>> fy = abs ( fft(y) );
>> f  = linspace(0 , 99 , length(y) );
```

در این رابطه ۹۹ فرکانس نمونه‌برداری است و در واقع ماکزیمم فرکانسی است که شدت آن در طیف فوریه وجود دارد.

f: بردار فرکانس است که بین ۰ تا ۹۹ تغییر می کند.

۸-۲-مثالی از کاربرد تبدیل فوریه-ادامه

نمودار طیف فوریه: >>plot(f , fy)



فصل نهم:
نمودارهای دو بعدی

فصل نهم: نمودارهای دو بعدی

1. تابع plot
2. رسم چند نمودار مجزا در یک پنجره شکل
3. برچسب گذاری محورهای افقی و عمودی و عنوان
4. رسم خطوط شبکه‌ای بر روی نمودار
5. ایجاد پنجره شکل جدید
6. افزودن متن به نمودار
7. افزودن راهنمای علائم
8. دستور axis
9. ثابت نگهداشتن نمودار
10. سایر دستورات
11. سایر نمودارهای دوبعدی

۹-۱-تابع plot

شکل کلی:

plot (x1,y1,'c1s1',x2,y2,'c2s2,x3,y3,'c3s3',...)

در این رابطه، sn می‌تواند هر یک از کاراکترهای زیر باشد:

. , o , x , + , - , * , -. , -- , penta , hexa

و cn نیز می‌تواند یکی از رنگهای زیر باشد:

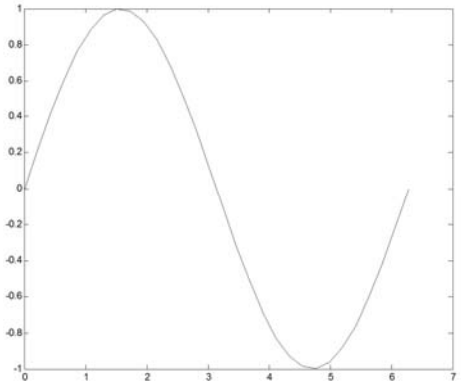
y , m , c , r , g , b , w , k

که به ترتیب معرف رنگهای زرد، سرخابی، فیروزه‌ای، قرمز، سبز، آبی، سفید و سیاه می‌باشد

۹-۱- تابع plot

مثال:

```
>> x= linspace(0,2*pi , 30); y= sin(x);
>> plot(x,y);
```



۹-۲- رسم چند نمودار مجزا در یک پنجره شکل

بمنظور تقسیم پنجره شکل به چند بخش می‌توان از تابع subplot استفاده کرد.

شکل کلی:

subplot(m ,n , p)

در این رابطه m تعداد بخشهای افقی، n تعداد بخشهای عمودی و p شماره بخش جاری است. هر دستور ترسیمی بعداز این دستور در مکان p ام اعمال خواهد شد. خانه‌ها بصورت ستونی شمارش می‌شوند.

واضح است که مقدار p باید بین ۱ و m*n باشد در غیر اینصورت متلب اعلان خطا می‌کند.

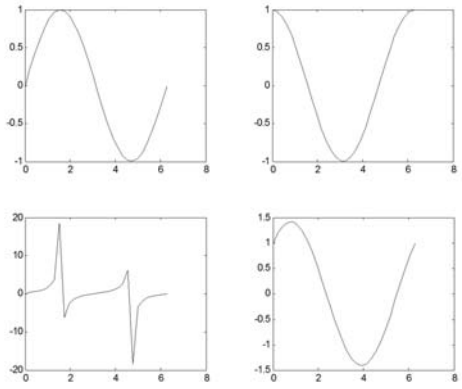
۹-۲- رسم چند نمودار مجزا در یک پنجره شکل

مثال:

```
>> x=linspace(0,2*pi,30);
>> subplot(2,2,1);plot(x,sin(x));
>> subplot(2,2,2);plot(x,cos(x));
>> subplot(2,2,3);plot(x,tan(x));
>> subplot(2,2,4);plot(x,sin(x)+cos(x));
```


۹-۲- رسم چند نمودار مجزا در یک پنجره شکل

مثال:-ادامه-



۹-۳- برچسب گذاری محورهای افقی و عمودی و عنوان

بمنظور برچسب‌گذاری محورها و ایجاد عنوان برای نمودار می‌توان از توابع xlabel, ylabel , title استفاده کرد.

```
>> xlabel(‘یک رشته متنی ’);
>> ylabel(‘یک رشته متنی ’);
>> title(‘یک رشته متنی ’);
```

این دستورات بر روی آخرین نمودار ترسیم شده اعمال میشوند بنابراین بعد از هر دستور plot یا دستور ترسیمی دیگر بلافاصله باید از این دستورات استفاده گردد.

۹-٤- رسم خطوط شبکه‌ای بر روی نمودار

بمنظور ایجاد خطوط شبکه‌ای (چهارخانه‌های نقطه‌چین) بر روی یک نمودار، می‌توان از دستور grid استفاده کرد. شکل کلی استفاده از دستور grid به صورتهای زیر است:

```
>> grid on      حالت شبکه‌ای را فعال می‌کند
>> grid off     حالت شبکه‌ای را غیر فعال میکند
>> grid
```

حالت شبکه‌ای را از فعال به غیرفعال و از غیر فعال به فعال تغییر می‌دهد.

۹-۵- ایجاد پنجره شکل جدید

بصورت پیش‌فرض در متلب هر نمودار جدید جایگزین نمودار قبلی در همان پنجره شکل میگردد. در صورتیکه بخواهیم چند نمودار در پنجره‌های شکل جداگانه ترسیم شوند از دستور figure استفاده می‌کنیم

```
>> figure;
```

این دستور باعث می‌شود که یک پنجره شکل جدید باز شده و نمودار بعدی در آن پنجره ترسیم گردد.

۹-٦- افزودن متن به نمودار

با استفاده از توابع text و gtext می‌توان متنی را به نمودار اضافه کرد:

```
>> text(x,y,’رشته متنی’)
>> gtext(‘رشته متنی’)
```

دستور اخیر اجازه می‌دهد که ناحیه قرار گیری رشته متنی را بتوان با ماوس انتخاب کرد.

۹–۷– افزودن راهنمای علائم

دستور legend

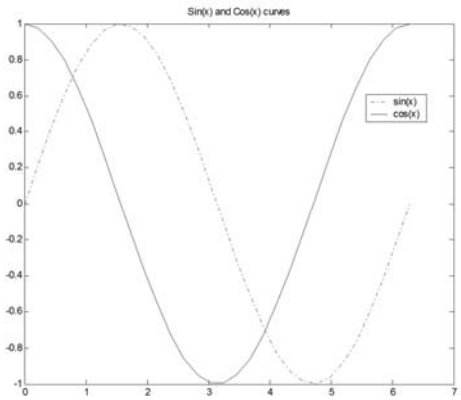
مثال:

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
plot(x,y,'g-.','b-');
legend('sin(x)','cos(x)');
title('Sin(x) and Cos(x) curves);
```


۹ –۷– افزودن راهنمای علائم

دستور legend

مثال-ادامه:



۹–۸– دستور axis

با استفاده از این دستور می‌توان دامنه ترسیم را تغییر داد:

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

نمودار در دامنه xmin تا xmax ، ymin تا ymax و... ترسیم می‌گردد.

---

axis off محورهای مختصات را حذف می‌کند

axis on محورهای مختصات را ترسیم می‌کند

۹–۹– ثابت نگهداشتن نمودار

دستور hold

بصورت پیش‌فرض متلب هر نمودار جدید را جایگزین نمودار قبلی میکند، اگر بخواهیم بدون پاک شدن نمودار فعلی نمودار جدیدی اضافه کنیم باید از دستور hold استفاده نماییم:

hold on فعال

hold off غیر فعال

hold تغییر حالت

۹-۱۰- سایر دستورات

clf: محتویات پنجره شکل جاری را پاک می‌کند

cla: محتویات نمودار جاری را پاک میکند

zoom: حالت زوم را فعال یا غیر فعال می‌کند

ginput:

برای گرفتن مختصات یک یا چند نقطه از نمودار با استفاده از ماوس

۹-۱۱- سایر نمودارهای دوبعدی

علاوه بر plot دستورات ترسیم نمودارهای دوبعدی دیگری نیز در متلب وجود دارد که عبارتند از:

polar: ترسیم نمودار در مختصات قطبی

fill: ترسیم نواحی بسته دو بعدی (چندضلعی‌ها)

semilogx, semilogy, loglog:

ترسیم نمودار در مختصات لگاریتمی

stairs: ترسیم نموار پله‌ای

hist: ترسیم نمودار فراوانی

bar: ترسیم نمودار میله‌ای

فصل دهم:
پردازش تصویر

فصل دهم: پردازش تصویر

1. مقدمه
2. کلاس های داده ای
3. فرمت های گرافیکی تصاویر
4. انواع تصویر
5. خواندن تصاویر- تابع imread
6. نمایش تصاویر- تابع imshow
7. نمایش تصاویر- تابع imtool
8. نوشتن فایلهای گرافیکی-imwrite
9. تعیین مشخصات یک فایل گرافیکی- تابع imfinfo
10. تبدیل تصاویر

فصل دهم: پردازش تصویر

11. عملیات ریاضی بر روی تصاویر
12. عملیات هندسی بر روی تصاویر
13. فیلترهای خطی و طراحی فیلتر
14. آنالیز و بهسازی تصویر
15. عملیات بر روی تصاویر باینری

۱۰-۱- مقدمه

- در متلب "جعبه ابزار پردازش تصویر" Image Processing Toolbox حاوی مجموعه ای از توابع است که توانائی های متلب را در عملیات پردازش تصویر توسعه می دهد.
- در متلب تصاویر بصورت ماتریسهای دو، سه و یا چهاربعدی تعریف می‌شوند.
- کیفیت تصویر: به دو پارامتر، دقت ابعادی و دقت عمقی در هنگام تصویربرداری و یا ذخیره‌سازی تصویر بستگی دارد.
  - دقت عمقی(Depth): منظور از دقت عمقی تعداد بیتهایی است که از حافظه کامپیوتر به هر نقطه (پیکسل) از تصویر اختصاص داده می‌شود.
  - دقت ابعادی (Resolution): منظور تعداد نقاط نمونه‌برداری شده در واحد طول یا عرض تصویر است. دقت ابعادی افقی و عمودی یک تصویر ممکن است متفاوت باشند اما معمولا چنین نیست. واحد دقت ابعادی dpi یا نقطه بر اینچ است.

۱۰-۲- کلاس های داده ای

نام	توضیح	محدوده اعداد	بایت ذخیره
single	اعداد ممیز شناور با دقت یگانه	-10^{38} - 10^{38}	4
double	اعداد ممیز شناور با دقت مضاعف	-10^{308} - 10^{308}	8
uint8	اعداد صحیح بدون علامت ۸ بیتی	0 – 255	1
uint16	اعداد صحیح بدون علامت ۱۶ بیتی	0 – 65535	2
uint32	اعداد صحیح بدون علامت ۳۲ بیتی	0 – 4294967295	4
uint64	اعداد صحیح بدون علامت ٦٤ بیتی	0 - 1.844674407370955e+19	8
int8	اعداد صحیح با علامت ۸ بیتی	-128 - 127	1
int16	اعداد صحیح با علامت ۱٦ بیتی	-32768 - 32767	2
int32	اعداد صحیح با علامت ۳۲ بیتی	-2147483648 - 2147483647	4
int64	اعداد صحیح با علامت ٦٤ بیتی	-4.611686018427388e+18 - 4.611686018427388e+18	8
char	متغیر کاراکتری		2
logical	مقادیر منطقی ۰ یا ۱	1 یا 0	1

۱۰-۳- فرمت های گرافیکی تصاویر

نام فرمت	توضیحات	پسوند فایل
TIFF	Tagged Image File Format	.tif, .tiff
JOEG	Joint Photographic Experts Group	.jpg, .jpeg
GIF	Graphic Interchange Format	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Windows Dump	.xwd
HDF	Hierarcial Data Format	.hdf, .h4, .hdf4, .he2, .h5, .hdf5, .he5
PCX	Paintbrush	.pcx

۱۰-٤- انواع تصویر

- تصاویر شدت Intensity
- تصاویر دودویی Binary
- تصاویر اندیس شده Indexed
- تصاویر RGB

۱۰-٤-۱- تصاویر شدت (Intensity Images)

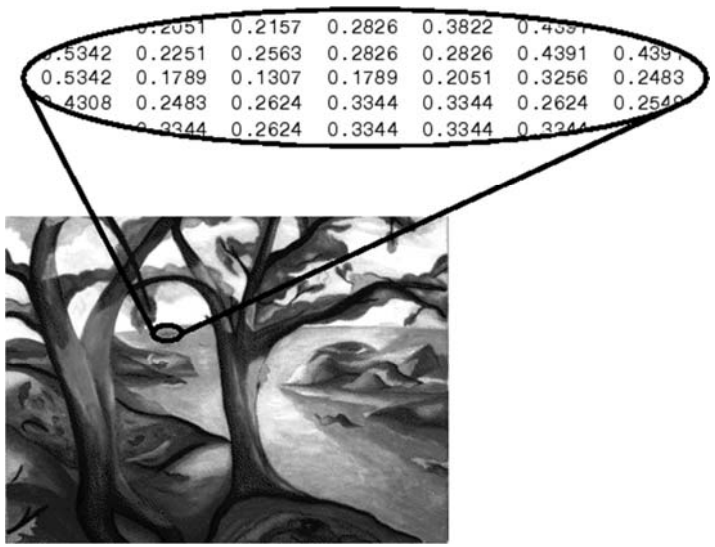
یا تصویر سطح خاکستری (Gray scale)

تنها دارای مقادیر روشنایی هستند و فاقد خصوصیات رنگ مانند: فام (ته رنگ Hue) و خلوص (اشباع Saturation)

در متلب توسط ماتریسهای دو بعدی تعریف می‌شوند بطوریکه مقدار هر عنصر از این ماتریس معرف میزان روشنایی پیکسل متناظرش در تصویر مربوطه می‌باشد.

دامنه تغییرات عناصر این ماتریس ممکن است بین ۰ تا ۱ (نوع دقت مضاعف) و یا بین ۰ تا ۲۵۵ (نوع uint8) تغییر کند. بجز توابع تعریف شده در جعبه ابزار پردازش تصویر و بعضی از توابع خود متلب، سایر عملیات ریاضی بر روی نوع uint8 در حال حاضر امکانپذیر نمی‌باشد. لذا در صورت نیاز، این نوع باید به نوع دقت مضاعف تبدیل شود.

۱۰-٤-۱- تصاویر شدت (Intensity Images)

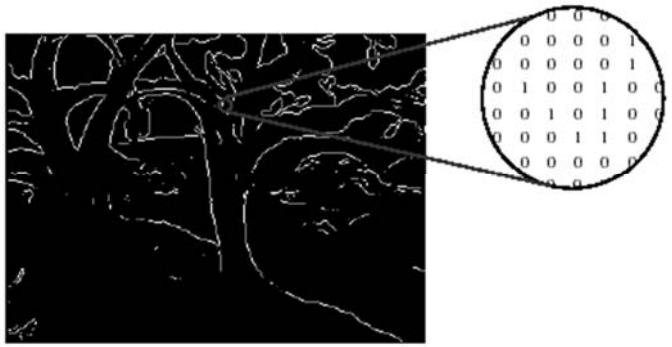


۱۰-٤-۲- تصاویر باینری (Binary Images)

یا تصاویر سیاه و سفید Black & White

هر پیکسل از تصویر تنها می تواند دارای یکی از دو مقدار ممکن (معمولا ۰ و ۱) باشد.

در متلب این تصاویر می‌توانند با فرمت double و یا uint8 ذخیره‌سازی شوند. اما بطور پیش‌فرض متلب فرمت uint8 را بکار خواهد برد که مقادیر آن می‌تواند، ۰ و ۱ و یا ۰ تا ۲۵۵ باشد.



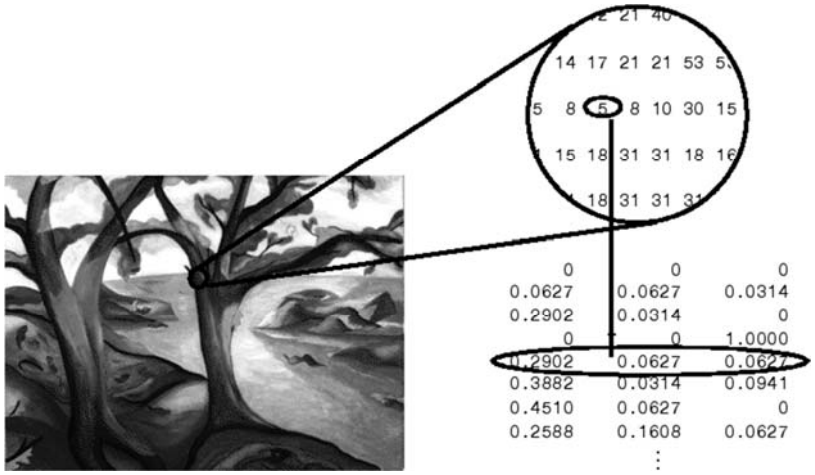
۱۰-٤-۳- تصاویر اندیس‌شده (Indexed Images)

این تصاویر توسط دو ماتریس زیر مشخص می‌شوند:

1. ماتریس اندیس: ماتریسی است که ابعاد آن برابر با ابعاد تصویر بر حسب پیکسل می‌باشد. مقادیر این ماتریس معمولا بین ۱ تا ۲۵٦ تغییر می‌کند و مقدار هر درایه از این ماتریس معرف شماره سطری از ماتریس نقشه رنگ است.
2. ماتریس نقشه‌رنگ(map): این ماتریس دارای ۳ ستون می‌باشد و هر سطر از آن معرف یکی از رنگهای موجود در تصویر است. بطوریکه عنصر اول هر سطر معرف نسبت اولیه قرمز، عنصر دوم معرف اولیه سبز و عنصر سوم معرف اولیه آبی است.

یک تصویر اندیس‌شده بسته به مقادیر ماتریس نقشه رنگ، ممکن است رنگی یا سطح خاکستری باشد.

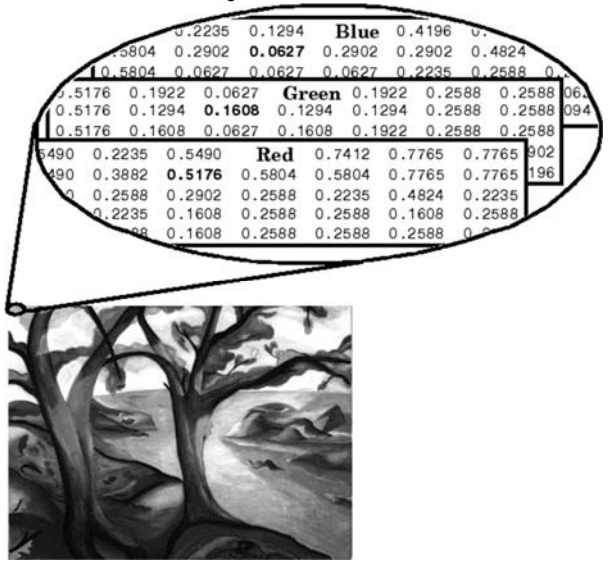
۱۰-٤-۳- تصاویر اندیس‌شده (Indexed Image)



۱۰-٤-٤- تصاویر RGB

- یا true color
- به ازای هر پیکسل از آن سه عدد بین ۰ تا ۲۵۵ در حافظه کامپیوتر ذخیره می شود.
- این اعداد معرف شدت هر یک از اولیه‌های قرمز، سبز و آبی می‌باشد.
- مثلا برای یک پیکسل سفید سه عدد ۲۵۵ و برای یک پیکسل سبز سه عدد ۰، ۲۵۵ و ۰ به ترتیب معرف شدت اولیه‌های قرمز، سبز و آبی ایجاد خواهد شد.
- بنابراین برای هر نقطه از تصویر بیش از ۱٦ میلیون (۲۵٦*۲۵٦*۲۵٦) حالت رنگی مختلف امکان پذیر خواهد بود.
- یک تصویر RGB سه برابر یک تصویر شدت هم‌اندازه با آن حافظه کامپیوتر را اشغال خواهد کرد و به همان نسبت هم به زمان پردازش بیشتری نیاز دارد.
- در متلب هر تصویر RGB بصورت یک ماتریس سه‌بعدی تعریف می‌شود که در بعد سوم آن مقادیر اولیه‌های رنگی هر نقطه (r,g,b) ذخیره می‌شوند. عناصر این ماتریس ممکن است بین ۰ تا ۱ (double) و یا بین ۰ تا ۲۵۵ (uint8) تغییر کند.
- دقت شود که یک تصویر rgb لزوما رنگی نیست اما می‌تواند رنگی باشد.

۱۰-٤-٤- تصاویر RGB



۱۰-۵- خواندن تصاویر- تابع imread

- برای تصاویر شدت، rgb و باینری: m=imread('filename')
- برای تصاویر اندیس‌شده: [m,map]=imread('filename')

در رابطه اخیر m ماتریس اندیس و map ماتریس نقشه‌رنگ خواهد بود.

نکته: تابع imread را با تعداد آرگومانهای بیشتری نیز می‌توان فراخوانی کرد. جهت اطلاع بیشتر به راهنمای متلب رجوع کنید.

۱۰-۶- نمایش تصاویر- تابع imshow

imshow(m); تصویر شدت یا rgb

imshow(I , map) تصویر اندیس‌شده

imshow('filename'); فایل گرافیکی

مثال:

```
>> imshow('fabric.png')
```

یا:

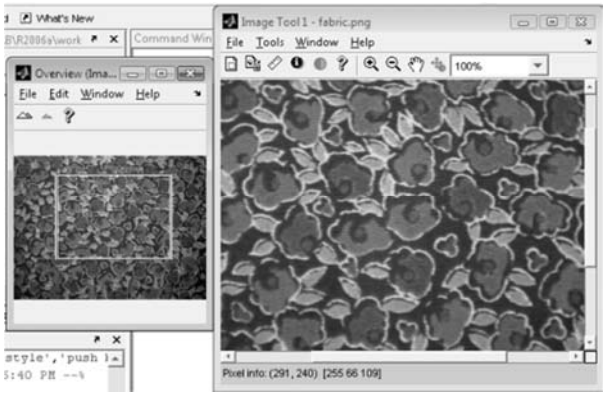
```
>> m=imread('fabric.png');
imshow(m)
```


۱۰-۶- نمایش تصاویر- تابع imshow

۱۰-۷- نمایش تصاویر- تابع imtool

روش استفاده از این تابع مانند تابع imshow است اما قابلیتهای بیشتری را در اختیار می‌گذارد:

```
>> imtool('fabric.png')
```



۱۰-۸- نوشتن فایلهای گرافیکی-imwrite

برای ایجاد یک فایل گرافیکی می‌توان از تابع imwrite استفاده کرد.
این تابع بسته به نوع تصویر می‌تواند به یکی از روشهای زیر بکار برده شود:

```
imwrite(m , 'filename');
imwrite(X , map , 'filename');
```


۱۰-۹- تعیین مشخصات یک فایل گرافیکی- تابع imfinfo

این تابع اطلاعاتی از فایل گرافیکی مانند: ابعاد تصویر، دقت ابعادی و دقت عمقی، نحوه فشرده‌سازی و... را ارائه می‌دهد. این تابع بصورت زیر بکار برده می‌شود:

```
info=imfinfo('filename')
```


۱۰-۱۰- تبدیل تصاویر

با استفاده از توابع زیر می‌توان نوع یک تصویر را تغییر داد:

```
bw=im2bw(m , level)
bw=im2bw(x , map , level)
```

level سطح آستانه می‌باشد.(که باید بین ۰ تا ۱ باشد)

```
m=ind2gray(x , map);
[x,map]=gray2ind(m);
[x,map]=rgb2ind(m);
m=ind2rgb(x , map);
m=rgb2gray(x);
```

برای کسب اطلاعات بیشتر به راهنمای متلب مراجعه کنید.

۱۰-۱۱- عملیات ریاضی بر روی تصاویر

در صورتیکه نوع داده‌های تصویر از نوع uint8 باشد امکان بکاربردن عملگرهای ریاضی و بسیاری از توابع متلب بر روی آنها وجود نخواهد داشت. بدین‌منظور پیش از انجام عملیات ریاضی باید نوع داده‌ها را به double تبدیل کرد. پس از انجام عملیات ریاضی در صورت نیاز می‌توان نوع متغیر را به uint8 بازگرداند:

```
a=double(m);
b=im2uint8(a);
```

۱۰-۱۲- عملیات هندسی بر روی تصاویر

منظور از عملیات هندسی هرگونه تغییر در ابعاد تصویر و یا شکل هندسی آن می‌باشد. سه نوع عملیات هندسی در متلب بر روی تصاویر امکانپذیر است:

- تغییر ابعاد تصویر: تابع imresize
- چرخش تصویر: تابع imrotate
- برش تصویر: تابع imcrop

که در ادامه به هریک خواهیم پرداخت.

۱۰-۱۲-۱- تغییر ابعاد تصویر: تابع imresize

این تابع به یکی از دو صورت زیر قابل استفاده است:

```
y=imresize(x , a);
y=imresize(x , [m , n]);
```

درحالت اول متغیر a نسبت تغییر در ابعاد تصویر است. مثلا اگر برابر با ۲ باشد یعنی ابعاد تصویر دوبرابر خواهد شد. اگر این عدد کمتر از ۱ باشد تصویر کوچکتر خواهد شد و اگر بیشتر از یک باشد تصویر بزرگتر می‌شود.

در حالت دوم تعداد سطر و ستون جدید تصویر به تابع ارایه میشود که باید اعداد صحیح مثبت باشند.

۱۰-۱۲-۲- چرخش تصویر: تابع imrotate

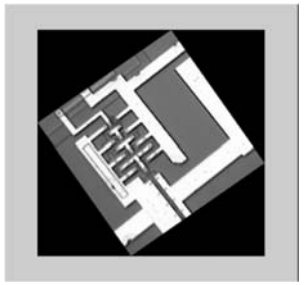
```
m2=imrotate(m , d , ['Option'] , ['crop'])
```

- آرگومان دوم میزان چرخش تصویر برحسب درجه می‌باشد.
- آرگومان سوم اختیاری بوده و می‌تواند یکی از مقادیر bilinear, nearest یا bicubic باشد. در صورتیکه این آرگومان بکار برده نشود، مقدار پیش‌فرض nearest خواهد بود.
- آرگومان چهارم نیز اختیاری می‌باشد و تنها می‌تواند مقدار 'crop' را داشته باشد. درصورتیکه بکار برده شود، ابعاد تصویر پس از چرخش تغییر نمی‌کند اما بخشی از تصویر برش داده و حذف می‌شود.

۱۰-۱۲-۲- چرخش تصویر: تابع imrotate

مثال:

```
m=imread('ic.tif');
n=imrotate(m , 35); p=imrotate(m , 35,'crop');
imshow(n); figure; imshow(p);
```



۱۰-۱۲-۳- برش تصویر: تابع imcrop

این تابع به یکی از شکلهای زیر قابل استفاده است:

```
I2 = imcrop(I,RECT)
X2 = imcrop(X,MAP,RECT)
RGB2 = imcrop(RGB,RECT)
[A,RECT] = imcrop(...)
```

در این روابط rect یک بردار سطری است به شکل [XMIN YMIN WIDTH HEIGHT] که مختصات یک ناحیه مستطیلی شکل که از تصویر برش داده می‌شود را مشخص می‌کند. درصورتیکه این آرگومان در ورودی مشخص نشود، تصویر نمایش داده شده و متلب منتظر می‌ماند تا کاربر یک ناحیه مستطیلی را با ماوس انتخاب کند.

۱۰-۱۲-۳- برش تصویر: تابع imcrop

مثال:

```
m=imread('pout.tif');
imshow(m);figure;imcrop(m,[size(m)/4,size(m)/2]
```

۱۰-۱۳- فیلترهای خطی و طراحی فیلتر

برای اعمال یک فیلتر بر روی تصویر می‌توان از تابع filter2 استفاده کرد:

```
m2=filter2(h , m)
```

در این رابطه، h ماتریس فیلتر و m ماتریس تصویر اولیه است. h می‌تواند هرماتریس با ابعاد دلخواه باشد، اما معمولا یک ماتریس ۳*۳ یا ۵*۵ است.

نمرین: تاثیر فیلترهای زیر را روی یک تصویر بررسی کنید.

```
h=[-1  -1  -1          h= [1/9  1/9  1/9        h=[0.17  0.67  0.17
   -1   8  -1              1/9  1/9  1/9           0.67  -3.33  0.67
   -1  -1  -1]             1/9  1/9  1/9]          0.17  0.67  0.17 ]
```

High Pass Filter (Edge Detection) (Sharpening)

Low Pass Filter (Smoothing)

۱۰-۱۳- فیلترهای خطی و طراحی فیلتر

- فیلترهای آماده

با استفاده از تابع fspecial می‌توان فیلترهای معمول در پردازش تصویر را برای استفاده با تابع filter2 یا تابع imfilter ایجاد کرد. روش استفاده از این تابع بصورت زیر است:

h=fspecial('motion',5) h=fspecial('نام فیلتر' , ابعاد فیلتر)

بسته به نوع آرگومان اول ممکن است این تابع با یک یا بیش دو آرگومان نیز بکار برده شود.

نام فیلتر می‌تواند یکی از پارامترهای زیر باشد:

gaussian:	پایین گذر گوسی	disk:	فیلتر میانگین حلقوی
sobel:	بالا گذر	motion:	فیلتر حرکت
prewitt:	بالا گذر		
laplacian:	فیلتر لاپلاس		
log:	اعمال فیلتر گوسی و پس از آن لاپلاس		
average:	فیلتر میانگین		
unsharp:	پایین گذر		

۱۰-۱۳- فیلترهای خطی و طراحی فیلتر

•فیلترهای آماده-مثال

```
S=fspecial('sobel');
rose=imread('rose.jpg');roseg=rgb2gray(rose);
  rosegf=imfilter(roseg,S);
imshow(roseg); figure; imshow( rosegf );
```

۱۰-۱۴- آنالیز و بهسازی تصویر

آنالیز و بهسازی تصویر شامل سه عملیات زیر است:

- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها
- آنالیز تصویر بمنظور استخراج اطلاعات در مورد ساختار کلی آن
- بهسازی تصویر بمنظور واضح‌تر شدن جزییات تصویر و حذف نویز به منظور آماده‌سازی برای عملیات پردازشی بعدی

۱۰-۱۴-۱- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

توابع impixel و impixelinfo

با استفاده از تابع impixel می‌توان مشخصات رنگی پیکسل هایی از تصویر را بدست آورد. این تابع بصورتهای زیر بکار می‌رود:

P = impixel(I)

P = impixel (X,MAP)

P = impixel (RGB)

در این حالت این تابع پنجره تصویر را نمایان ساخته امکان انتخاب نقاط مورد نظر را به کاربر می‌دهد. با زدن دکمه سمت راست ماوس، آخرین نقطه انتخاب و مشخصات این نقاط در ماتریس p ذخیره خواهد شد. البته این تابع بصورتهای دیگری نیز می‌توان بکار برد که برای کسب اطلاعات بیشتر می‌توانید به راهنمای متلب مراجعه کنید.

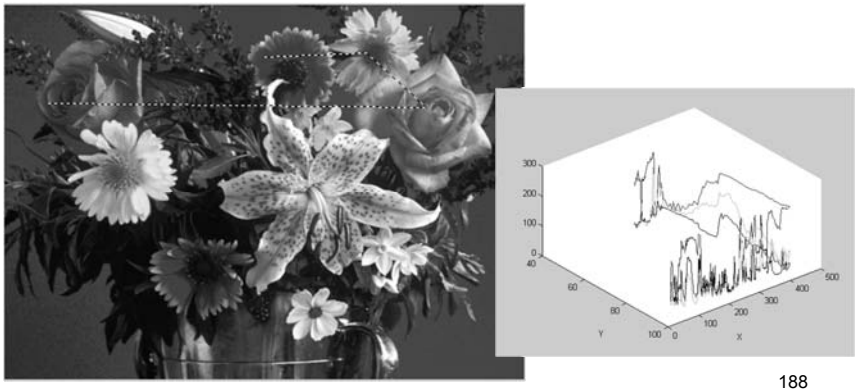
تابع impixelinfo به پایین پنجره تصویر کادری را اضافه می‌کند که با حرکت ماوس بر روی تصویر مشخصات رنگی نقاط نصویر در این کادر نمایش داده می‌شود. این تابع باید پس از نمایش تصویر با تابع imshow صدا زده شود.

۱۰-۱۴-۱- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

تابع improfile: این تابع نمودار تغییرات رنگ تصویر را در یک مسیر دلخواه که با ماوس آنتخاب می‌شود رسم می‌کند:

مثال:

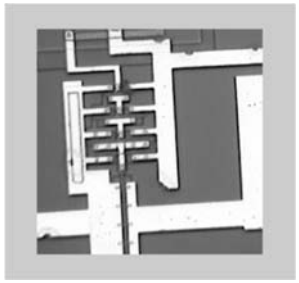
```
imshow('flowers.tif');improfile;
```

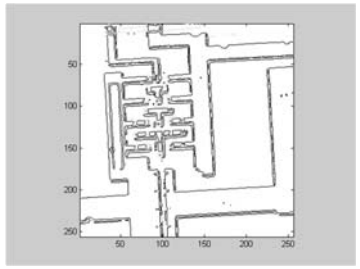


۱۰-۱٤-۱- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

تابع imcontour: رسم نمودار تراز داده‌های تصویر:

```
im=imread('ic.tif');
imshow(im);figure;imcontour(im,3);
```

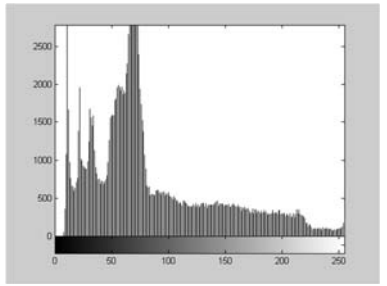




۱۰-۱٤-۱- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

تابع imhist: رسم نمودار فراوانی نقاط تصویر:

```
I=imread('flowers.tif');I2=rgb2gray(I);
imshow(I2);figure;imhist(I2);
```



۱۰-۱٤-۱- بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

توابع mean2 و std2:

توابع mean و std در متلب به ترتیب برای بدست آوردن میانگین و انحراف معیار بکار برده می‌شوند. اما این توابع بصورت برداری عمل می‌کنند یعنی میانگین یا انحراف معیار عناصر یک بردار را محاسبه می‌کنند. اگر این توابع را بر روی یک ماتریس اعمال کنیم مانند اکثر توابع متلب بصورت ستونی روی عناصر آن ماتریس عمل خواهند کرد. یعنی میانگین یا انحراف معیار هر ستون ماتریس را بصورت جداگانه بدست می‌آورند. برای آنکه بتوان میانگین یا انحراف معیار تمامی نقاط یک ماتریس را بدست آورد باید از توابع mean2 و std2 استفاده کرد.

۱۰-۱٤-۲- آنالیز تصویر

از آنجاییکه آنالیز تصویر بیشتر بر روی تصاویر باینری انجام می‌گردد این مبحث به سرفصل "عملیات بر روی تصاویر باینری " ارجاع می‌شود.

۱۰-۱٤-۳- بهسازی تصویر

این عملیات که به عملیات پیش‌پردازش نیز مشهور است معمولا پیش از عملیات پردازش اصلی یا عملیات آنالیز تصویر انجام می‌گیرد. در این عملیات بهبودهایی بر روی داده‌های تصویر اعمال می‌شود تا امکان استخراج دقیقتر و صحیح‌تر اطلاعات میسر گردد. این عملیات در سه بخش زیر شرح داده خواهد شد:

- تنظیم شدت
- متعادل کردن هیستوگرام یا بهسازی تباین
- حذف نویز

۱۰-۱٤-۳- بهسازی تصویر

تنظیم شدت-تابع imadjust

با استفاده از این تابع می‌توان دامنه تغییرات روشنایی یک تصویر را تغییر داد. شکل کلی کاربرد این تابع بصورت زیر است:

J=imadjust(I , [low-in , high-in] , [low-out , high-out])

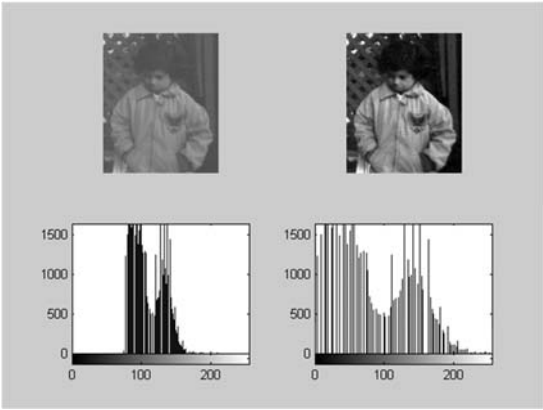
آرگومان دوم برداری دو عنصری است که بیانگر دامنه حاوی روشنایی‌هایی از تصویر است که عملیات تنظیم شدت بر روی آنها باید اعمال گردد. آرگومان سوم، دامنه تغییرات جدید روشنایی برای نقاط فوق است.

مثال:

```
I=imread('pout.tif');
J=imadjust(I , [0.3 , 0.7] , [0 ,1]);
subplot(2,2,1);imshow(I); subplot(2,2,2);imshow(J);
subplot(2,2,3); imhist(I); subplot(2,2,4); imhist(J)
```


۱۰-۱٤-۳- بهسازی تصویر

تنظیم شدت-تابع imadjust



۱۰-۱٤-۳- بهسازی تصویر

متعادل کردن هیستوگرام یا بهسازی تباین-تابع histeq

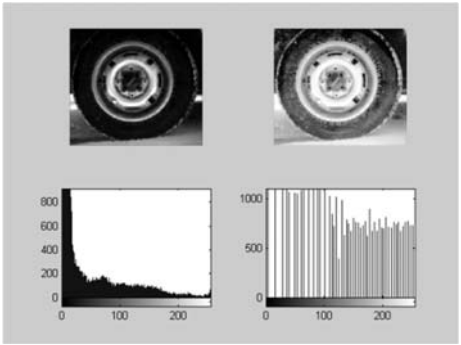
تابع histeq بصورت اتوماتیک بهترین تنظیم هیستوگرام را بر روی تصویر انجام می‌دهد و معمولا کیفیت روشنایی تصویر را به میزان زیادی بهبود می‌بخشد.

مثال:

```
I=imread('tire.tif');
J=histeq(I);figure;
subplot(2,2,1);imshow(I);
subplot(2,2,2);imshow(J);
subplot(2,2,3);imhist(I);
subplot(2,2,4);imhist(J);
```

۱۰-۱٤-۳- بهسازی تصویر

متعادل کردن هیستوگرام یا بهسازی تباین-تابع histeq



۱۰-۱٤-۳- بهسازی تصویر

حذف نویز

معمولا تصاویر دیجیتال کم و بیش دارای نویز هستند. حذف نویز قبل از هرگونه عملیات پردازشی باید انجام گیرد. فیلترهای متعددی برای حذف نویز طراحی شده‌اند. در متلب نیز چندین فیلتر برای حذف نویز وجود دارد که از این میان به ساده‌ترین آنها اشاره خواهیم کرد:

- فیلتر میانگین
- فیلتر میانه

برای ایجاد فیلتر میانگین از تابع fspecial که قبلا توضیح داده شد و تابع filter2 می‌توان استفاده کرد. برای اعمال فیلتر میانه از تابع medfilt2 استفاده کنید. بطورکلی تمامی فیلترهای حذف نویز از وضوح (sharpness) تصویر می‌کاهند. در میان دو فیلتر میانگین و میانه، فیلتر میانه معمولا نتیجه بهتری ایجاد می‌کند و وضوح تصویر را نیز کمتر تحت تاثیر قرار می‌دهد.

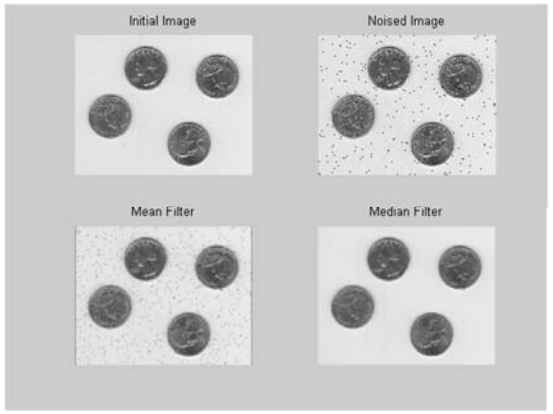
۱۰-۱٤-۳- بهسازی تصویر

حذف نویز-مثال: مقایسه فیلتر میانه و فیلتر میانگین

```
I = imread('eight.tif');
J= imnoise(I , 'Salt & pepper' , 0.02); % افزودن نویز
K= filter2(fspecial('average' , 3) , J) / 255; % فیلتر میانگین
L=medfilt2(J , [3 , 3]); % فیلتر میانه
subplot(2,2,1); imshow( I ); title('Initial Image')
subplot(2,2,2); imshow( J ); title('Noised Image');
subplot(2,2,3); imshow( K ); title('Mean Filter');
subplot(2,2,4); imshow( L ); title('Median Filter');
```


۱۰-۱٤-۳- بهسازی تصویر

حذف نویز-مثال: مقایسه فیلتر میانه و فیلتر میانگین



۱۰-۱۵- عملیات بر روی تصاویر باینری

اگرچه عملیات بر روی تصاویر باینری زیرمجموعه مبحث آنالیز تصویر است لکن بخاطر اهمیت تصاویر باینری در علم پردازش تصویر، این مبحث را در بخش جدیدی ارایه نموده‌ایم.

همانگونه که قبلا گفته شد تصویر باینری به تصویری گفته می‌شود که پیکسلهای آن تنها دارای یکی از دو مقدار ممکن ۰ و ۱ یا ۰ تا ۲۵۵ باشند. در متلب تصاویر باینری می‌توانند بصورت تصاویر شدت و یا بصورت تصاویر اندیس‌شده ذخیره و معرفی شوند. در حالت دوم ماتریس نقشه رنگ تنها دارای دو سطر خواهد بود.

۱۰-۱۵- عملیات بر روی تصاویر باینری

نمایش تصاویر باینری

برای نمایش تصاویر باینری نیز از تابع imshow استفاده می‌شود. درصورتیکه تصویر از نوع شدت باشد فرم : imshow(m) و اگر از نوع اندیس‌شده باشد فرم : imshow(I ,map) بکار برده خواهد شد.

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations

عملیات ساختاری به عملیاتی گفته می‌شود که بر روی تصاویر باینری اعمال شده و هدف از آن ایجاد تغییر و یا تصحیح در اجزا داخل یک تصویر باینری باشد. این عملیات معمولا یک مرحله قبل از عملیات پردازش نهایی انجام میشود. منظور از عملیات پردازش نهایی عملیاتی است که در آن اطلاعاتی از تصویر استخراج میشود. مثلا محیط یا مساحت اجزا تصویر محاسبه می‌گردد.

از میان این عملیات در ادامه چهار نوع از مهترین آنها شرح داده خواهد شد که عبارتند از:

- عملیات افزایش
- عملیات فرسایش
- عملیات گشودن
- عملیات بستن

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات افزایش و فرسایش(Dilation & Erosion)

منظور از عملیات افزایش عملیاتی است که باعث افزایش ابعاد اجزا داخل تصویر به اندازه یک یا چند پیکسل می‌گردد. در اثر این عمل ممکن است نقاطی که از یک تصویر باینری در اثر عواملی چون تاثیر نویز یا اعمال حد آستانه نامطلوب جا افتاده است، تصحیح گردند. مثلا ممکن است دو جزء از تصویر به یکدیگر متصل گردند. الگوریتم اعمال فیلتر افزایش بدین صورت است که تمامی نقاط سیاه تصویر بررسی شده در صورتیکه حداقل یکی از همسایگان انتخابی نقطه مورد بررسی سفید باشند، نقطه مزبور نیز سفید خواهد شد در غیر اینصورت سیاه باقی خواهد ماند.

عملیات فرسایش دقیقا عکس عملیات افزایش است. در این عملیات معمولا نقاط ناخواسته تصویر باینری حذف می‌شوند و سایر اجزا تصویر نیز به اندازه یک یا چند پیکسل نازکتر خواهند شد. عملا تمامی نقاط سفید تصویر بررسی شده در صورتیکه حداقل یکی از همسایگان انتخابی آن سیاه باشد، آن نقطه نیز سیاه خواهد شد.

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات افزایش و فرسایش-ادامه

ابعاد همسایگی و انتخاب همسایه‌ها توسط یک ماتریس ماسک (Mask) مشخص می‌شوند. مثلا اگر ماتریس ماسک یک ماتریس ۳* ۳ باشد که تمای عناصر آن برابر با ۱ باشد. یعنی یک همسایگی ۳*۳ بکار برده شود و تمامی ۹ همسایه نقطه مورد بررسی برای عملیات افزایش یا فرسایش مد نظر قرار گیرند.

برای عملیات افزایش در متلب از تابع imdilate و برای عملیات فرسایش از تابع imerode استفاده کنید. اگرچه هردو عملیات را با استفاده از تابع کلی‌تر bwmorph نیز می‌توان انجام داد.

فرمول کلی استفاده از این توابع بصورت زیر است:

```
bw2=imerode(bw1, se);
bw2=imdilate(bw1 , se);
```


۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات افزایش و فرسایش-مثال

```
bw1=imread('circbw.tif'); SE=eye(5);
bw2=imerode(bw1 , SE);
imshow(bw1); figure; imshow(bw2);
```

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات گشودن و بستن Open & Close

از ترکیبهای مختلف دو عملیات افزایش و فرسایش می‌توان عملیات دیگری ایجاد کرد. مهمترین این عملیات، عملیات گشودن و بستن است. در عملیات گشودن اجزایی از تصویر باینری که از یک اندازه تعیین شده کوچکتر باشند حذف می‌شوند بدون آنکه ابعاد سایر اجزا تغییر کند. در عملیات بستن نیز نواحی جاافتاده تصویر باینری بدون تغییر در ابعاد سایر اجزا ترمیم می‌گردند.

عملا در صورتیکه ابتدا عملیات فرسایش و سپس افزایش بر یک تصویر باینری اعمال شود، نتیجه، عملیات گشودن خواهد بود اما اگر ابتدا افزایش و سپس فرسایش اعمال گردد، عملیات بستن حاصل خواهد شد.

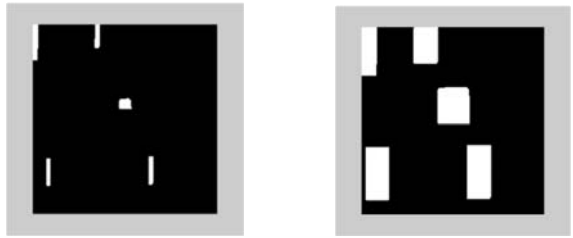
در متلب برای اعمال عملیات گشودن و بستن و همچنین سایر عملیات مورفولوژی از تابع bwmorph باید استفاده کرد. اگرچه می‌توان این دو عملیات را از عملیات فرسایش و افزایش نیز بدست آورد.(همانگونه که در مثال بعدی عمل شده است)

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات گشودن و بستن Open & Close-مثال

```
bw1=imread('circbw.tif');
se= ones(40 , 30); bw2= imerode(bw1 , se);
bw3=imdilate(bw2 , se);
imshow(bw2); figure; imshow(bw3);
```



۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات از پیش تعریف شده: تابع immorph

با استفاده از تابع immorph می‌توان بسیاری از عملیات ساختاری معروف پردازش تصویر را اعمال نمود. شکل کلی استفاده از این تابع بصورت زیر است:

```
bw2 = bwmorph(bw1 , operation , [n]);
```

آرگومان سوم اختیاری بوده و بیانگر ابعاد ماسک مورد استفاده یا فاکتور دیگری با توجه نوع آرگومان دوم در عملیات است. در صورت حذف آرگومان سوم، مقدار پیش فرض آن بکار برده خواهد شد. مقدار آرگومان دوم یکی از رشته‌های زیر است:

erode fill hbreak open skel remove close dilate

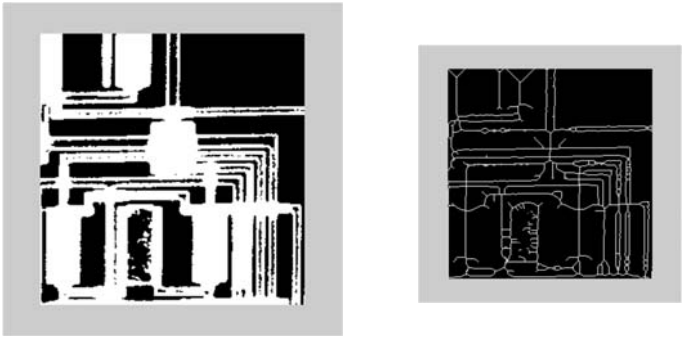
مثال بعدی نتیجه عملیات اسکلتون را بر روی تصویر قبلی نشان می‌دهد

۱۰-۱۵- عملیات بر روی تصاویر باینری

عملیات ساختاری Morphological Operations-ادامه

عملیات از پیش تعریف شده: تابع immorph- مثال:

```
bw1= imread('circbw.tif'); bw2= bwmorph(bw1 , 'skel' , inf)
imshow(bw1); figure; imshow(bw2);
```



تکلیف

تکلیف ۱۳-۱- تصویری به نام flower.tif از نوع rgb در دست است. این تصویر شامل یک گل به رنگ قرمز و ساقه و برگ به رنگ سبز بر روی یک زمینه آبی است. برنامه‌ای بنویسید که:

الف – تصویر فوق را خوانده و داده‌های آنرا در ماتریسی به نام m بریزد.

ب- با استفاده از حد آستانه ۱۲۰ برای جزء سبز و حد آستانه ۱۸۰ برای جز قرمز، دو تصویر باینری بنامهای b1 و b2 ایجاد کند که در اولی تنها تصویر گل و در دومی تنها اجزاء ساقه و برگ وجود داشته باشند.

راهنمایی: برای استخراج برگها تنها استفاده از یک شرط برای حد آستانه کافی نیست. مثلا شرط: m(:,:,1) < 100 & m(:,:,2)>120 را امتحان کنید.

ج- مرز گل را در تصویر b1 استخراج کرده و در b11 بریزد.

د- تصاویر b11 و b2 را با استفاده از عملگر یای منطقی در متلب، با یکدیگر تلفیق نماید تا تصویر باینری c بدست آید.

ه- مساحت برگ و ساقه و مساحت و محیط گل را از تصاویر b1، b11 و b2 بدست آورد.

و- مختصات نخستین پیکسل سفید (نسبت به گوشه بالا- سمت چپ تصویر) در تصاویر b1 و b2 را بدست آورد.

ز- با استفاده از دستور text و نتایج قسمتهای "ه" و " و" پس از نمایش تصویر مساحت و محیط هر جز را در کنار آن نمایش دهد.

تکلیف

تکلیف ۱۳-۲- تصویر یک پارچه سفید با نام fabric.tif و از نوع شدت(grayscale) در دست است. این تصویر دارای یک طرح بافت خاص می‌باشد برنامه‌ای بنویسید که با استفاده از تبدیل فوریه یک بعدی فرکانس تکرار طرح مزبور در جهت افقی و عمودی و با استفاده از این فرکانسها و طول و عرض تصویر، ابعاد طرح فوق را محاسبه کند و نمایش دهد. رزولوشن تصویر را ۶۰۰ dpiدر نظر بگیرید.

راهنمایی: بدین منظور یک سطر و یک ستون از تصویر را انتخاب و طیف فوریه آنرا بدست آورید...