

فصل اول ویژگیهای اصلی MATLAB

MATLAB®

کلاس آموزشی

فصل اول: ویژگیهای اصلی MATLAB

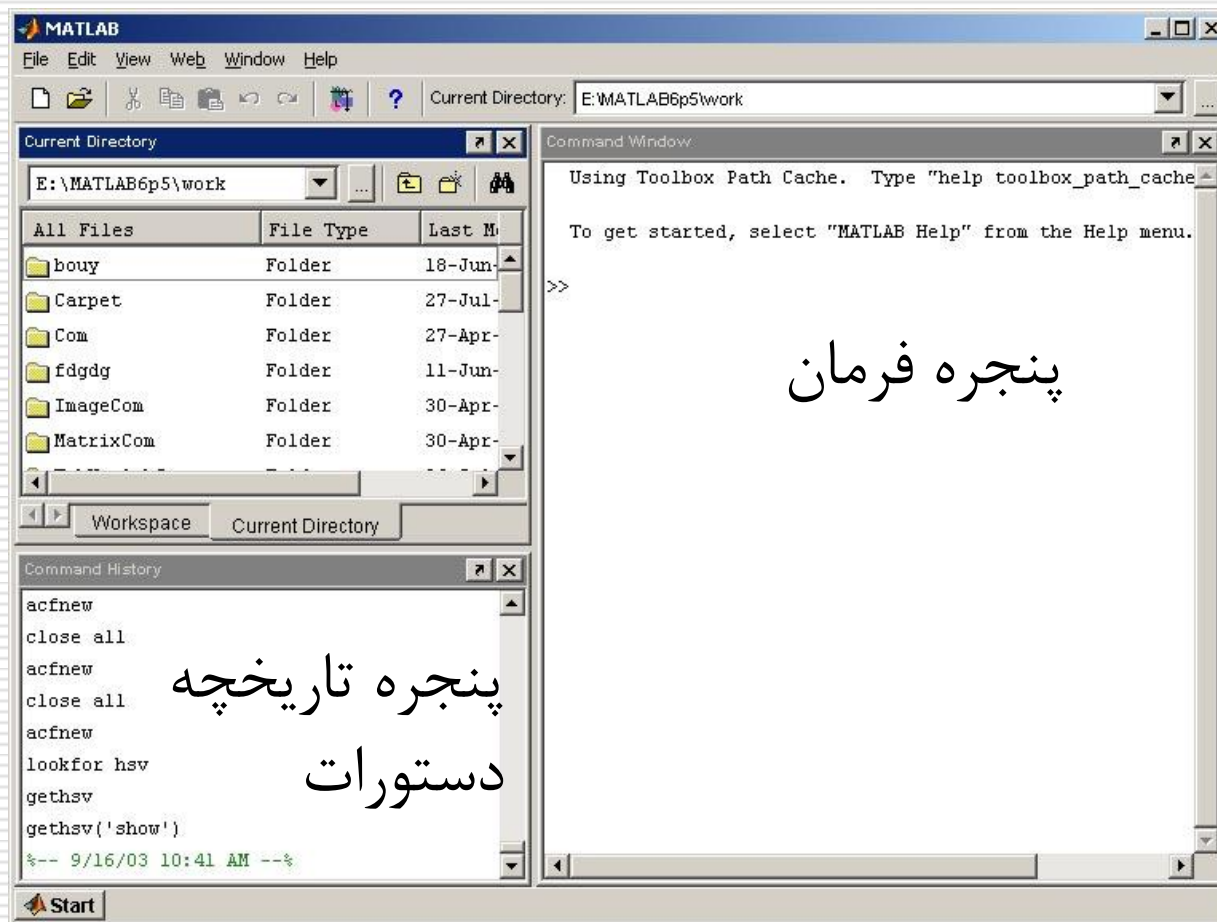
- ☐ آشنایی با محیط متلب
- ☐ عملیات ریاضی ساده
- ☐ عملگرهای ریاضی متلب
- ☐ فضای کاری متلب (Workspace)
- ☐ فرمت نمایش اعداد
- ☐ انواع متغیرها
- ☐ نامگذاری متغیرها
- ☐ متغیرهای ویژه
- ☐ علائم نقطه گذاری و جملات توضیحی
- ☐ اعداد مختلط
- ☐ بعضی از توابع ریاضی در متلب
- ☐ راهنمای متلب
- ☐ فایل‌های متنی یا m-فایلها
- ☐ مدیریت فایل در متلب

فصل اول: ویژگیهای اصلی MATLAB

۱-۱- آشنایی با محیط متلب

- پنجره فرمان : Command window
- پنجره تاریخچه دستورات: Command History
- پنجره دایرکتوری جاری : Current Directory
- پنجره فضای کاری : Work Space
- دایرکتوری جاری
- منوی Start

فصل اول: ویژگیهای اصلی MATLAB



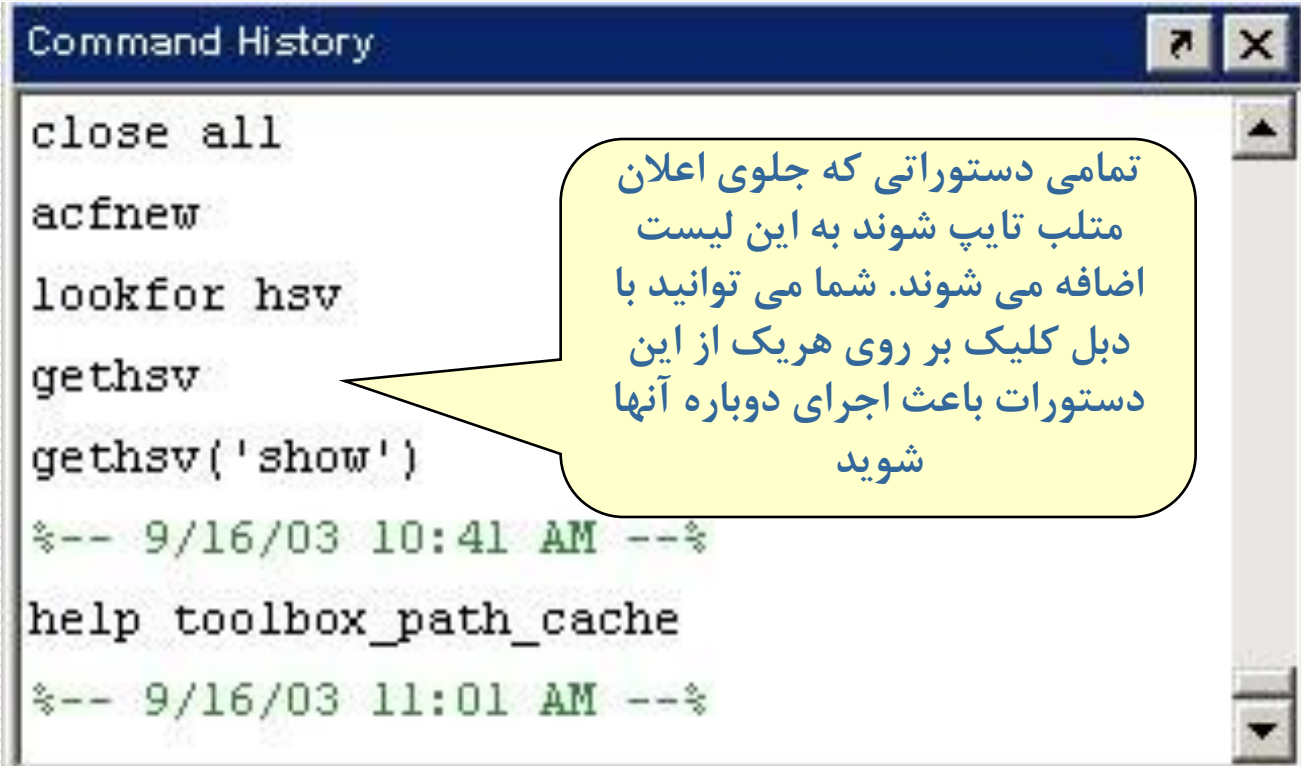
فصل اول: ویژگیهای اصلی MATLAB

پنجره فرمان : Command Window



فصل اول: ویژگیهای اصلی MATLAB

پنجره تاریخچه دستورات: Command History



The image shows the MATLAB Command History window. The title bar is 'Command History'. The list of commands includes: 'close all', 'acfnew', 'lookfor hsv', 'gethsv', 'gethsv('show')', a separator line with date and time, 'help toolbox_path_cache', and another separator line with date and time. A yellow callout box points to the 'gethsv' command with Persian text explaining that all commands are listed and can be double-clicked to re-execute them.

```
Command History
```

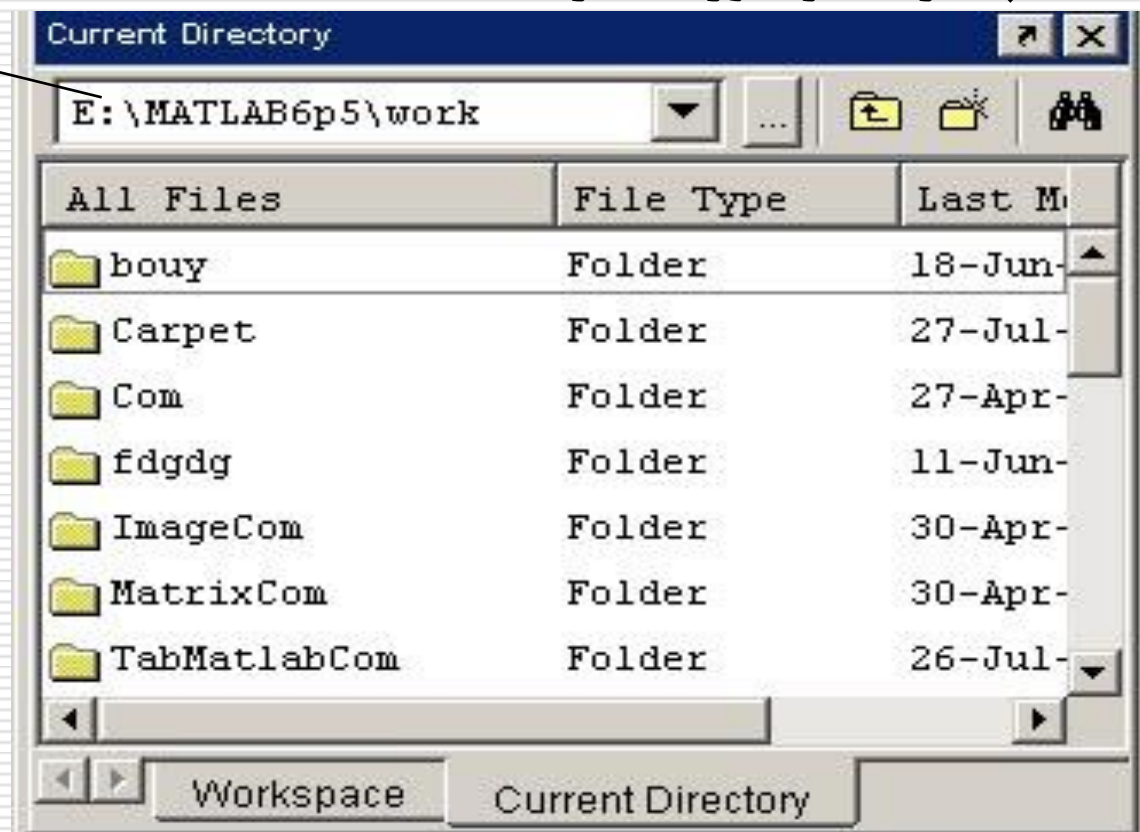
```
close all  
acfnew  
lookfor hsv  
gethsv  
gethsv('show')  
%-- 9/16/03 10:41 AM --%  
help toolbox_path_cache  
%-- 9/16/03 11:01 AM --%
```

تمامی دستوراتی که جلوی اعلان
متلب تایپ شوند به این لیست
اضافه می شوند. شما می توانید با
دبل کلیک بر روی هریک از این
دستورات باعث اجرای دوباره آنها
شوید

فصل اول: ویژگیهای اصلی MATLAB

پنجره دایرکتوری جاری : Current Directory

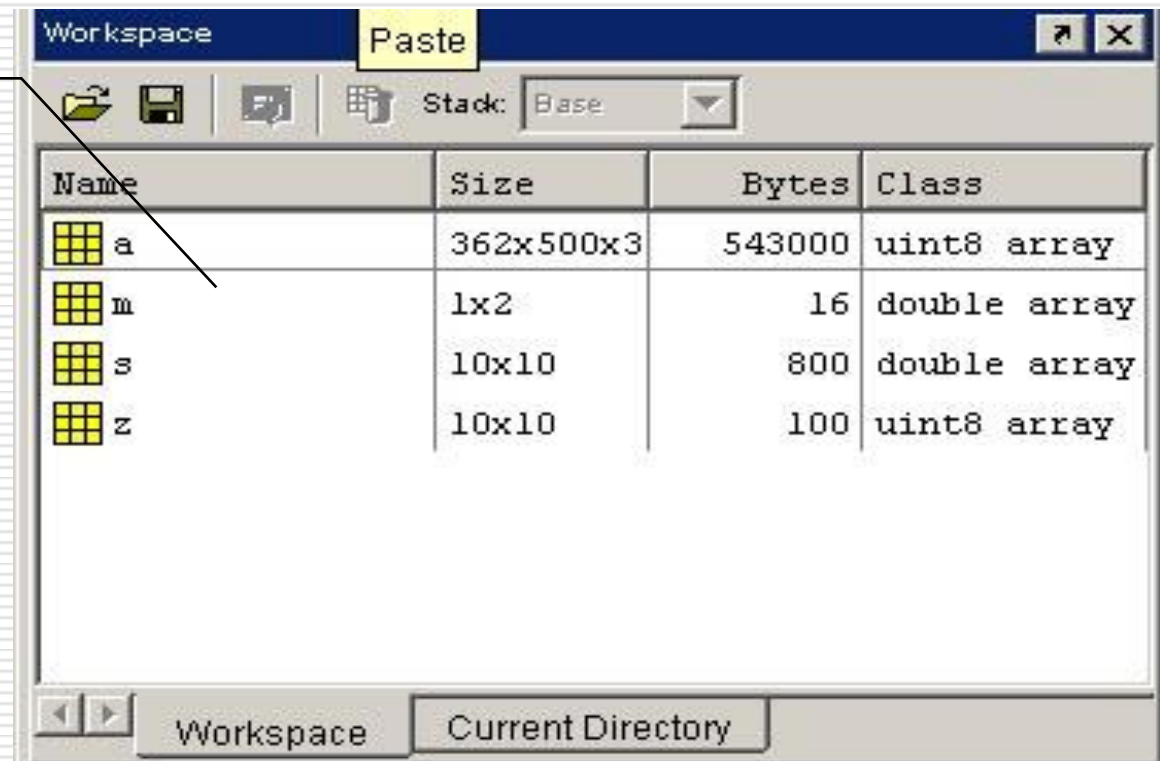
در هر زمان تنها یک دایرکتوری یا پوشه به عنوان دایرکتوری جاری در متلب شناخته می شود. هر فایل متلب (برنامه نوشته شده توسط شما) که نام آن جلوی اعلان متلب تایپ شود تنها در صورتی اجرا می شود که در دایرکتوری جاری یا در مسیر متلب باشد



فصل اول: ویژگیهای اصلی MATLAB

فضای کاری : Work Space

متغیرهایی که در حال حاضر در محیط کاری متلب وجود دارند و شما می توانید از مقادیر آنها استفاده کنید یا آنها را تغییر دهید



فصل اول: ویژگیهای اصلی MATLAB

۱-۲- عملیات ریاضی ساده

مثال: محاسبه یک عبارت:

راه اول:

```
>> 4*25 + 6*22 + 2*99
```



```
ans=
```

```
430
```

فصل اول: ویژگیهای اصلی MATLAB

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 ,

نکته ۲: تعریف متغیرها

نکته ۳: متغیرهای ویژه

فصل اول: ویژگیهای اصلی MATLAB

۱-۳- عملگرهای ریاضی متلب:

\backslash , / , * , - , + , ^

مثال:

```
>> 5^2
```

```
ans=
```

```
25
```

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

□ ترتیب حق تقدم: $- + > * \backslash / > ^$

فصل اول: ویژگیهای اصلی MATLAB

۱-۴- فضای کاری متلب Work space

متغیرهایی که در محیط متلب ایجاد می شوند در بخشی از حافظه بنام محیط کاری متلب ذخیره می گردند. فضای کاری برنامه های اسکریپت متلب با فضای کاری متلب یکسان است. یعنی اگر تغییری در محیط متلب تعریف شده باشد در یک برنامه اسکریپت می توان از آن استفاده کرد و برعکس. اما برنامه های تابعی متلب دارای فضای کاری مختص به خود هستند و متغیرهای آنها در فضای کاری متلب وارد نمی شود.

■ در مورد انواع برنامه های متلب در فصلهای آتی توضیح داده خواهد شد.

فصل اول: ویژگیهای اصلی MATLAB

۱-۴ فضای کاری متلب Work space

نکاتی در مورد فضای کاری متلب:

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

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۱- 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

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۲- 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.

فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۳- 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.

مثال:

```
>>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'
```


فصل اول: ویژگیهای اصلی MATLAB

۱-۴-۳- ذخیره و بازیابی متغیرها: دستورات `save` و `load`:
فرم کلی کاربرد دستورات `save` و `load` بصورت زیر است:

`save [filename] [variables]`

`Load [filename] [variables]`

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

فصل اول: ویژگیهای اصلی MATLAB

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, hex,
+ , ...

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

فصل اول: ویژگیهای اصلی MATLAB

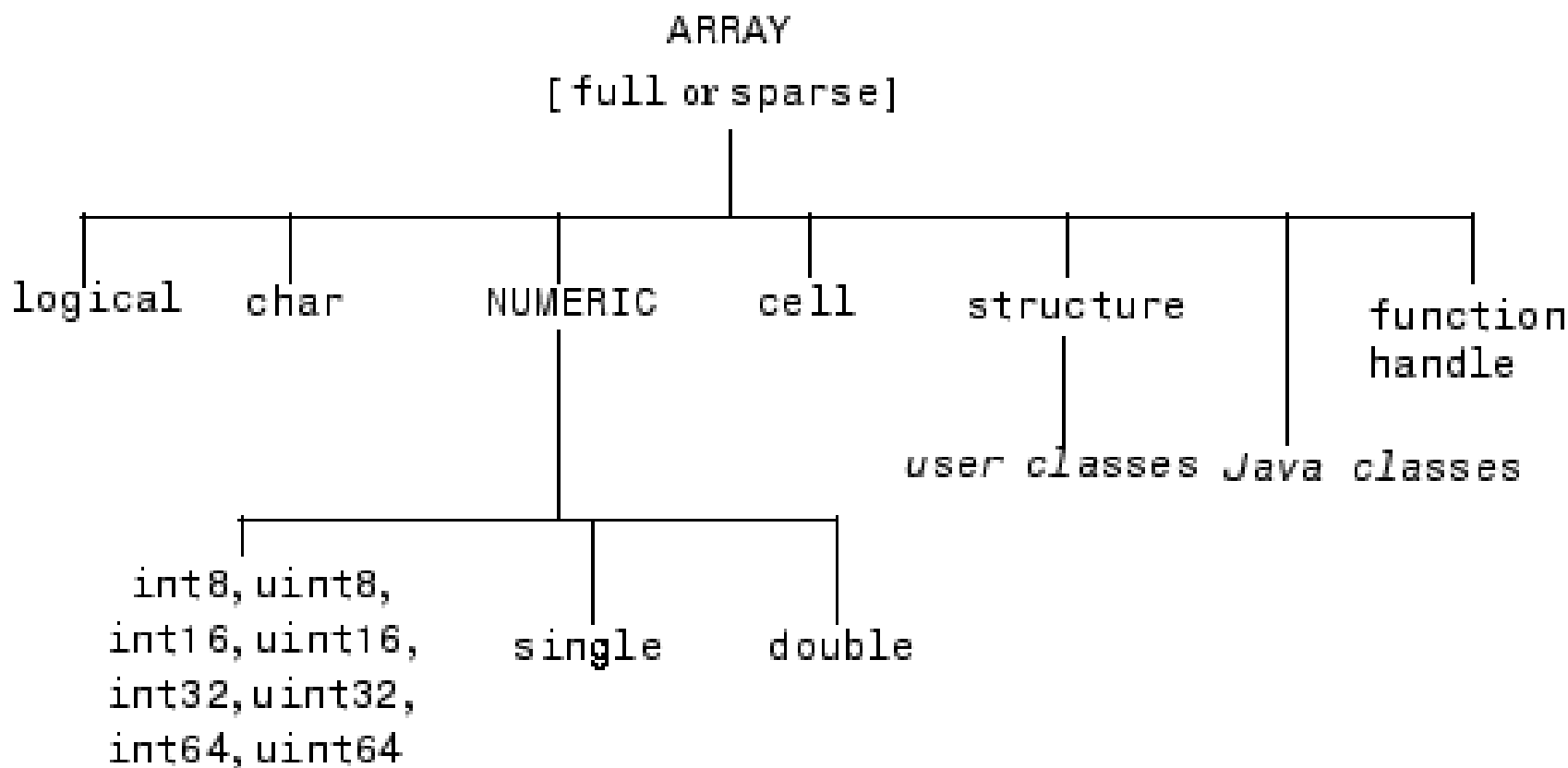
۱-۶- انواع متغیرها

بعضی از مهمترین انواع متغیر در متلب:

double	: نقطه اعشار با دقت مضاعف (۸ بایت)
struct	: نوع تعریف شده توسط کاربر
single	: نقطه اعشار (۴ بایت)
uint8	: عدد صحیح بی علامت ۸ بیتی
uint16	: عدد صحیح بی علامت ۱۶ بیتی
uint32	: عدد صحیح بی علامت ۳۲ بیتی
uint64	: عدد صحیح بی علامت ۶۴ بیتی
int8	: عدد صحیح ۸ بیتی
int16	: عدد صحیح ۱۶ بیتی
int32	: عدد صحیح ۳۲ بیتی
int64	: عدد صحیح ۶۴ بیتی

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

فصل اول: ویژگیهای اصلی MATLAB



فصل اول: ویژگیهای اصلی MATLAB

۱-۶- 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);
```

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

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

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 می توانید 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.

فصل اول: ویژگیهای اصلی MATLAB

۱-۷- نامگذاری متغیرها

- 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 ۲۰۰۶، حداکثر ۶۳ کاراکتر است.
- مراقب باشید متغیر شما با یک تابع درونی MATLAB یا تابعی که توسط خود شما نوشته شده است همنام نباشد. برای اطمینان از دستور `which -all varName` استفاده کنید

مثال:

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

فصل اول: ویژگیهای اصلی MATLAB

۱-۸- متغیرهای ویژه

متغیرهای زیر در محیط متلب بصورت پیش فرض وجود دارند.

ans	NaN	nargin
pi	i	nargout
eps	j	
inf		

فصل اول: ویژگیهای اصلی MATLAB

۱-۹-علائم نقطه گذاری 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;
```


فصل اول: ویژگیهای اصلی MATLAB

۱-۱۰- اعداد مختلط

□ برای تعریف اعداد مختلط از متغیرهای ویژه i و j می‌توان استفاده کرد:

```
>>c=1-2i;
```

```
>>k=(-1)^(1/2);
```

```
>>c=1-2j;
```

```
>>c=1-2*k;
```

```
>>c=1-2*j;
```

□ توابع کار با اعداد مختلط:

abs

angle

real

imag

فصل اول: ویژگیهای اصلی MATLAB

۱-۱-۱- بعضی از توابع ریاضی در متلب

abs	conj	log10
acos	exp	real
asin	fix	imag
acosh	round	rem(x,y)
asinh	gcd(x,y)	sign
atan	lcm(x,y)	sqrt
atanh	log	

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۲- راهنمای متلب

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

demo

help

lookfor

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳- 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.

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۳-۱ - مراحل ایجاد فایل‌های متنی

۱. باز کردن یک فایل جدید در ویرایشگر متلب:

File>New>m-file

۲. تایپ کردن دستورات متلب در فایل مذکور

۳. ذخیره کردن فایل با نامی مشخص:

File>Save As...

فصل اول: ویژگیهای اصلی MATLAB

۱-۲-۳- روش اجرای یک فایل متنی

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

نکته: از این پس متن برنامه ها (کد نوشته شده در فایل های **m**) با رنگ سبز نشان داده خواهد شد.

مثال: برنامه sample1.m

```
% SAMPLE1: A Simple m-file
n=10;a=2;b=4;
c=n*a^3/b + 3*n*a^2/b^2+6*n*a/b^3
```

```
>> sample1
```

```
C=
```

```
29.3750
```

فصل اول: ویژگیهای اصلی MATLAB

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

۱. 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
```

فصل اول: ویژگیهای اصلی MATLAB

۱-۳-۴- توابع و دستورات مفید در فایل‌های m

۲. 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
```


فصل اول: ویژگیهای اصلی MATLAB

۱-۳-۴- توابع و دستورات مفید در فایل‌های m
 ۳. 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 + 3*n*a^2/b^2 + 6*n*a/b^3;
disp('Please wait 5 seconds only!');pause(5);
disp('Press any key to see answer. '); pause;
disp(' "C" Value is= '); disp(c)
```

فصل اول: ویژگیهای اصلی MATLAB

تکلیف ۱-۱: برنامه‌ای بنویسید که یک عدد را از کاربر بگیرد و آنرا در متغیری به نام x ذخیره کند. با استفاده از آن، عبارت زیر را محاسبه کند و مقدار y را با پیغام مناسب نمایش دهد.

$$y = x^3 + 3 * x^2 + 6 * x + 6;$$

□ با تایپ نام برنامه در جلوی اعلان MATLAB، آنرا اجرا کنید.

□ با استفاده از ویرایشگر MATLAB، برنامه خود را اجرا و trace کنید.

فصل اول: ویژگیهای اصلی MATLAB

۱-۱۴- 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

MATLAB®

کلاس آموزشی

فصل دوم: آرایه‌ها

۲-۱- ایجاد آرایه

روشهای ایجاد آرایه:

۱. با استفاده از علائم `;` ، `,` و `[]`
۲. با استفاده از علامت `:`
۳. با استفاده از توابع `linspace` و `logspace`
۴. با استفاده از ترکیبی از روشهای فوق

فصل دوم: آرایه‌ها

۲-۱-۱- ایجاد آرایه با استفاده از علائم ; ، ، و []
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
```

فصل دوم: آرایه‌ها

۲-۱-۱- ایجاد آرایه با استفاده از علائم `;`، `,` و `[]`
 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
```

فصل دوم: آرایه‌ها

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.

فصل دوم: آرایه‌ها

۲-۱-۲- ایجاد آرایه با استفاده از علامت ":" - ادامه...
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
```

فصل دوم: آرایه‌ها

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
```

فصل دوم: آرایه‌ها

۲-۱-۳- 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
```

فصل دوم: آرایه‌ها

۲-۱-۴- ماتریسهای 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

فصل دوم: آرایه‌ها

۲-۱-۴- ماتریسهای ویژه- ادامه...

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.

فصل دوم: آرایه‌ها

۲-۲- عملیات ریاضی بر روی آرایه‌ها

۱. عملیات اسکالر-آرایه: $-, +, ^, /, \backslash, *$
۲. عملیات عنصری: $-, +, .^, ./, .\, , *$
۳. عملیات ماتریسی: $-, +, ^, /, \backslash, *$ (بعداً توضیح داده خواهد شد)

فصل دوم: آرایه‌ها

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

فصل دوم: آرایه‌ها

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
```


فصل دوم: آرایه‌ها

۲-۳- ترانهاده یک ماتریس

برای محاسبه ترانهاده یک ماتریس از علامت ' استفاده می‌شود.
مثال:

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

```
>> b=a'  
      2      4      6  
      1      5      6  
      7     -1      0
```

فصل دوم: آرایه‌ها

۲-۴- 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
```

فصل دوم: آرایه‌ها

تمرین ۱-۲

۱. برنامه ای بنویسید که عدد صحیح n را از کاربر بگیرد و برداری 100 عنصری بین 0 و $2n\pi$ ایجاد نموده در متغیر x قرار دهد. سپس مقادیر y را از رابطه زیر محاسبه کرده نمایش دهد:

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

۲. برنامه فوق را طوری تغییر دهید که علاوه بر مقدار n ، عددی بین 0 و 1 را نیز از کاربر بگیرد و در متغیر جدید d قرار دهد. سپس بردار x را بین 0 و $2n\pi$ اما با گامهایی برابر با d محاسبه نماید.

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه

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

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
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>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
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>e=a([1,2,3],2)
```

```
e=
```

```
2
```

```
5
```

```
8
```

```
>>f=a(1:2:3 , 3:-2:1)
```

```
f=
```

```
3  1
```

```
9  7
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>>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
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

```
>> k=a( : , : )
```

```
k=
```

```
1  2  3
```

```
4  5  6
```

```
7  8  9
```

```
>> l=a(1:end,end)
```

```
l=
```

```
3
```

```
6
```

```
9
```


فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

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
```

فصل دوم: آرایه‌ها

۲-۵- استخراج بخشی از آرایه-ادامه-

نکته:

```
>>p=a( : )
```

```
p=
```

```
1
```

```
4
```

```
7
```

```
2
```

```
5
```

```
8
```

```
3
```

```
6
```

```
9
```

فصل دوم: آرایه‌ها

تمرین ۲-۲

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

۲. ماتریسی شامل ستونهای سوم تا هشتم و سطرهاى چهارم تا نهم ماتریس فوق ایجاد کنید.

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

فصل دوم: آرایه‌ها

۲-۶- حذف بخشی از آرایه

بمنظور 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
```

فصل دوم: آرایه‌ها

۲-۷- جستجوی زیرآرایه

بمنظور 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
```

فصل دوم: آرایه‌ها

۲-۷- جستجوی زیرآرایه-ادامه-

```
>>b=a(k)
```

```
b=
```

```
7
```

```
8
```

```
6
```

```
9
```

فصل دوم: آرایه‌ها

۲-۷- جستجوی زیرآرایه-ادامه-

دستور **find** در 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

فصل دوم: آرایه‌ها

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 باز می‌گرداند.
 - اگر با دو آرگومان ورودی بکار برده شود، بطوریکه آرگومان دوم ۱ یا ۲ باشد، بترتیب تعداد سطرها یا ستونهای ماتریس را باز می‌گرداند
 - اگر با یک آرگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را در یک بردار سطری دو عنصری باز می‌گرداند
 - اگر با دو آرگومان خروجی بکار برده شود، تعداد سطر و ستون ماتریس را بترتیب در آرگومان اول و دوم باز می‌گرداند

فصل دوم: آرایه‌ها

۲-۸- اندازه آرایه-ادامه-

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
```

فصل دوم: آرایه‌ها

۲-۸- اندازه آرایه-ادامه-

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
```

فصل دوم: آرایه‌ها

۲-۹-چند تابع برای دستکاری آرایه‌ها

- **flipud**: ماتریس را حول محور افقی 180° درجه می‌چرخاند.
- **fliplr**: ماتریس را حول محور عمودی 180° درجه می‌چرخاند
- **rot90**: ماتریس را در جهت مثلثاتی 90° 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

فصل دوم: آرایه‌ها

تمرین ۲-۳

۱. 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

MATLAB®

کلاس آموزشی

فصل سوم: توابع و عملیات 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. بدین منظور باید بردار سمت راست معادلات را بر ماتریس ضرایب متغیرها تقسیم کرد.

فصل سوم: توابع و عملیات ماتریسی

۳-۱- حل دستگاه معادلات خطی-ادامه
مثال:

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

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

فصل سوم: توابع و عملیات ماتریسی

۳-۲- تعدادی از توابع ماتریسی

- **det**: دترمینان ماتریس را محاسبه می کند
- **inv**: معکوس ماتریس را محاسبه می کند
- **pinv**: شبه معکوس یک ماتریس غیرمربعی را محاسبه می کند
- **trace**: مجموع عناصر قطر اصلی یک ماتریس را بازمی گرداند

فصل سوم: توابع و عملیات ماتریسی

تکلیف ۱-۳: 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 ۲-۳: 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

MATLAB®

کلاس آموزشی

فصل چهارم: عملیات منطقی و رابطه‌ای

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

۴-۱- عملگرهای رابطه‌ای

عملگرهای رابطه‌ای زیر در متلب تعریف شده‌اند:

$<$, $>$, $<=$, $>=$, $==$, \sim

فصل چهارم: عملیات منطقی و رابطه‌ای

۴-۱-۱- مقایسه دو آرایه

با استفاده از عملگرهای رابطه‌ای می‌توان دو آرایه را عنصر به عنصر با یکدیگر مقایسه کرد. 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];
>> tf = (a == b)
tf =
    0    1    0    1    0
```

متغیر **tf** یک متغیر از نوع منطقی (logical) خواهد بود. یعنی تنها می‌تواند مقادیر ۰ و ۱ را در خود نگهدارد. **بعنوان تمرین سعی کنید عنصر سوم tf را با ۵۰ جایگزین کنید.**

فصل چهارم: عملیات منطقی و رابطه‌ای

۴-۱-۲- 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
```

فصل چهارم: عملیات منطقی و رابطه‌ای

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

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

```
a=
```

```
1     2     3
4     2     2
1    10     0
```

```
>> a4 = a .* (a >= 3)
```

```
a4=
```

```
0     0     3
4     0     0
0    10     0
```

فصل چهارم: عملیات منطقی و رابطه‌ای

تکلیف ۴-۱: برنامه‌ای بنویسید که نمرات دروس ریاضی ۱(۴) واحد)، مکانیک(۳ واحد) و معارف اسلامی(۲ واحد) چند دانشجو را بصورت یک ماتریس ($n \times 3$) 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 ۱۰ بدون احتساب واحد هر درس

فصل چهارم: عملیات منطقی و رابطه‌ای

۴-۲- عملگرهای منطقی

عملگرهای $\&$ و $|$ و \sim عملگرهای منطقی در متلب هستند که به ترتیب معادل **AND** و **OR** و **NOT** می‌باشند.

فصل چهارم: عملیات منطقی و رابطه‌ای

۴-۲- عملگرهای منطقی (ادامه...)

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
```

فصل چهارم: عملیات منطقی و رابطه‌ای

۴-۳- توابع رابطه‌ای و منطقی

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

all (x) : تمامی عناصر یک بردار نامساوی ۰ باشد
مقدار ۱ و در غیر اینصورت ۰ باز می‌گرداند

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

xor(x,y) : یای انحصاری

فصل چهارم: عملیات منطقی و رابطهای

۴-۳- توابع رابطهای و منطقی-ادامه...
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 چهارم: عملیات منطقی و رابطهای

۴-۳- توابع رابطهای و منطقی-ادامه...
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

MATLAB®

کلاس آموزشی

فصل پنجم: متن

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.

فصل پنجم: متن

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
```

فصل پنجم: متن

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
```


فصل پنجم: متن

۵-۴- رفتار ماتریسی رشته‌ها

با رشته‌های کاراکتری متلب دقیقاً می‌توان مانند ماتریس‌های عددی رفتار کرد. مثلاً 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
```

فصل پنجم: متن

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'];
```

فصل پنجم: متن

۵-۶- ایجاد ماتریسهای کاراکتری (روش دوم)

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

```
>> line1='This is first line' ;
```

```
>> line2= 'This is second line';
```

```
>> sm=char(line1,line2)
```

```
sm=
```

```
    This is first line
```

```
    This is second line
```

فصل پنجم: متن

۵-۷- 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.

فصل پنجم: متن

۵-۷- گرفتن رشته در حین اجرای برنامه-ادامه-
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
```

فصل پنجم: متن

۵-۸- سایر توابع کار با رشته‌ها

<code>strcmp(s1,s2)</code>	: در صورتیکه دو رشته یکسان باشند ۱ و در غیر این صورت ۰ باز می‌گرداند
<code>upper</code>	: تمامی حروف یک رشته را به حروف بزرگ تبدیل می‌کند
<code>lower</code>	: تمامی حروف یک رشته را به حروف کوچک تبدیل می‌کند
<code>num2str</code>	: تبدیل عدد به رشته عددی
<code>str2num</code>	: تبدیل رشته عددی به عدد
<code>mat2str</code>	: تبدیل ماتریسی از اعداد به رشته
<code>eval</code>	: 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.

فصل پنجم: متن

۵-۸- سایر توابع کار با رشته‌ها-ادامه...
مثال:

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

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

Your number is 12 . Thank you!

فصل پنجم: متن

تکلیف ۱-۵: 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 ۲-۵: برنامه‌ای بنویسید که یک رشته کاراکتری را از کاربر بگیرد و با تغییر کد اسکی آن، آنرا بصورت رمز در آورده نمایش دهد.

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

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



MATLAB®

کلاس آموزشی

فصل ششم: تصمیم‌گیری و کنترل روند

در این فصل در مورد جملات شرطی و انواع حلقه‌های تکرار صحبت خواهیم کرد.

۶-۱- حلقه for:

شکل کلی حلقه for در متلب بصورت زیر است:

```
for x = آرایه  
    دستورات  
end
```

در اینصورت حلقه فوق به تعداد ستونهای آرایه مشخص شده 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.

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۱- حلقه for-ادامه-
مثال:

```
for n=1:10
    x(n) = sin(n * pi / 10);
end;
-----
for k=[1,2,3,7]
    x(k) = k+1;
end;
>>x
x=
    2    3    4    0    0    0    0    8
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۲- حلقه while :

در مواردی که بخواهیم یک یا چند دستور تا برقراری شرط خاصی تکرار گردند از این حلقه استفاده می‌کنیم. شکل کلی حلقه **while** بصورت زیر است:

while شرط

دستورات

end

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

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۲- حلقه while – ادامه –

مثال:

```
t=1;
while t ~= -1
    t = input( ' Enter a number to continue or -1 to exit from
               this block: ');
    ...
end
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۲- حلقه while – ادامه-

مثال:

در این مثال بالاترین دقت محاسبات نقطه اعشار در متلب محاسبه می‌شود

```
Epsilon=1;
while ( 1 + Epsilon ) > 1
    Epsilon = Epsilon / 2;
end;
disp('This is the smallest floating point number in matlab: ');
disp ( Epsilon);
```

نکته : متغیر ویژه eps در متلب حاوی کوچکترین عددی است که اگر با یک جمع شود مقدار حاصل از یک بزرگتر خواهد بود. که این عدد دو برابر Epsilon بدست آمده از روش فوق است.

فصل ششم: تصمیم‌گیری و کنترل روند

۳-۶- 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;
```

فصل ششم: تصمیم‌گیری و کنترل روند

۶-۳- ساختار if-else-end – ادامه –
مثال:

```
Epsilon = 1;  
while 1 > 0  
    Epsilon = Epsilon / 2;  
    if Epsilon + 1 == 1  
        break;  
    end  
end
```

نکته: با دستور **break** می‌توان یک حلقه **while** یا **for** را شکست. در اینصورت اجرای برنامه از نخستین دستور بعد از حلقه ادامه خواهد یافت. □

فصل ششم: تصمیم‌گیری و کنترل روند

تکلیف ۱-۶: 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 ۲-۶: برنامه دیگری بنویسید که همان کارهای برنامه ۱-۶ را بدون استفاده از حلقه انجام دهد.

فصل هفتم:
ایجاد توابع در متلب
Functions

MATLAB®

کلاس آموزشی

فصل هفتم: ایجاد توابع در متلب

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

۱. سرعت بالاتر

۲. صرفه‌جویی در حافظه کامپیوتر

۳. 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

فصل هفتم: ایجاد توابع در متلب

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

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 اشکالزدایی آن آسانتر باشد اما پس از کامل شده برنامه آنرا به فانکشن تبدیل کنیم تا سرعت و کیفیت آن افزایش یابد.

فصل هفتم: ایجاد توابع در متلب

۷-۲- تفاوت‌های توابع و فایل‌های متنی

۱. فایل‌های متنی سطر به سطر ترجمه و اجرا می‌شوند اما توابع یکبار بطور کامل ترجمه و سپس اجرا می‌گردند.
۲. محیط کاری فایل‌های متنی همان محیط کاری متلب است اما محیط کاری تابعی مختص خود اوست یعنی اگر تغییری 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)
۳. توابع تنها از طریق آرگومان‌هایشان با محیط خارج در ارتباطند

فصل هفتم: ایجاد توابع در متلب

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

...

فصل هفتم: ایجاد توابع در متلب

۷-۳- نحوه ایجاد توابع-ادامه-

نکات:

۱. تابع ممکن است هیچ آرگومان ورودی یا خروجی نداشته باشد.
۲. اولین سطر بعد از اعلان تابع، یک جمله توضیحی است که در هنگام استفاده از دستور **lookfor** در متلب مورد جستجو قرار می گیرد
۳. 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.

فصل هفتم: ایجاد توابع در متلب

۷-۴- 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 می‌تواند یک پیغام خطا نیز بمنظور آگاهسازی 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);
```


فصل هفتم: ایجاد توابع در متلب

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. مثلاً 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 در صورتیکه با تعداد درخواست شده برابر نباشند پیام خطای مناسب را نشان می‌دهند.

فصل هفتم: ایجاد توابع در متلب

۷-۶- 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.

فصل هفتم: ایجاد توابع در متلب

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 در صورتیکه 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 $\text{plot}(x,y)$ استفاده کنید.

فصل هفتم: ایجاد توابع در متلب

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



MATLAB®

کلاس آموزشی

فصل هشتم: تجزیه و تحلیل 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
```

فصل هشتم: تجزیه و تحلیل فوریه

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)

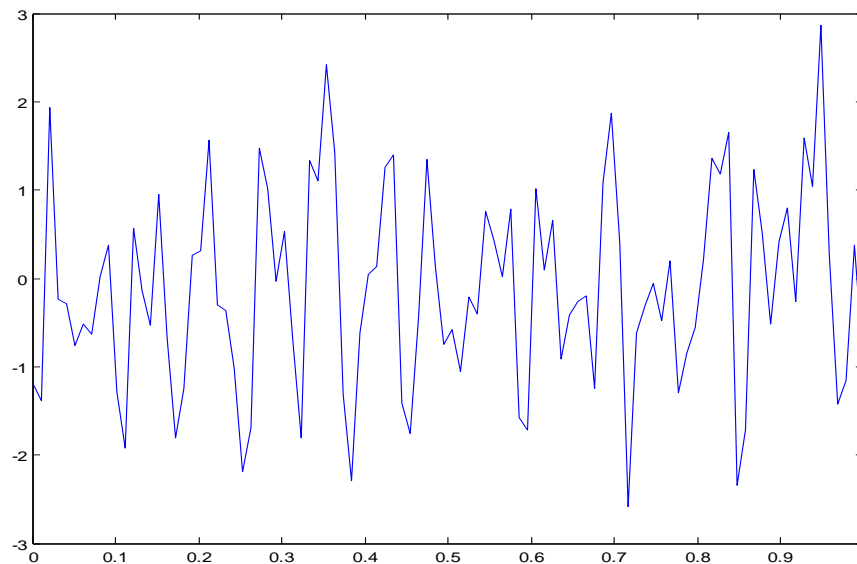
```
>> t= 0 : 1/99 : 1; بردار زمان
```

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

فصل هشتم: تجزیه و تحلیل فوریه

۸-۲-مثالی از کاربرد تبدیل فوریه-ادامه

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



فصل هشتم: تجزیه و تحلیل فوریه

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.

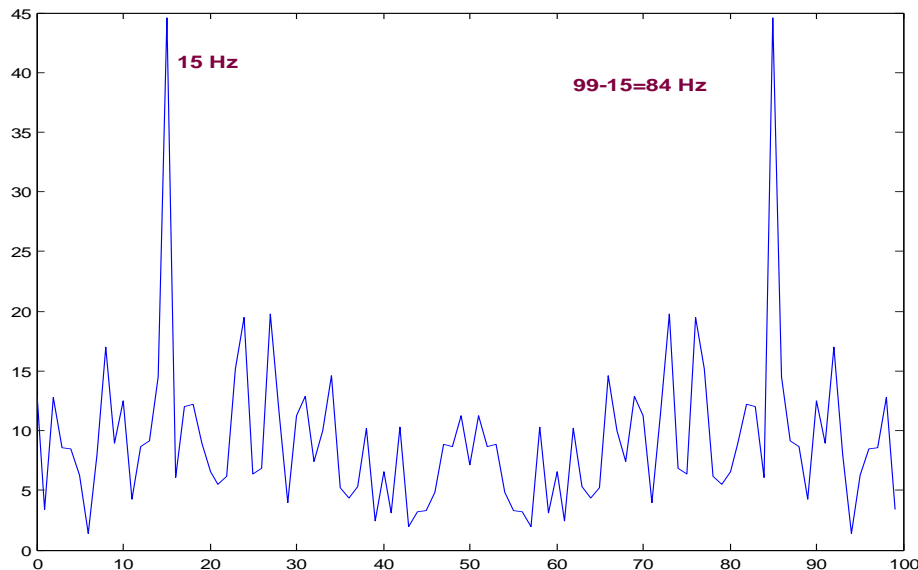
f: بردار فرکانس است که بین ۰ تا ۹۹ تغییر می‌کند

فصل هشتم: تجزیه و تحلیل فوریه

۸-۲-مثالی از کاربرد تبدیل فوریه-ادامه

`>>plot(f , fy)`

نمودار طیف فوریه:



فصل نهم: نمودارهای دو بعدی



کلاس آموزشی

فصل نهم: 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`

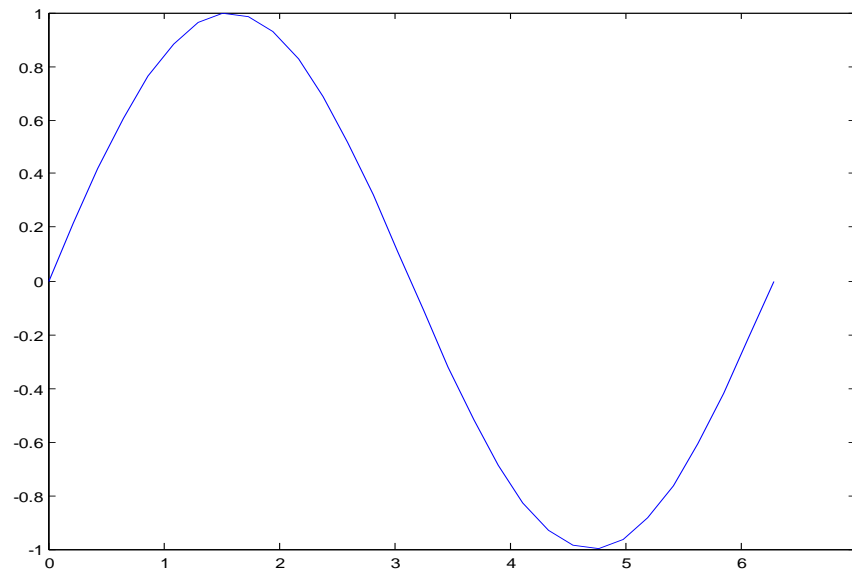
که به ترتیب معرف رنگهای زرد، سرخابی، فیروزه‌ای، قرمز، سبز، آبی، سفید و سیاه می‌باشد

فصل نهم: نمودارهای دوبعدی

۹-۱- تابع plot – ادامه

مثال:

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



فصل نهم: نمودارهای دوبعدی

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 بعد از 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.

فصل نهم: نمودارهای دوبعدی

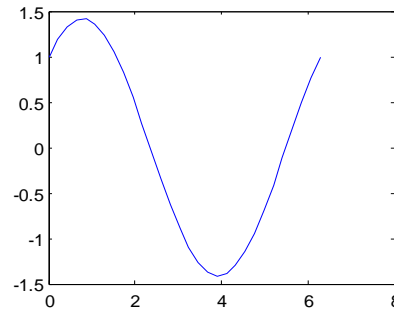
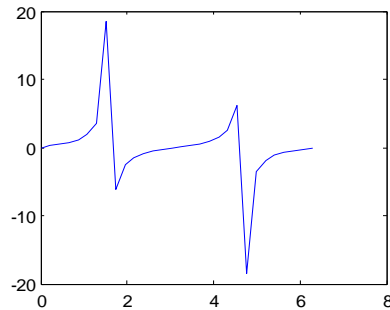
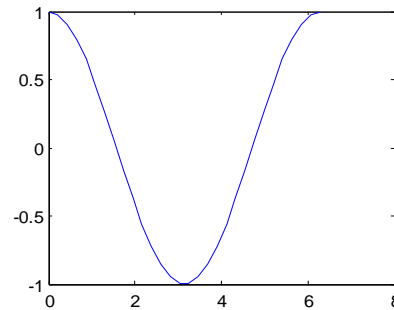
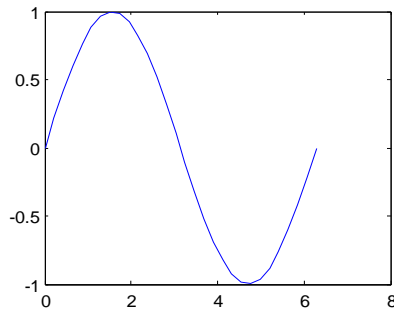
۹-۲- رسم چند نمودار مجزا در یک پنجره شکل-ادامه

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));
```

فصل نهم: نمودارهای دوبعدی

۹-۲- رسم چند نمودار مجزا در یک پنجره شکل-ادامه
مثال:-ادامه-



فصل نهم: نمودارهای دوبعدی

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

بمنظور برچسب گذاری محورها و ایجاد عنوان برای نمودار می توان 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.

فصل نهم: نمودارهای دوبعدی

۹-۴- 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** بصورت‌های 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

فصل نهم: نمودارهای دوبعدی

۹-۵- ایجاد پنجره شکل جدید

بصورت پیش فرض 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.

فصل نهم: نمودارهای دوبعدی

۹-۶- 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.

فصل نهم: نمودارهای دوبعدی

۷-۹- افزودن راهنمای علائم: دستور **legend**

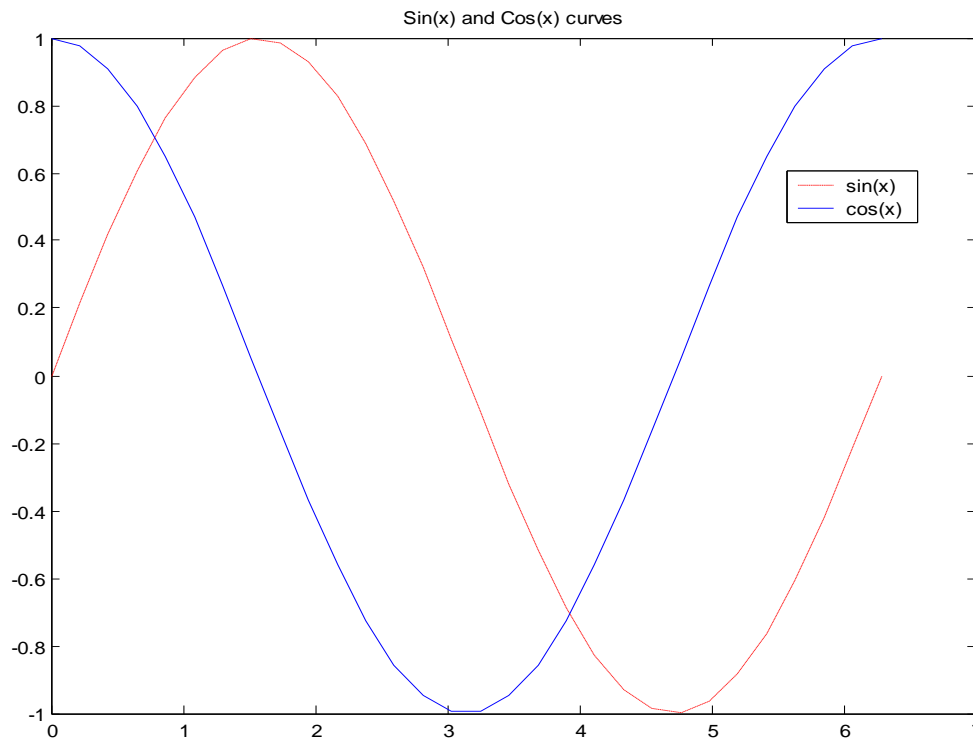
مثال:

```
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');
```

فصل نهم: نمودارهای دوبعدی

۹-۸- افزودن راهنمای علائم: دستور **legend**

مثال-ادامه:



فصل نهم: نمودارهای دوبعدی

۹-۹-دستور **axis**

با استفاده از این دستور می توان دامنه ترسیم را تغییر داد:

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

نمودار در دامنه **xmin** تا **xmax** ، **ymin** تا **ymax** و... ترسیم می گردد.

axis off	محورهای مختصات را حذف می کند
axis on	محورهای مختصات را ترسیم می کند

فصل نهم: نمودارهای دوبعدی

۹-۱۰- ثابت نگهداشتن نمودار: دستور **hold**

بصورت پیش فرض 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

فصل نهم: نمودارهای دوبعدی

۹-۱۱- سایر دستورات

clf: محتویات پنجره شکل جاری را پاک می کند

cla: محتویات نمودار جاری را پاک میکند

zoom: حالت زوم را فعال یا غیر فعال می کند

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

فصل نهم: نمودارهای دوبعدی

۹-۱۲- 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 نمودار پله‌ای

hist: ترسیم نمودار فراوانی

bar: ترسیم نمودار میله‌ای

فصل دهم چند جمله ایها

The MATLAB logo is displayed in a blue serif font. It is positioned on the left side of the slide, below the title and above the course name. The background of the slide features a 3D surface plot with a color gradient from blue to red, set against a light gray grid pattern.

MATLAB®

کلاس آموزشی

فصل دهم: چند جمله‌ایها

۱۰-۱- تعریف یک چند جمله‌ای در متلب

در متلب یک چند جمله‌ای توسط یک بردار سطری تعریف می‌شود.

مثال:

$$>> p = [1 \ 2 \ 3 \ 0 \ -5] \equiv p = x^4 + 2x^3 + 3x^2 + 0x - 5$$

۱۰-۲- یافتن ریشه‌های چند جمله‌ای

با استفاده از تابع **roots** می‌توان ریشه‌های یک چند جمله‌ای را بدست آورد:

مثال:

```
>> r = roots(p)
p = -0.7202 + 1.7518i
     -0.7202 - 1.7518i
     -1.4930
     0.9335
```

فصل دهم: چند جمله‌ایها

۱۰-۳- یافتن یک چند جمله‌ای با استفاده از ریشه‌هایش

با استفاده از تابع **poly** می‌توان یک چند جمله‌ای را از روی ریشه‌هایش بدست آورد.
مثال:

```
>> r=[1 ; -1];  
>> p=poly(r)  
p=  
1 0 -1
```

نکته: بر خلاف خود چند جمله‌ای ریشه‌های چند جمله‌ای باید بصورت یک بردار ستونی تعریف شوند.

فصل دهم: چند جمله‌ایها

۱۰-۴- ضرب چند جمله‌ایها

بمنظور ضرب دو چند جمله‌ای می‌توان از تابع **conv** استفاده کرد.

مثال:

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

```
>> c = conv(a , b)
```

```
c =
```

```
1 6 20 50 75 84 64
```

فصل دهم: چند جمله‌ایها

۱۰-۵- جمع و تفریق چند جمله‌ایها

برای اینکه بتوان دو بردار را با یکدیگر جمع یا تفریق کرد باید آن دو بردار هم طول باشند. لذا در صورت لزوم باید ضرایبی که تنها در یکی از چند جمله‌ایها وجود دارد را در چند دوم برابر با صفر قرار داد تا دو چند جمله‌ای هم طول شوند.

مثال:

```
>>p1= [4 5 3 2]
>>p2= [0 5 2 0]
>>p_sum=p1+p2
p_sum=
4 10 5 2
```

فصل دهم: چند جمله‌ایها

۱۰-۶- تقسیم چند جمله‌ایها

با تابع **deconv** می‌توان دو چند جمله‌ای را بر یکدیگر تقسیم کرد. این تابع دو آرگومان خروجی می‌گیرد که اولی خارج قسمت و دومی باقیمانده تقسیم خواهد بود.

```
>>a=[ 1 2 3 4 5 6];
>> b=[ 2 3 4];
>> [q , r] = deconv( a , b )
```

```
q =
    0.5000    0.2500    0.1250    1.3125
```

```
r =
     0         0         0         0    0.5625    0.7500
```


فصل دهم: چند جمله‌ایها

۱۰-۷- مشتق چند جمله‌ای

با استفاده از تابع **polyder** می‌توان مشتق یک چند جمله‌ای را بدست آورد

مثال:

```
>> g = [1 6 20 48 69 72 44]
>> h= polyder(g)
h=
    6    30    80   144   138    72
```

فصل دهم: چند جمله‌ایها

۱۰-۸- محاسبه چند جمله‌ای

بمنظور محاسبه مقادیر چند جمله‌ای در یک یا چند نقطه از تابع `polyval` می‌توان استفاده کرد.

مثال:

```
>> p = [1 4 -7 -10]
>> x = linspace(-1,3,100);
>> y = polyval(p , x);
>> plot(x , y);
```

فصل دهم: چند جمله‌ایها

تکلیف ۱۰-۱- تابعی بنویسید که یک چند جمله‌ای (به صورت یک بردار سطری) و یک بردار دو عنصری (حاوی دامنه ترسیم) را از کاربر بگیرد و نمودار چند جمله‌ای را در دامنه مشخص شده ترسیم کند.

تکلیف ۱۰-۲- تابع فوق را طوری تغییر دهید که اگر با دو آرگومان خروجی بکار برده شود به جای ترسیم، مقادیر X و Y محاسبه شده را بازگرداند.

فصل یازدهم برازش منحنی و درونیابی



MATLAB®

کلاس آموزشی

فصل یازدهم: برازش منحنی و درونیابی

۱۱-۱- برازش منحنی: تابع `polyfit`

با استفاده از تابع `polyfit` می‌توان بهترین منحنی گذرنده از چند نقطه را بدست آورد. این تابع چند جمله‌ای معرف منحنی فوق را بعنوان آرگومان خروجی باز می‌گرداند. شکل کلی استفاده از این تابع بصورت زیر است:

`P= polyfit(x , y , n)`

که در این رابطه، `x` و `y` نقاط معلوم و `n` درجه چندجمله‌ای مطلوب است.
مثال:

```
>> x= [ 1 2 5 7]; y=[10 22 48 75];
```

```
>> p= polyfit(x,y,1)
```

```
p=  
10.45 -0.4396
```

فصل یازدهم: برازش منحنی و درونیابی

۱۱-۲- درونیابی یک بعدی: تابع `interp1`

تفاوت درونیابی با برازش آن است که در برازش منحنی لزوماً خود نقاط اولیه بر روی منحنی برازش شده قرار ندارند اما در درونیابی، نقاط اولیه جزیی از منحنی مورد استفاده برای درونیابی می‌باشند. شکل کلی استفاده از تابع `interp1` بصورت زیر است:

```
y_new = interp1 (x , y, x_new , ['method'])
```

که در این رابطه x, y نقاط اولیه، x_new مقادیری از x است که باید مقادیر y ان درونیابی شوند و y_new مقادیر درونیابی شده می‌باشند. `method` می‌تواند یکی از مقادیر زیر باشد:

'nearest' , 'linear', 'spline', 'pchip', 'cubic',
'cubic5v'

فصل یازدهم: برازش منحنی و درونیابی

۱۱-۲- درونیابی یک بعدی-ادامه-

مثال:

```
>> h = 1:12;  
>> temps = [5 8 9 15 25 29 31 30 22 25 27 24];  
>> plot ( h, temps); % عملا درونیابی خطی بکار برده می شود  
>> h_new=1.5;  
>> t_new = interp1(h , temps , h_new)  
t_new=  
    6.5;
```

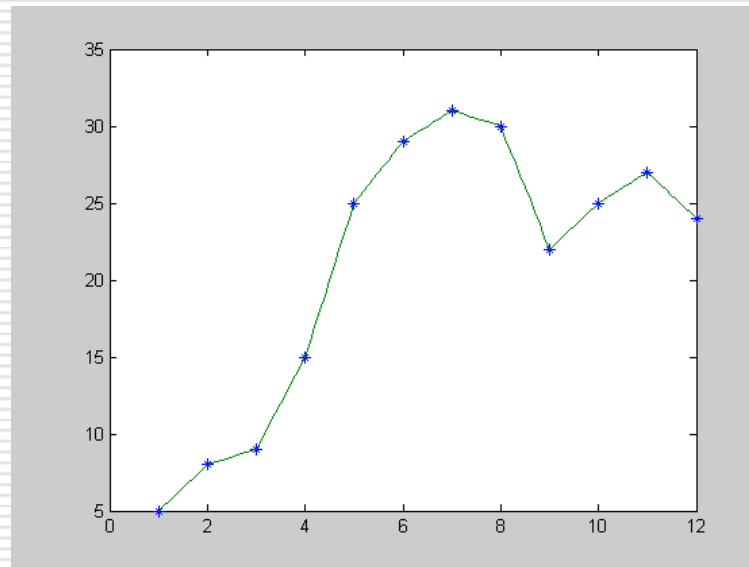
ادامه ...

فصل یازدهم: برازش منحنی و درونیابی

۱۱-۲- درونیابی یک بعدی-ادامه-

ادامه مثال:

```
>> h_new2 = 1: 0.1 : 12;  
>> t_new2 = interp1(h , temps , h_new2);  
>> plot( h, temps , '*' , h_new2 , t_new2);
```

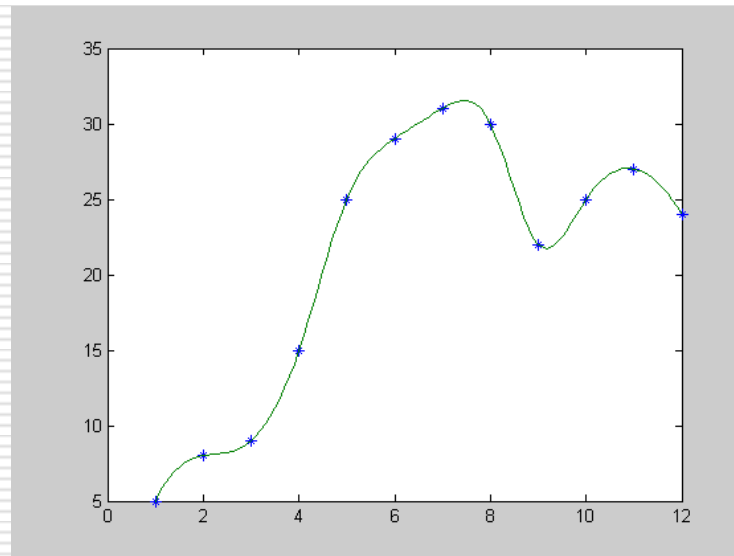


فصل یازدهم: برازش منحنی و درونیابی

۱۱-۲- درونیابی یک بعدی-ادامه-

ادامه مثال:

```
>> t_spline = interp1(h , temps , h_new2 , 'spline');  
>> plot(h , temps , '*' , h_new2 , t_spline);
```



فصل یازدهم: برازش منحنی و درونیابی

۱۱-۳- درونیابی دو بعدی: تابع interp2

شکل کلی استفاد از تابع:

`z_new = interp2(x, y, z, x_new, y_new, ['method'])`

method می‌تواند یکی از مقادیر زیر باشد:

`'nearest' , 'linear' , 'spline' , 'cubic'`

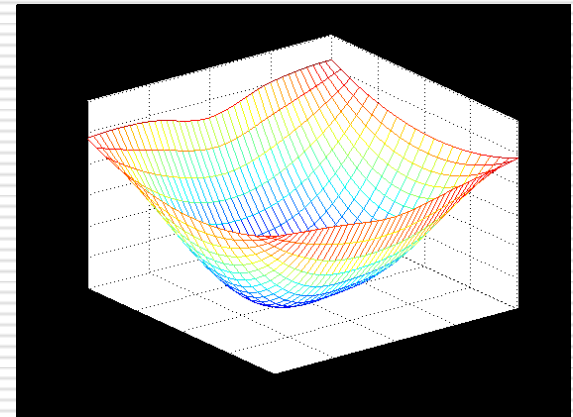
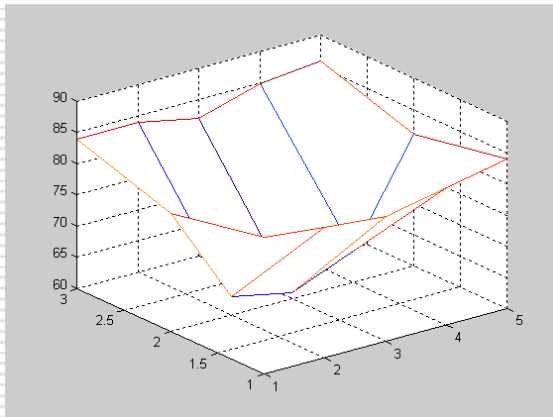
مثال:

```
>> w=1:5; d=1:3;
>> t = [82 81 80 82 84
        79 63 61 65 81
        84 84 82 85 86];
>> w_new = 1:0.1:5;
>> d_new = 1:0.1:3;
>> t_new = interp2(w, d, t, w_new, d_new, 'cubic');
```

فصل یازدهم: برازش منحنی و درونیابی

۱۱-۳- درونیابی دو بعدی - ادامه
ادامه مثال:

```
>> mesh(w,d,t);  
>> figure;mesh(w_new,d_new,t_new)
```



فصل دوازدهم: نمودارهای سه بعدی



MATLAB®

کلاس آموزشی

فصل دوازدهم: نمودارهای سه‌بعدی

۱۱-۱- خمهای فضایی - تابع `plot3`

با استفاده از تابع `plot3` در متلب می‌توان یک منحنی را در فضای سه‌بعدی ترسیم کرد. روش استفاده از این تابع بسیار شبیه تابع `plot` است. جز اینکه برای هر منحنی به سه بردار هم طول نیاز است.

مثال: رسم یک فنر با شعاع برابر با یک:

$$x=t$$

$$y= \sin(t)$$

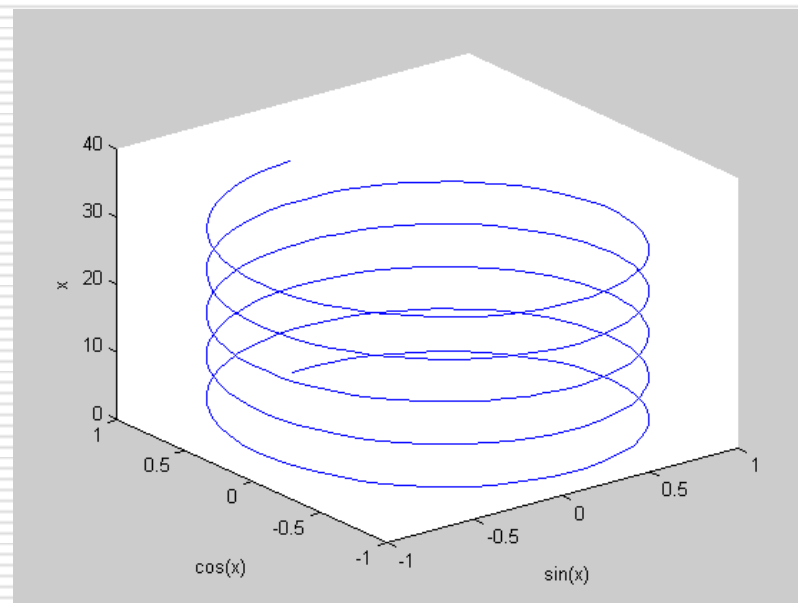
$$z=\cos(t)$$

$$t \in R$$

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۱- خمهای فضایی-ادامه

```
>>t=0: pi/50:10*pi;  
>>plot3(sin(t) , cos(t) , t); xlabel('sin(x)'); ylabel('cos(x)'); zlabel('x')
```



فصل دوازدهم: نمودارهای سه بعدی

۱۱-۲- تغییر زاویه دید

با استفاده از دکمه فشاری **Rotate 3D** بر روی هر پنجره شکل متلب و یا با استفاده از دستور **view** می توان زاویه دید را تغییر داد. همچنین در منوی **Tools** با استفاده از زیرمنوهای **Camera Motion** می توان در فضای سه بعدی حرکت کرد.

شکل کلی استفاده از دستور **view** بصورت زیر است:

view([az , el])

که در این رابطه **az** و **el** به ترتیب زاویه دوربین نسبت به صفحه **XY** و بخش منفی محور **Y** است.

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۳- نمودارهای شبکه‌ای: توابع `mesh`, `meshc`, `meshz`

با استفاده از این توابع می‌توان سطوح شبکه‌ای (یا توری) ایجاد کرد. شکل کلی استفاده از تابع `mesh` بصورت زیر است:

`mesh(x,y,z)`

که در این رابطه `Z` تابعی دو متغیره از متغیرهای `X` و `Y` می‌باشد. بنابراین لازم است که `Z` یک ماتریس دو بعدی باشد که تعداد سطرهای آن برابر با تعداد عناصر `Y` و تعداد ستونهایش برابر با تعداد عناصر `X` باشد. `X` و `Y` باید بردار باشند اگرچه می‌توانند ماتریس‌هایی هم‌بعد نیز باشند بدینصورت که بردار `X` به تعداد عناصر بردار `Y` بصورت سطری تکرار شود و بردار `Y` به تعداد عناصر `X` بصورت ستونی تکرار گردد. که در اینصورت دو ماتریس هم‌بعد خواهیم داشت. تابع `meshgrid` می‌تواند این عمل را انجام دهد:

`[x_new,y_new]=meshgrid(x,y);`

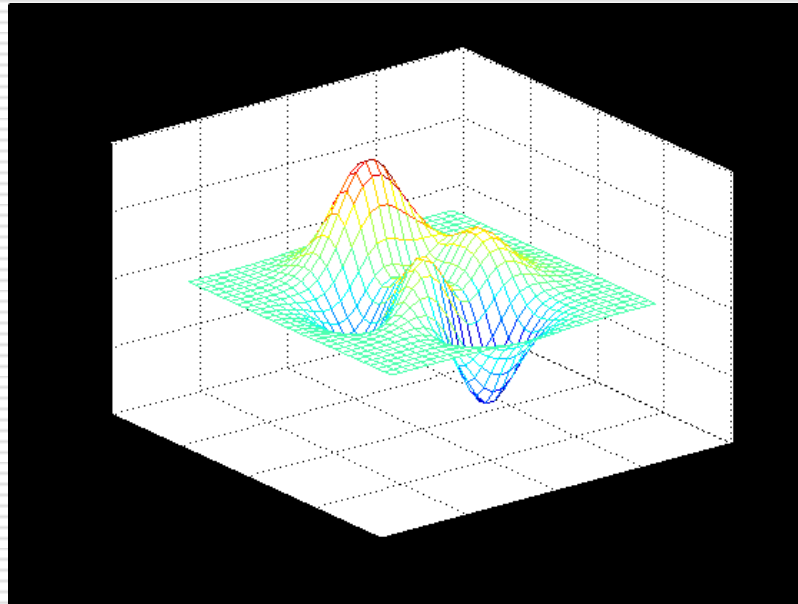
تابع `meshc` علاوه بر نمودار شبکه‌ای، نمودارهای تراز را نیز رسم می‌کند.
تابع `meshz` دیواره‌هایی را در پایین نمودار به سمت صفحه `X-Y` رسم می‌کند.

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۳- نمودارهای شبکه‌ای - ادامه -

مثال ۱ : تابع **peaks** یکی از توابع متلب است که یک مدل ریاضی از پیش تعریف شده را ایجاد می‌کند:

```
>> [x ,y,z]= peaks(30);  
>> mesh(x,y,z);
```

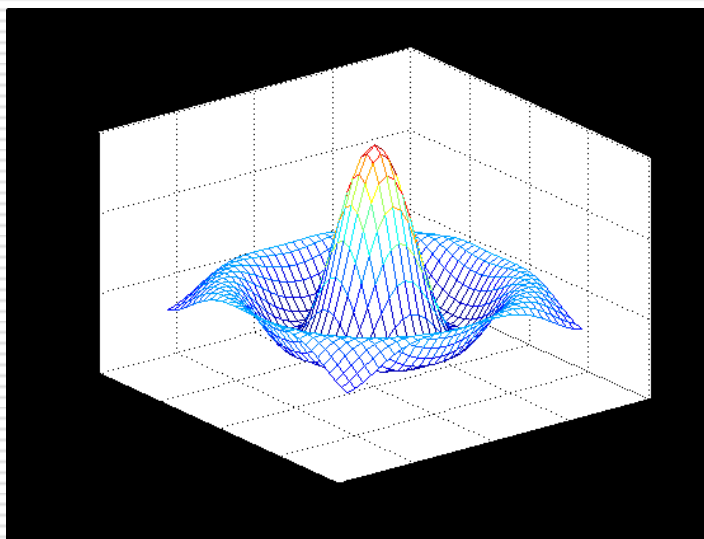


فصل دوازدهم: نمودارهای سه بعدی

۱۱-۳- نمودارهای شبکه‌ای - ادامه -

مثال ۲: رسم یک تابع دو بعدی به فرمول $z = \sin(r)/r$ که $r = \sqrt{X^2 + Y^2}$

```
>>x=-7.5: 0.5: 7.5; y=x;
>> [x_new,y_new]=meshgrid(x,y);
>> r = sqrt(x_new.^2 + y_new.^2) + eps;
>>z=sin(r) ./ r; mesh(x_new , y_new , z) یا mesh(x,y,z);
```



فصل دوازدهم: نمودارهای سه بعدی

۱۱-۳- نمودارهای شبکه‌ای - ادامه -

به صورت پیش فرض متلب نقاطی از نمودار سه بعدی که پشت نقاط جلویی قرار گرفته‌اند را مخفی می‌کند. با استفاده از تابع **hidden** می‌توان این رفتار را تغییر داد:

>> hidden off

>> hidden on

>> hidden

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۴- نمودارهای رویه: surf و surfl و surfc

تفاوت نمودارهای رویه با نمودارهای شبکه‌ای آن است که در اینجا بجای خطوط شبکه‌ای، وصله‌ها یا فواصل بین شبکه‌ها ترسیم می‌گردد.

تابع surf دقیقاً مانند تابع mesh بکار برده می‌شود.

تابع surfl علاوه بر تواناییهای تابع surf می‌تواند خصوصیات نوردهی شیء سه بعدی را نیز تنظیم کند. از قبیل جهت منبع نوری و خصوصیات انعکاسی شیء

تابع surfc مانند تابع meshc عمل می‌کند

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۴- نمودارهای رویه-ادامه

تابع shading :

با استفاده از تابع shading می توان نوع سایه رنگ نمودار را تعیین کرد. این تابع باید پس از یکی از توابع surf بیاید و با یکی از سه پارامتر زیر قابل فراخوانی است:

shading flat

shading interp

shading faceted

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۴- نمودارهای رویه-ادامه

تابع colormap :

با استفاده از این تابع می توان رنگهای بکار برده شده برای ترسیم نمودار رویه را تعریف کرد تنها پارامتر مورد نیاز می تواند یکی از ثابتهای زیر باشد:

hsv, cool , hot, prism, gray

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۵- چرخش دوربین در فضای سه بعدی

با استفاده از تابع `camorbit` می توان پس از رسم نمودار سه بعدی، زاویه دید دوربین نسبت به شیء را تغییر داد:

`camorbit(theta,phi)`

`theta`: زاویه چرخش افقی

`phi`: زاویه چرخش عمودی

و یا می توانید از این فرم تابع استفاده کنید:

`camorbit(theta,phi,'محور چرخش');`

آرگومان سوم محوری است که چرخش حول آن انجام می شود را مشخص می کند که به صورت پیش فرض محور Z است.

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۵- چرخش دوربین در فضای سه بعدی - ادامه...

مثال:

```
surf(peaks)
axis vis3d
axis off
for i=1:360
    camorbit(2,0,'data')
    drawnow
end
```


فصل دوازدهم: نمودارهای سه بعدی

۱۱-۶- تعیین موقعیت دوربین

به منظور تعیین موقعیت دوربین می توان از تابع `campos` استفاده کرد:
آرگومان ورودی این تابع موقعیت دوربین را در فضای `campos([x,y,z])` سه بعدی مشخص می کند.

مثال:

```
surf(peaks)
axis vis3d off
for x = -200:5:200
    campos([x,5,10])
    drawnow
end
```

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۷- نمایش میله رنگ

با استفاده از تابع `colorbar` می توان پس از نمایش نمودار، رنگهای بکار برده شده در آن را بصورت یک میله رنگ در کنار نمودار نمایش داد:

```
>> surf(peaks)
```

```
>> colorbar
```

اعداد نمایش داده شده در کنار میله رنگ مقداری را که هر رنگ به آن اشاره می کند(به صورت پیش فرض مقدار Z) را نمایش می دهد.

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۸- نمودارهای تراز- contour3 , contour

نمودارهای تراز خطوط و منحنی‌های بسته‌ای می‌باشند که برای نمایش ارتفاع‌های مختلف یک نمودار سه بعدی، در فضای دوبعدی یا سه بعدی بکار برده می‌شوند. در این نمودارها، نواحی هم‌ارتفاع توسط خطوطی به یکدیگر متصل می‌شوند.

نحوه استفاده از توابع contour و contour3 نیز دقیقا شبیه تابع mesh است.

مثال:

```
[c,h] = contour(peaks); clabel(c,h), colorbar
```

فصل دوازدهم: نمودارهای سه‌بعدی

۱۱-۹- تابع pcolor

این تابع داده‌های مربوط به یک نمودار سه‌بعدی را در فضای دوبعدی رسم می‌کند و برای نمایش ارتفاع (مقدار Z) از تغییرات رنگ استفاده می‌نماید. نتیجه حاصل از این تابع، مشابه تصویر از بالای تابع surf است.

مثال ۱:

```
>> pcolor(peaks)
```

مثال ۲:

```
t=tril(ones(10));
ts=[t,flipr(t)];
tss=[flipud(ts);ts];
pcolor(tss); colormap(gray(2))
```

فصل دوازدهم: نمودارهای سه بعدی

۱۱-۹- اجرای برنامه‌های نمایشی متلب در زمینه ترسیم سه بعدی

demo های Graphics در متلب راهنمای بسیار خوبی برای آموزش قابلیت‌های گرافیکی متلب می‌باشد. این برنامه‌ها از منوی **Start** متلب در شاخه **Demo-> Graphics** در دسترسند.

فصل سیزدهم پردازش تصویر



MATLAB®

کلاس آموزشی



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱-مقدمه

- در متلب تصاویر بصورت ماتریسهای دو، سه و یا چهاربعدی تعریف می‌شوند.
 - کیفیت تصویر: کیفیت تصویر به دو پارامتر یکی دقت ابعادی و دیگری 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 معمولاً چنین نیست. واحد دقت ابعادی dpi یا نقطه بر اینچ است.
-

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱- مقدمه-ادامه-

□ انواع تصاویر: انواع تصاویر عبارتند از :

■ تصاویر اندیس دار

■ تصاویر شدت

■ تصاویر باینری

■ تصاویر RGB

■ تصاویر چندفریمی

که در ادامه فصل مفصلاً به هریک خواهیم پرداخت

□ فرمت‌های گرافیکی: تصاویر با فرمت‌های مختلفی می‌توانند بر روی دیسک ذخیره شوند. مهم‌ترین فرمت‌های گرافیکی در زمان حاضر عبارتند از: BMP, JPG, PNG, GIF, TIFF که تمامی آنها به‌علاوه چندین فرمت دیگر توسط متلب پشتیبانی می‌شوند.

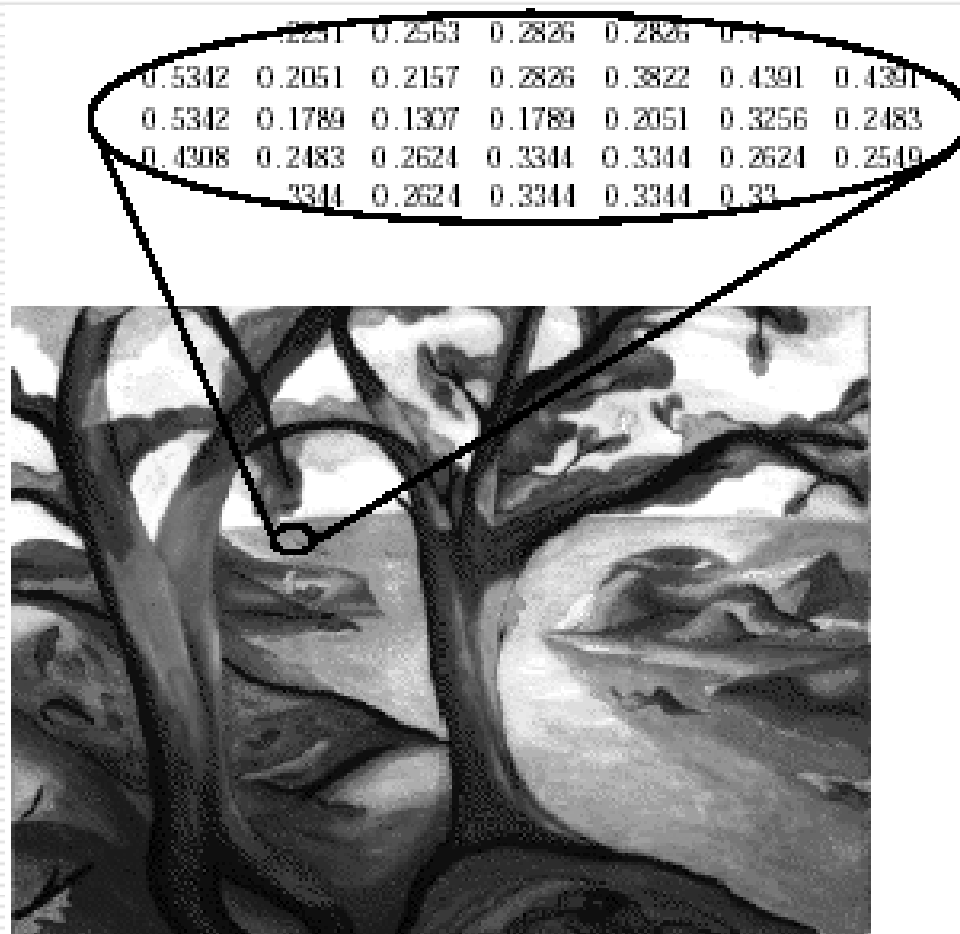
فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۲- انواع تصاویر

تصاویر شدت (Intensity Image)

تصویر شدت یا تصویر سطح خاکستری، به تصویری گفته می‌شود که تنها دارای مقادیر روشنایی باشد و فاقد خصوصیات رنگ مانند: فام و خلوص باشد. در متلب این تصاویر 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 و یا بین ۰ تا ۲۵۵ تغییر کند. در حالت اول داده‌های ماتریس از نوع دقت مضاعف و در حالت دوم از نوع `uint8` خواهد بود. بجز توابع تعریف شده در جعبه‌ابزار `images` 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 که میزان حافظه مورد نیاز آن چهار برابر نوع `uint8` است.

فصل سیزدهم: جعبه ابزار پردازش تصویر



۱۳-۲- انواع تصاویر-ادامه
تصاویر شدت-ادامه
نمونه‌ای از یک تصویر شدت:

فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۲- انواع تصاویر-ادامه

تصاویر اندیس‌شده (Indexed Image)

این تصاویر توسط دو ماتریس زیر مشخص می‌شوند:

۱. ماتریس اندیس: ماتریسی است که ابعاد آن برابر با ابعاد تصویر بر حسب پیکسل می‌باشد. مقادیر این ماتریس معمولاً بین ۱ تا ۲۵۶ تغییر می‌کند و مقدار هر درایه از این ماتریس معرف شماره سطری از ماتریس نقشه‌رنگ است.

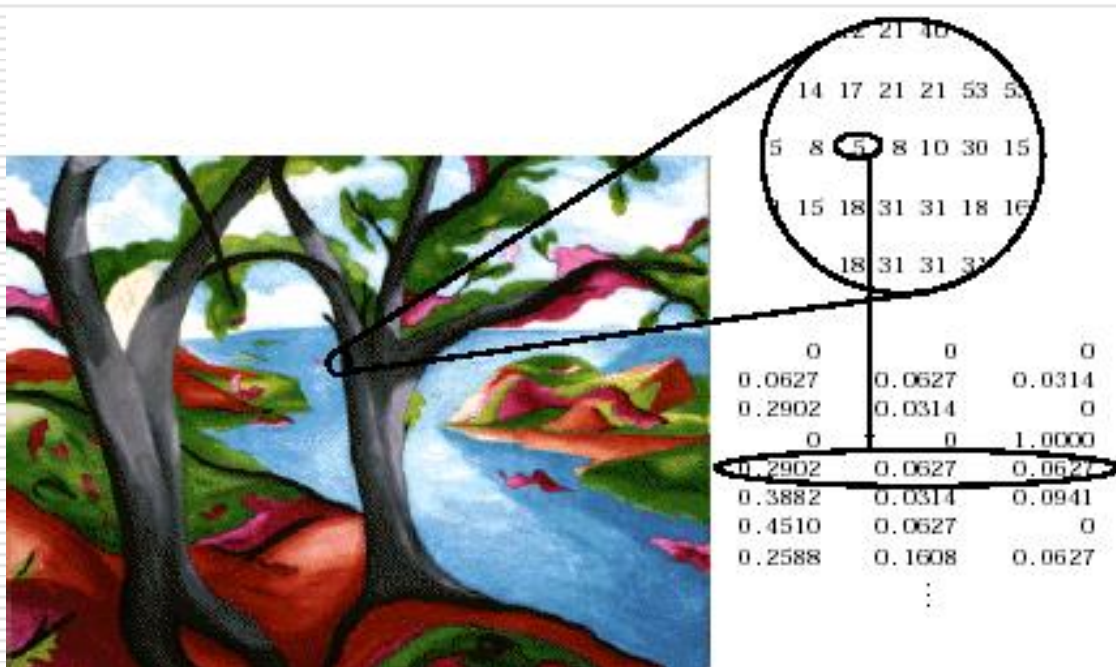
۲. ماتریس نقشه‌رنگ (map): این ماتریس دارای ۳ ستون می‌باشد و هر سطر از آن معرف یکی از رنگهای موجود در تصویر است. بطوریکه عنصر اول هر سطر معرف نسبت اولیه قرمز، عنصر دوم معرف اولیه سبز و عنصر سوم معرف اولیه آبی است. یک تصویر اندیس‌شده بسته به مقادیر ماتریس نقشه‌رنگ، ممکن است رنگی یا سطح خاکستری باشد.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۲- انواع تصاویر-ادامه

تصاویر اندیس شده (Indexed Image)-ادامه

نمونه‌ای از یک تصویر اندیس شده

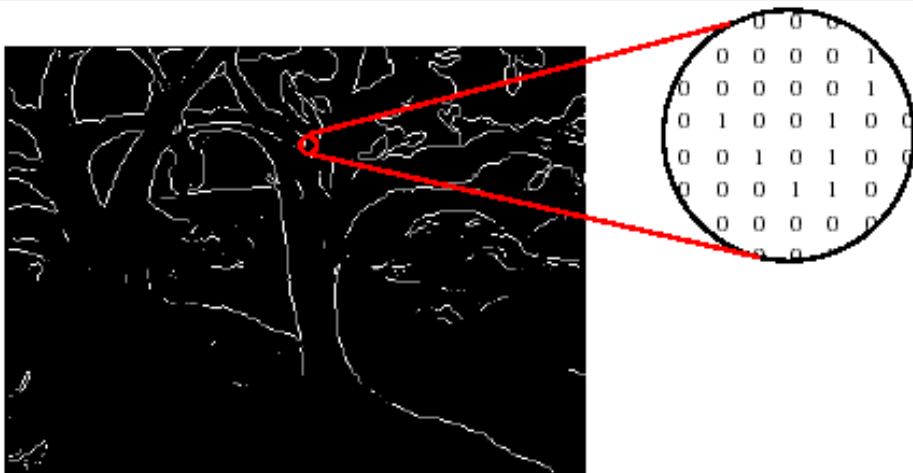


فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۲- انواع تصاویر-ادامه

تصاویر باینری

یک تصویر باینری به تصویری گفته می‌شود که هر پیکسل از آن تنها بتواند دارای یکی از دو مقدار ممکن (معمولاً 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 و ۲۵۵ باشد.



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۲- انواع تصاویر-ادامه

تصاویر RGB

یک تصویر **RGB** یا **true color** به تصویری گفته می‌شود که به ازای هر پیکسل از آن سه عدد بین ۰ تا ۲۵۵ در حافظه کامپیوتر ذخیره شده باشد که این اعداد معرف شدت هر یک از اولیه‌های قرمز، سبز و آبی می‌باشد. مثلاً 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 ۱۶ میلیون ($256 * 256 * 256$) حالت رنگی مختلف امکانپذیر خواهد بود. واضح است که 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

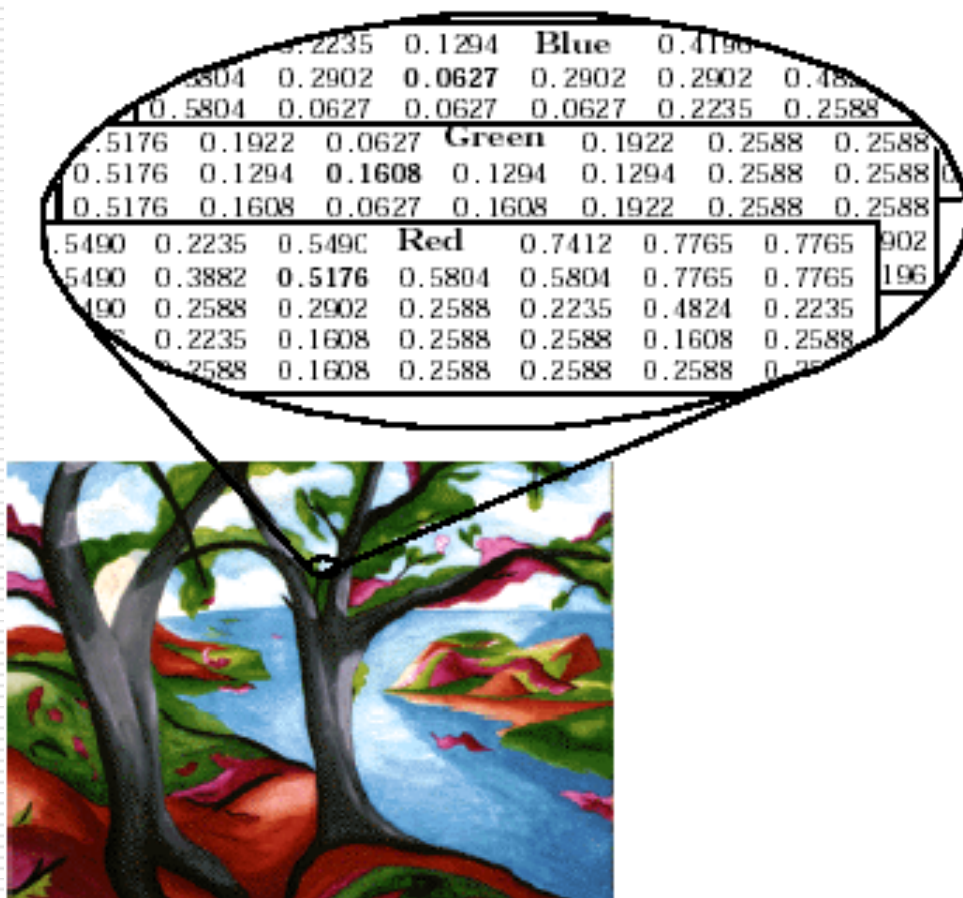
دقت شود که یک تصویر **rgb** لزوماً رنگی نیست اما می‌تواند رنگی باشد.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۲- انواع تصاویر-ادامه

تصاویر RGB-ادامه

یک تصویر **rgb** نمونه



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۳- خواندن تصاویر-تابع imread

به منظور خواندن یک فایل گرافیکی در متلب می‌توان از تابع **Imread** استفاده کرد. بسته به نوع تصویر فرمت کلی استفاده از این تابع به یکی از صورتهای زیر است:

□ برای تصاویر شدت، **rgb** و باینری: `m=imread('filename')`

□ برای تصاویر اندیس‌شده: `[m,map]=imread('filename')`

که 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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۴- نمایش تصاویر-تابع imshow

تابع **imshow** می‌تواند یک تصویر خوانده شده و یا مستقیماً یک فایل تصویری را نمایش دهد:

`imshow(m);` تصویر شدت یا **rgb**
`imshow(I , map)` تصویر اندیس شده
`imshow('filename');` فایل گرافیکی

مثال:

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

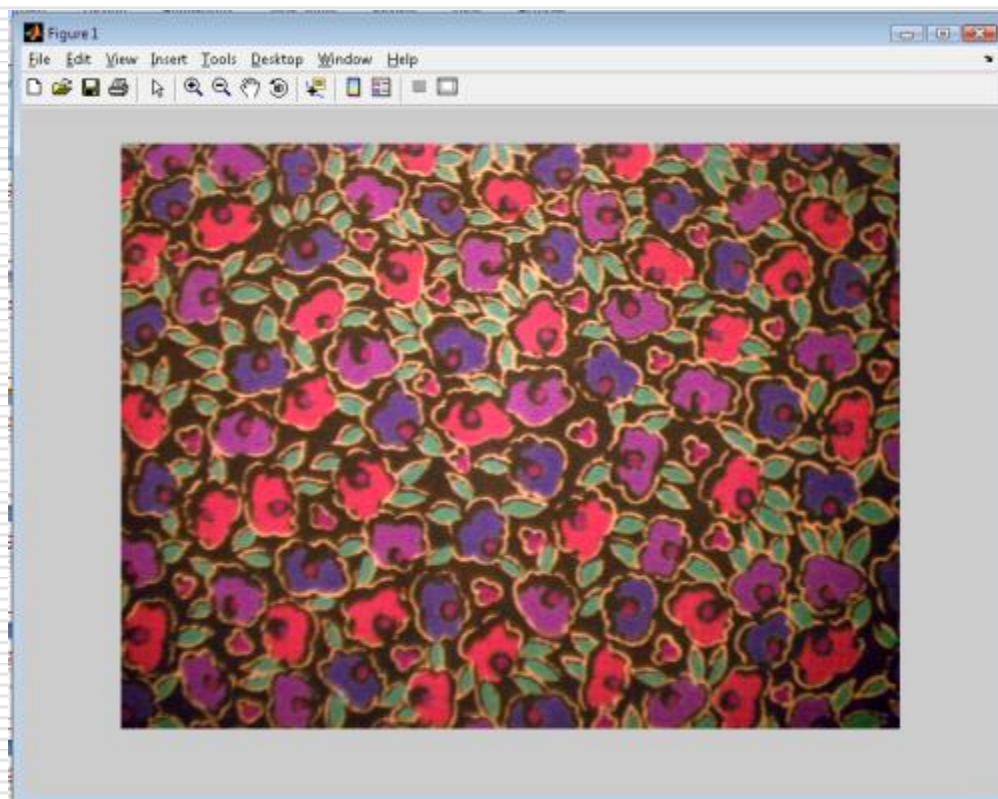
یا:

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

```
imshow(m)
```

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۴- نمایش تصاویر-تابع imshow-ادامه

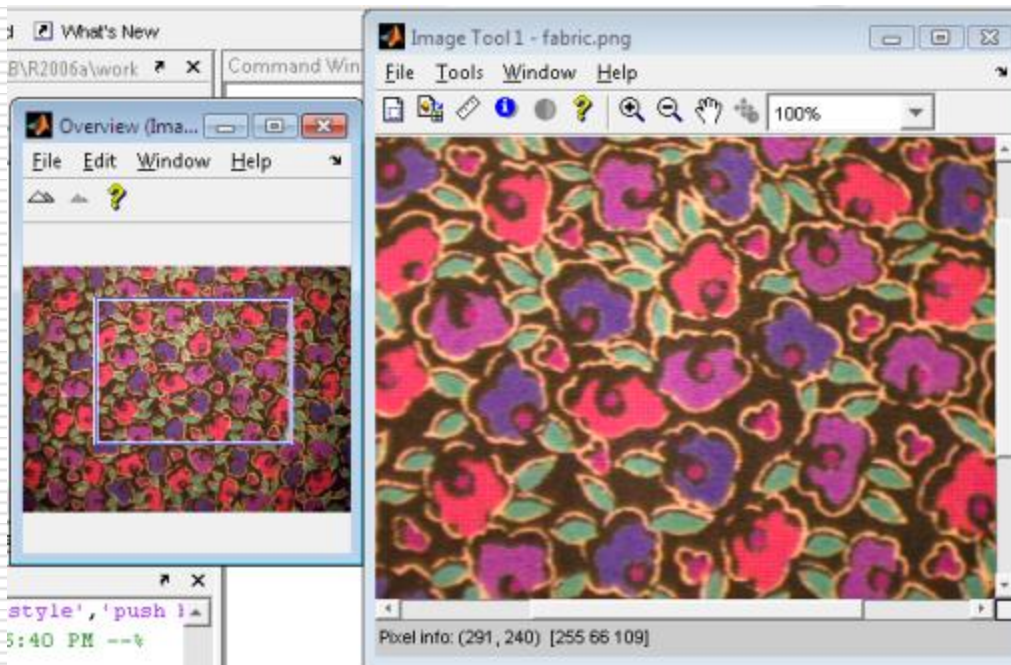


فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۴- نمایش تصاویر-تابع imshow

تابع دیگری که برای نمایش تصاویر در متلب وجود دارد تابع **imshow** است. روش استفاده از این تابع مانند تابع **imshow** است اما قابلیت‌های بیشتری را در اختیار می‌گذارد:

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



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۵- 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');
```

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۶- تعیین مشخصات یک فایل گرافیکی-تابع imfinfo

این تابع اطلاعاتی از فایل گرافیکی مانند: ابعاد تصویر، دقت ابعادی و دقت عمقی، نحوه فشرده سازی و... را ارائه می دهد. این تابع بصورت زیر بکار برده می شود:

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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۷- 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(m);
```

برای کسب اطلاعات بیشتر به راهنمای متلب مراجعه کنید.

فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۸- 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:

```
m=double(m);
```

```
m=im2uint8(m);
```

فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۹- 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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۹- عملیات هندسی بر روی تصاویر-ادامه

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]);`

در حالت اول متغیر `a` نسبت تغییر در ابعاد تصویر است. مثلاً اگر برابر با ۲ باشد یعنی ابعاد تصویر دو برابر 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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۹- عملیات هندسی بر روی تصاویر-ادامه

چرخش تصویر-تابع `imrotate`

نحوه استفاده از این تابع بصورت زیر است:

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

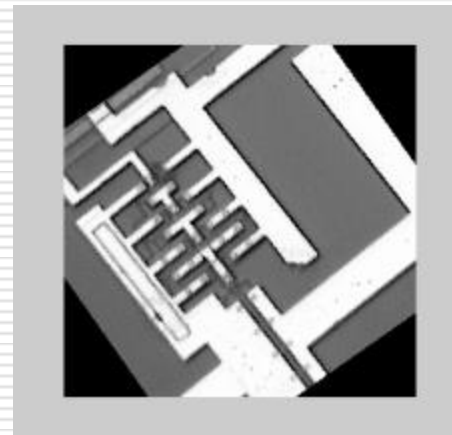
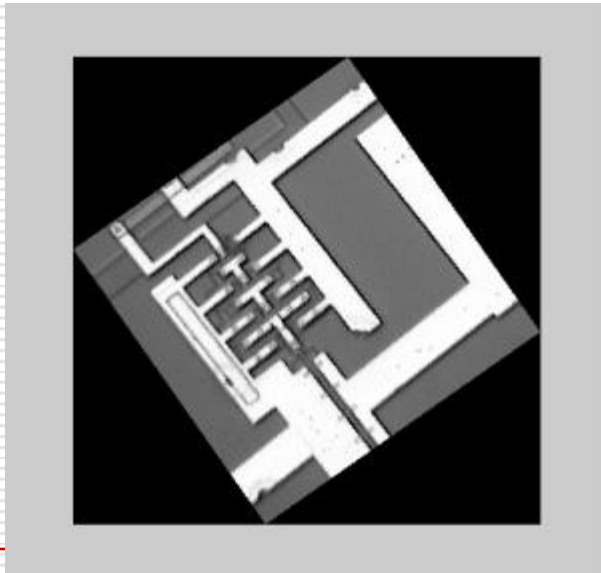
آرگومان دوم میزان چرخش تصویر بر حسب درجه می باشد. آرگومان سوم اختیاری بوده و می تواند یکی از مقادیر `bilinear, nearest` یا `bicubic` باشد. در صورتیکه این آرگومان بکار برده نشود، مقدار پیش فرض `nearest` خواهد بود. آرگومان چهارم نیز اختیاری می باشد و تنها می تواند مقدار `'crop'` را داشته باشد. در صورتیکه بکار برده شود، ابعاد تصویر پس از چرخش تغییر نمی کند اما بخشی از تصویر برش داده و حذف می شود.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۹- عملیات هندسی بر روی تصاویر-ادامه

چرخش تصویر-تابع `imrotate`-ادامه
مثال:

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



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۹- عملیات هندسی بر روی تصاویر-ادامه

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(...)
```

که در این روابط `rect` یک بردار سطری است که مختصات یک ناحیه مستطیلی شکل که از تصویر برش داده می شود را مشخص می کند. در صورتیکه این آرگومان در ورودی مشخص نشود، تصویر نمایش داده شده و متلب منتظر می ماند تا کاربر یک ناحیه مستطیلی را با ماوس انتخاب کند.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۹- عملیات هندسی بر روی تصاویر-ادامه

برش تصویر: تابع `imcrop`-ادامه
مثال:

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



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۰-۱۳- فیلترهای خطی و طراحی فیلتر

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

$m2 = \text{filter2}(h, m)$

در رابطه h ماتریس فیلتر و m ماتریس تصویر اولیه است. h می‌تواند هر ماتریس با ابعاد دلخواه باشد، اما معمولاً یک ماتریس 3×3 یا 5×5 است.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۰- فیلترهای خطی و طراحی فیلتر-ادامه

□ فیلترهای آماده

با استفاده از تابع **fspecial** می توان فیلترهای معمول در پردازش تصویر را برای استفاده با تابع **filter2** ایجاد کرد. روش استفاده از این تابع بصورت زیر است:

$h = \text{fspecial}(\text{'نام فیلتر' , ابعاد فیلتر})$

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

نام فیلتر می تواند یکی از پارامترهای زیر باشد:

gaussian: پایین گذر

sobel: بالا گذر

prewitt: بالا گذر

laplacian: فیلتر لاپلاس

log: اعمال فیلتر گوسی و پس از آن لاپلاس

average: فیلتر میانگین

unsharp: پایین گذر

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۰- فیلترهای خطی و طراحی فیلتر-ادامه

□ فیلترهای آماده-مثال

```
SobelFilter=fspecial('sobel');  
[I,map]=imread('kids.tif');I=ind2gray(I,map);  
I2=filter2(SobelFilter,I);  
imshow( I ); figure; imshow( I2 );
```



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۱-۱۳- 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 واضح تر شدن جزییات تصویر و حذف نویز بمنظور آماده سازی برای عملیات پردازشی بعدی

که در ادامه به هر یک خواهیم پرداخت

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

□ بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها

تابع `pixval` و `impixel`

با استفاده از تابع `impixel` می توان مشخصات رنگی پیکسلهایی از تصویر را بدست آورد. این تابع بصورتهای زیر بکار می رود:

`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 می دهد. پس از زدن یک کلید یا دکمه سمت راست ماوس، مشخصات این نقاط در ماتریس `P` ذخیره خواهد شد. البته این تابع بصورتهای دیگری نیز می توان بکار برد که برای کسب اطلاعات بیشتر می توانید به راهنمای متلب مراجعه کنید.

تابع `pixval` به پایین پنجره تصویر کادری را اضافه می کند که با حرکت ماوس بر روی تصویر مشخصات رنگی نقاط تصویر در این کادر نمایش داده می شود. این تابع باید پس نمایش تصویر با تابع `imshow` صدا زده شود.

فصل سیزدهم: جعبه ابزار پردازش تصویر

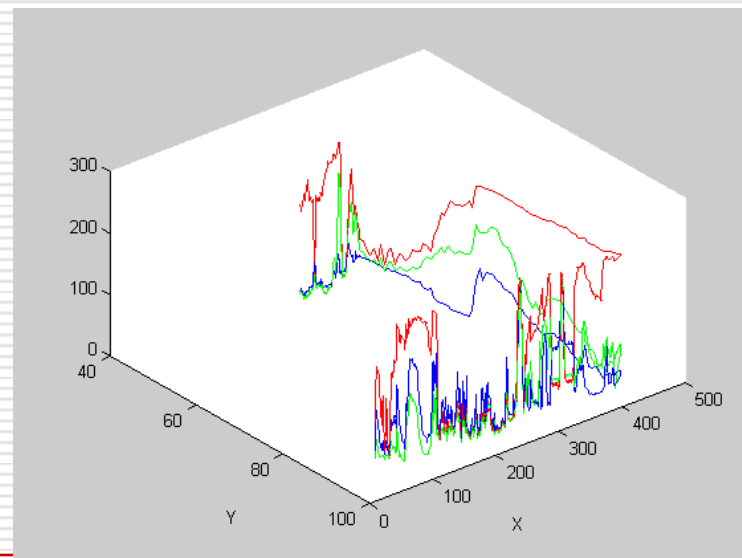
۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها-ادامه

تابع `improfile`:

این تابع نمودار تغییرات رنگ تصویر را در یک مسیر دلخواه که با ماوس انتخاب می شود رسم می کند:
مثال:

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



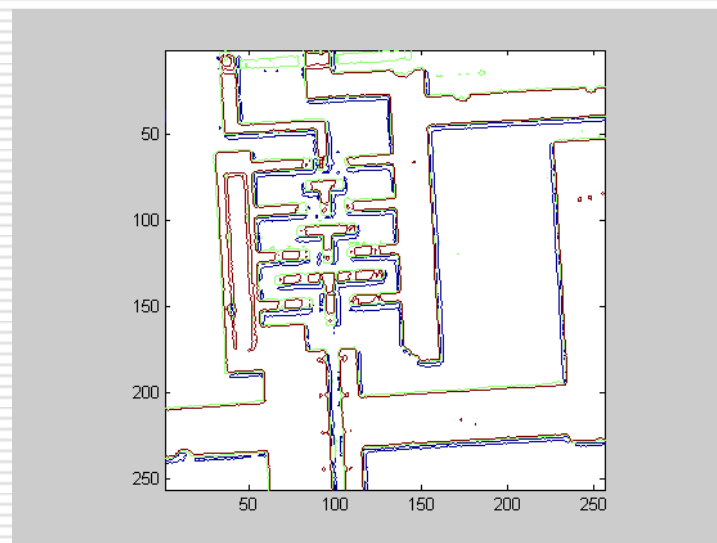
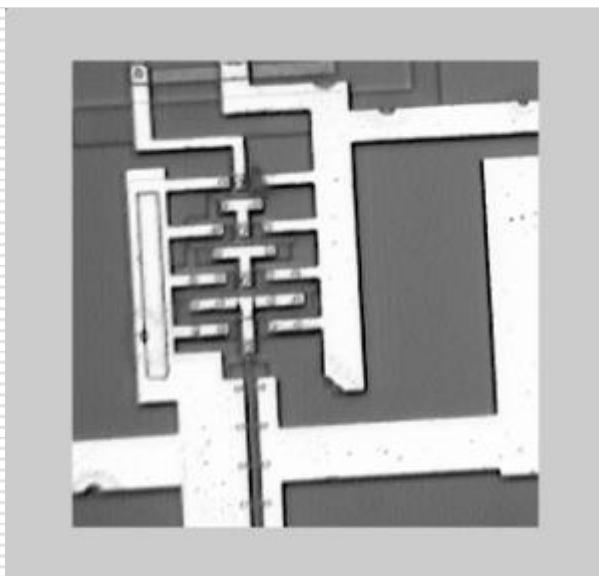
فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها-ادامه

تابع `imcontour`: رسم نمودار تراز داده‌های تصویر:

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



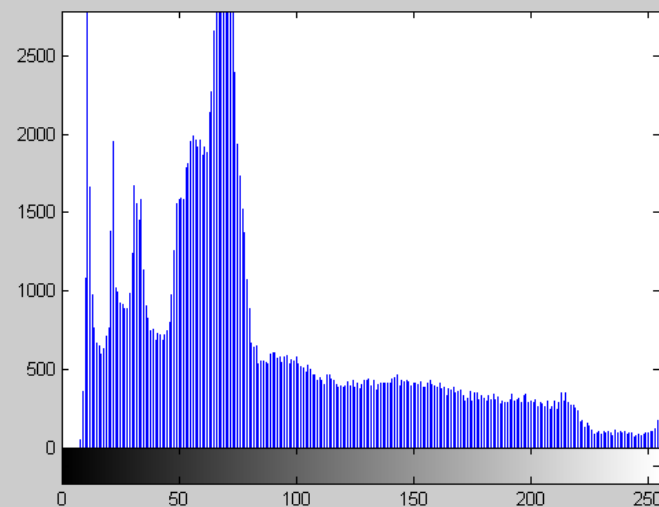
فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها-ادامه

تابع `imhist`: رسم نمودار فراوانی نقاط تصویر:

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



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بدست آوردن ارزش نقاط تصویر و اعمال عملیات آماری بر روی آنها-ادامه
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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

آنالیز تصویر:

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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر:

این عملیات که به عملیات پیش پردازش نیز مشهور است معمولاً پیش از عملیات پردازش اصلی یا عملیات آنالیز تصویر انجام می گیرد. در این عملیات بهبودهایی بر روی داده های تصویر اعمال می شود تا امکان استخراج دقیقتر و صحیح تر اطلاعات میسر گردد. این عملیات در سه بخش زیر شرح داده خواهد شد:

■ تنظیم شدت

■ متعادل کردن هیستوگرام یا بهسازی تباين

■ حذف نویز

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱-آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

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 , high] , [bottom , top])`

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:

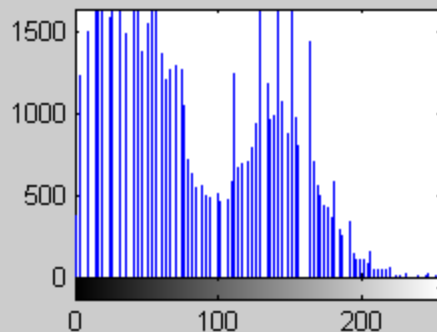
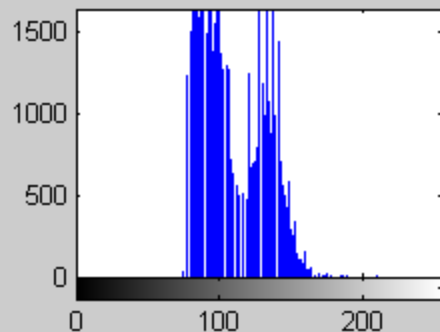
```
I=imread('pout.tif');
J=imadjust(I , [0.3 , 0.7] , [0 ,1]);
subplot(2,2,1);imshow(I); subplot(2,2,2);imshow(J);
subplot(2,2,3); imhist(I); subplot(2,2,4); imhist(J)
```

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

تنظیم شدت-تابع imadjust-ادامه



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

متعادل کردن هیستوگرام یا بهسازی تباین-تابع **histeq**

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

مثال:

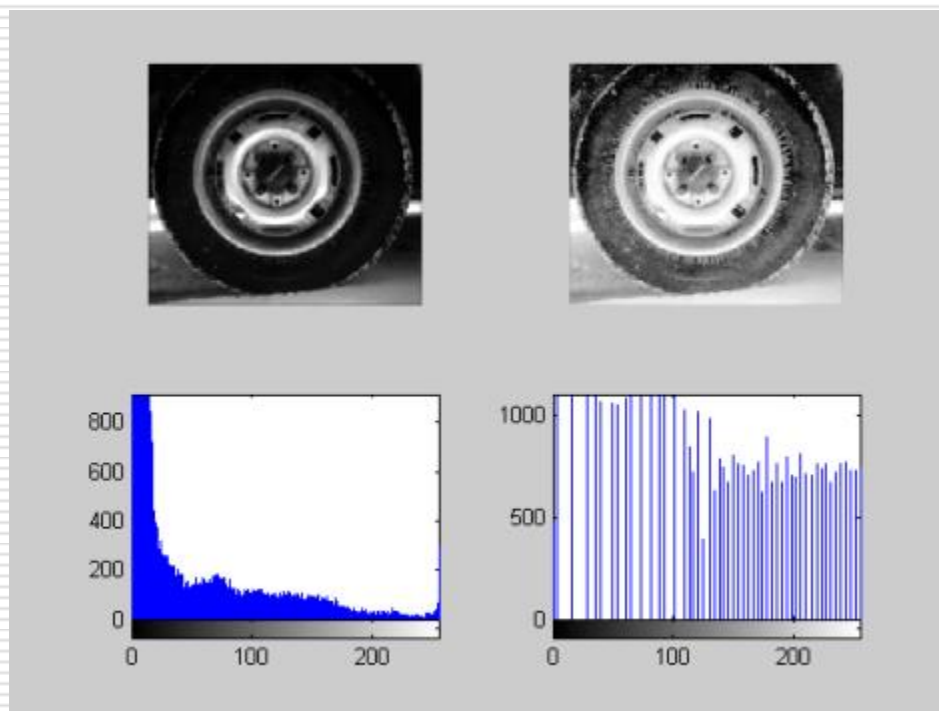
```
I=imread('tire.tif');  
J=histeq(I);figure;  
subplot(2,2,1);imshow(I);  
subplot(2,2,2);imshow(J);  
subplot(2,2,3);imhist(I);  
subplot(2,2,4);imhist(J);
```

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

متعادل کردن هیستوگرام یا بهسازی تباین-تابع histeq-ادامه



فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

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. بطور کلی 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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

حذف نویز-مثال: مقایسه فیلتر میانه و فیلتر میانگین

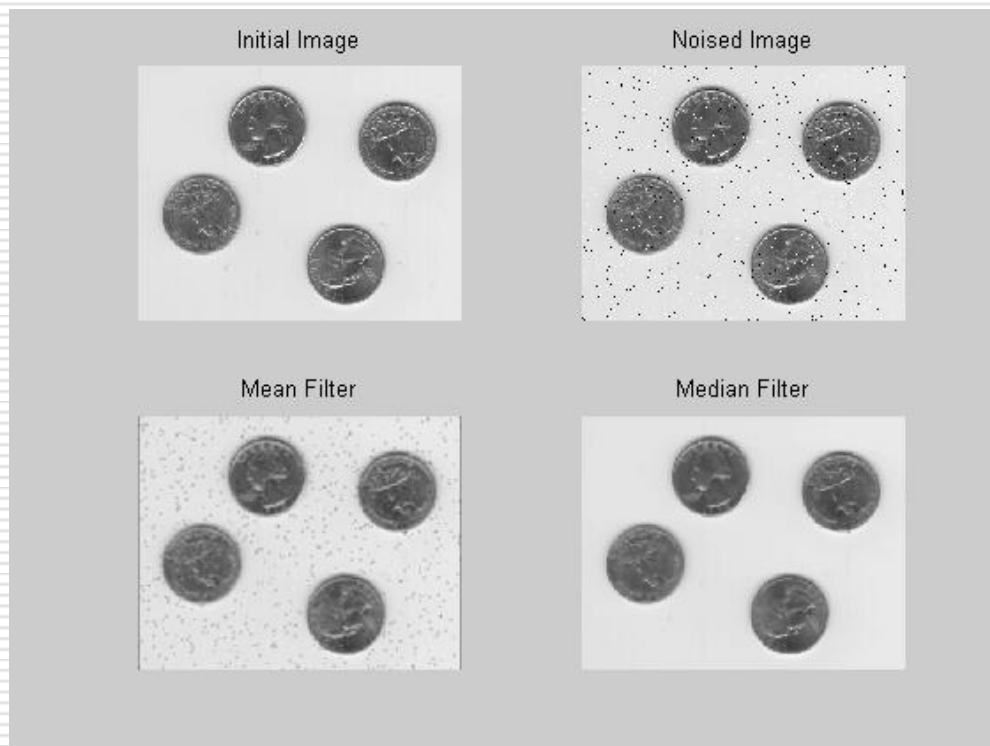
```
I = imread('eight.tif');  
J= imnoise(I , 'Salt & pepper' , 0.02); % افزودن نویز  
K= filter2(fspecial('average' , 3) , J) / 255; % فیلتر میانگین  
L=medfilt2(J , [3 , 3]); % فیلتر میانه  
subplot(2,2,1); imshow( I ); title('Initial Image')  
subplot(2,2,2); imshow( J ); title('Noised Image');  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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۱- آنالیز و بهسازی تصویر-ادامه

بهسازی تصویر-ادامه

حذف نویز-مثال: مقایسه فیلتر میانه و فیلتر میانگین-ادامه



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۲-۱۳- 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 ارائه نموده‌ایم.

همانگونه که قبلاً گفته شد تصویر باینری به تصویری گفته می‌شود که پیکسل‌های آن تنها دارای یکی از دو مقدار ممکن ۰ و ۱ یا ۰ و 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.

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۲ - عملیات بر روی تصاویر باینری

نمایش تصاویر باینری

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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۲- 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 معمولاً یک مرحله قبل از عملیات پردازش نهایی انجام میشود. منظور از عملیات پردازش نهایی عملیاتی است که در آن اطلاعاتی از تصویر استخراج میشود. مثلاً محیط یا مساحت اجزا تصویر محاسبه می‌گردد.

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

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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۲- عملیات بر روی تصاویر باینری-ادامه

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

فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۲-۱۳- عملیات بر روی تصاویر باینری-ادامه

عملیات ساختاری Morphological Operations-ادامه

عملیات افزایش و فرسایش-ادامه

ابعاد همسایگی و انتخاب همسایه‌ها توسط یک ماتریس ماسک (Mask) مشخص می‌شوند. مثلاً اگر ماتریس ماسک یک ماتریس 3×3 باشد که تمامی عناصر آن برابر با ۱ باشد. یعنی یک همسایگی 3×3 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 هر دو 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);
```

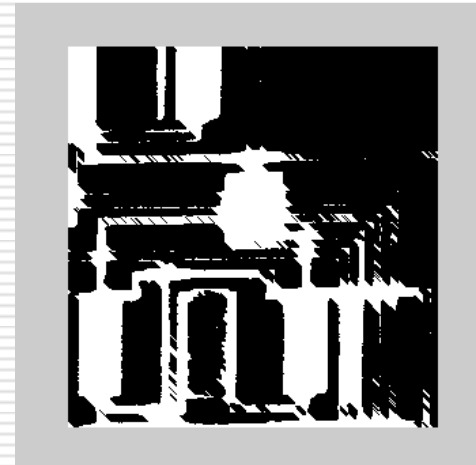
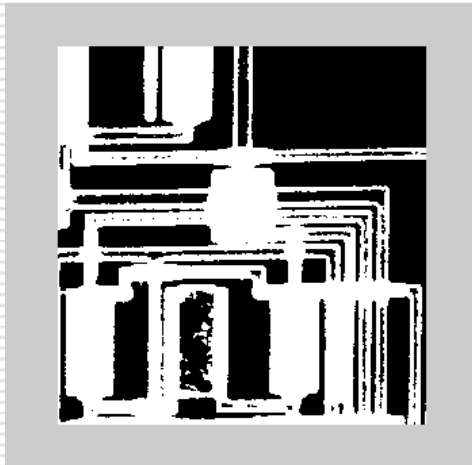
فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۲- عملیات بر روی تصاویر باینری-ادامه

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);
```



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۲-۱۳- عملیات بر روی تصاویر باینری-ادامه

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.

عملاً 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)

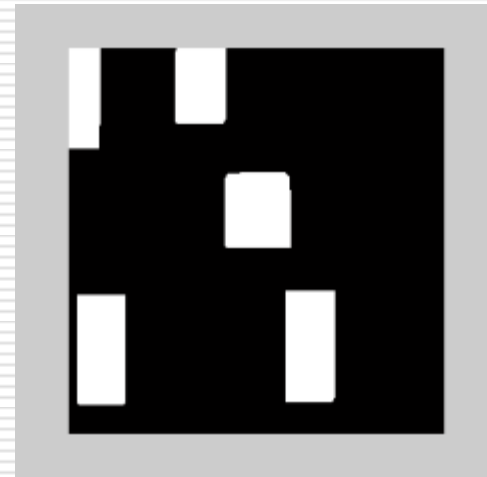
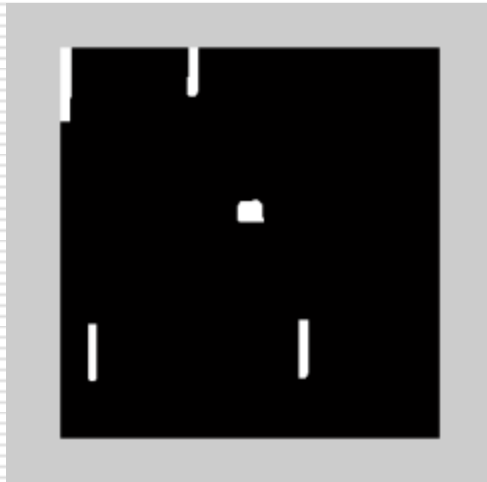
فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۲-۱۳- عملیات بر روی تصاویر باینری-ادامه

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);
```



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

۱۳-۱۲- عملیات بر روی تصاویر باینری-ادامه

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

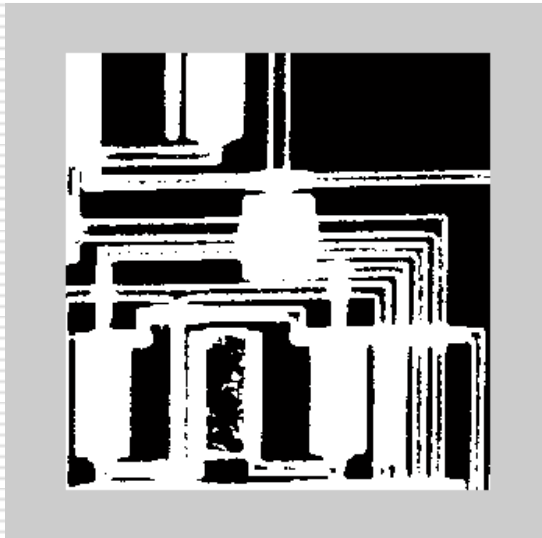
فصل سیزدهم: جعبه ابزار پردازش تصویر

۱۳-۱۲- عملیات بر روی تصاویر باینری-ادامه

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);
```



فصل سیزدهم: جعبه‌ابزار پردازش تصویر

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 جزء 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. مثلاً شرط:

$$m(:, :, 1) < 100 \text{ \& } m(:, :, 2) > 120$$
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

فصل سیزدهم: جعبه ابزار پردازش تصویر

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 می باشد برنامه ای 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 dpi در نظر بگیرید.

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