آموزش جامع
MATLAB

مقدماتی – متوسط - پیشرفته

- ❑ شروع کار با متلب
- ❑ صفحات متلب
- ❑ آرایه ها وماتریس ها
- ❑ ماتریس های سه بعدی
- ❑ توابع مقدماتی
- ❑ توابع کاراکتری
- ❑ رسم انواع نمودارها دوبعدی
- ❑ رسم انواع نمودارهای سه بعدی
- ❑ توابع خاص
- ❑ محاسبات سمبولیک

حد - مشتق – انتگرال – سری ها

معادلات خطی و غیر خطی

آنالیز برداری - معادلات دیفرانسیل

تبدیل لاپلاس – فوریه –z

- ❑ معادلات خطی و غیر خطی
- ❑ بهینه سازی
- ❑ معادلات مشتقات جزئی
- ❑ سیگنال ها و سیستم ها
- ❑ کنترل

- ❑ سیمولینک

SIMULINK

- ❑ رابط گرافیکی کاربر

GUI

مهندس موسوی

باسمه تعالی

اولین ویرایش آموزش جامع متلب در اختیار علاقمندان قرار می گیرد. این کتاب حاصل چندین سال تجربه تدریس و برنامه نویسی با متلب می باشد و امید است برای آموزش این نرم افزار مفید، سودمند باشد. سعی شده است با رعایت پوستگی مطالب و ضرورت آشنایی با آنها برای کار با متلب و همچنین ارائه مثالهای مرتبط کتاب بگونه ای باشد که بصورت خودآموز مورد استفاده قرار گیرد

این کتاب الکترونیکی هفت فصل اول کتاب آموزش جامع متلب می باشدکه عموما مطالب پرکاربردی هستند

بی شک این کتاب مانند هر مجموعه دیگری، بی نیاز از انتقاد و پیشنهاد نیست انتقادات و پیشنهادات خود می توانید با ما با آدرس پست الکترونیک musavi_sn@yahoo.com در میان بگذارید

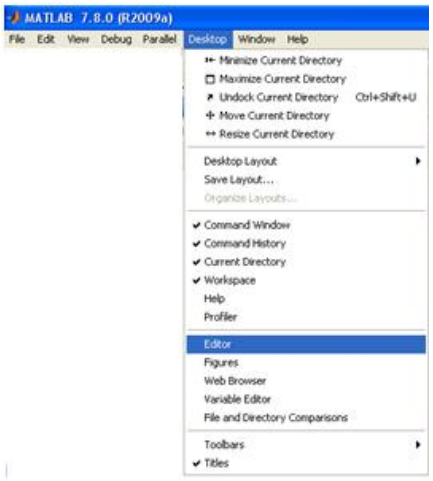
موسوی ۳ بهمن ۹۱

فصل ۱

۱ آشنایی با صفحات مختلف MATLAB

۱.۱ مقدمه

متلب صفحه‌های زیادی دارد که هر یک کاربردهای خاص خود را دارند مکان این صفحات با سلیقه‌ی کاربر تنظیم می‌شود اگر متلب را اجرا کنیم ممکن است همه‌ی صفحات متلب فعال نباشند با تیک زدن در جلوی عناوین صفحات می‌توانیم آنها را فعال کنیم



۱.۲ تنظیم صفحات متلب

اندازه‌ی صفحه‌های متلب را می‌توانیم تغییر دهیم. می‌توانیم صفحات را بصورت جدا از بدنه‌ی اصلی متلب مشاهده کنیم و یا تنظیمات دیگر را بر روی آنها انجام دهیم شکلهای زیر تعدادی از این تنظیمها برای صفحات متلب را نشان می‌دهد

با کلیک بر روی یک صفحه رنگ نوار عنوان تغییر می‌کند و صفحه‌ی مربوط به آن آماده‌ی استفاده کاربر و یا اعمال تغییرات در آن می‌شود برای مثال دو شکل زیر تغییرات صفحه فرمان را قبل و بعد از کلیک در این صفحه نشان می‌دهند

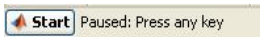
- ☑ با کلیک بر روی می‌توانیم یک صفحه را از بدنه‌ی اصلی متلب جدا کنیم در این صورت صفحه‌ی مورد نظر بصورت یک صفحه‌ی مجزا نشان داده می‌شود و آیکن به آیکن تغییر شکل می‌دهد که با کلیک بر روی آن می‌توانیم دوباره صفحه‌ی یاد شده را به بدنه‌ی اصلی متلب الصاق کنیم
- ☑ با کلیک بر روی آیکن می‌توانیم یک صفحه را بصورت تمام‌نما مشاهده کنیم
- ☑ با کلیک بر روی ویا صفحه به حاشیه‌ی راست و یا چپ رانده می‌شود برای بازگشت به حالت قبلی می‌توانیم پس از انتخاب صفحه از روی عنوان آن بر روی کلیک کنیم.
- ☑ با کلیک بر روی عنوان یک صفحه و نگه داشتن ماوس و جابجایی آن(برداشتن صفحه) می‌توانیم مکان صفحات متلب را جابجا کنیم کادر سیاه‌رنگ نشاندهنده‌ی محل جدید پیشنهادی برای صفحه‌ی انتخاب شده است
- ☑ در صورتی که چند صفحه روی‌هم افتاده باشند با کلیک بر روی نام صفحه‌ی مورد نظر می‌توانیم آن صفحه را فعال کنیم

- ☑ با نگه‌داشتن ماوس بر روی خط فاصل و یا محل تقاطع چند صفحه می‌توانیم اندازه‌ی صفحات را تغییر دهیم
- ☑ عبارت کنار گزینه‌ی start متلب جملاتی توضیحی را به کاربر اعلام می‌کند:

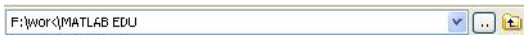
:که بمعنای آن است که متلب در حال بارگذاری است

متلب آماده‌ی اجرای برنامه‌ها و دستورها است

متلب در حال پردازش می‌باشد



متلب برای ادامه‌ی پردازش منتظر است تا کاربر از صفحه کلید،کلیدی را بفشارد (این عبارت معمولا بعد از اجرای دستور pause() ظاهر می‌گردد)

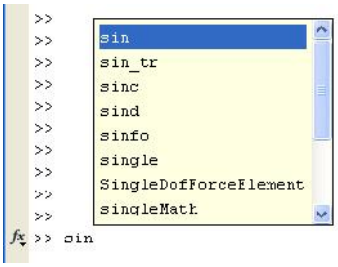
- ☑ از قسمت 

می‌توان مسیر جاری متلب را تغییر داد.

۱.۳ صفحه‌ی Command Window

صفحه فرمان متلب (Command Window) صفحه‌ای است که تقریبا همواره با آن سروکار داریم می‌توانیم کدهای خود را در آن بنویسیم این صفحه بیشتر برای مشاهده‌ی نتایج برنامه‌ها و دستورهای متلب بکار می‌رود

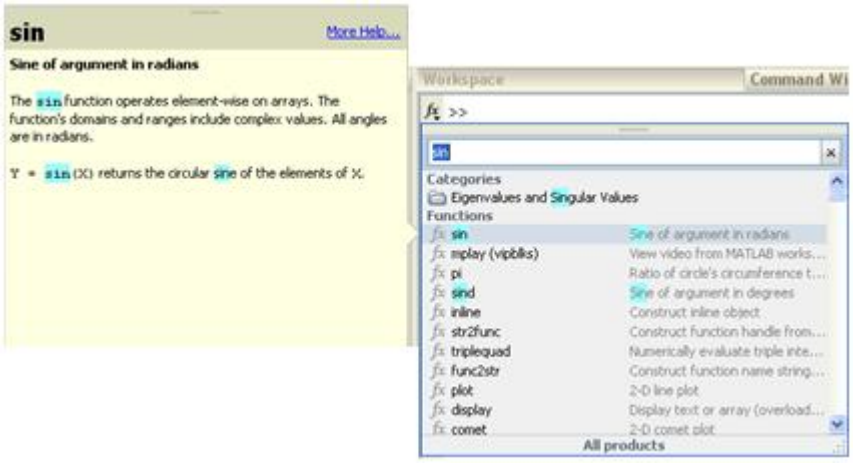
- ☑ خط فرمان متلب با علامت >> شروع می‌شود
- ☑ clc موجب پاک شدن صفحه فرمان می‌گردد
- ☑ اگر در ابتدای خط فرمان جاری کلید بالاپیمای ↑ صفحه کلید را فشار دهیم به دستور قبل می‌رسیم
- ☑ اگر عبارتی را تایپ و سپس کلید بالاپیما ↑ را فشار دهیم دستور قبلی متلب که با این عبارت آغاز می‌شود در خط فرمان جاری ظاهر خواهد شد برای مثال با تایپ a= و فشردن کلید بالاپیما آخرین دستور تایپ شده‌ای که با a= شروع می‌شود در خط فرمان ظاهر می-گردد و یا با تایپ sort و سپس فشردن کلید بالاپیما آخرین دستوری که در صفحه فرمان متلب با عبارت sort شروع می‌شود در این سطر ظاهر خواهد شد
- ☑ اگر پس از نوشتن چند کاراکتر کلید Tab از صفحه کلید را فشار دهیم متلب لیست جامعی از دستوراتی که با این عبارت شروع می‌شوند را برای ما نشان می‌دهد بعنوان مثال عبارت sin >> را در صفحه فرمان تایپ کرده و کلید Tab را فشار دهید مشاهده خواهید کرد که علاوه بر دستور sin متلب چند دستور دیگر را نیز متلب به ما نشان می‌دهد



- ☑ کلید Esc خط فرمان جاری را پاک می‌کند

- ☑ دستور clc موجب پاک شدن صفحه فرمان می‌شود
- ☑ دستور home کاری مشابه را انجام می‌دهد با این تفاوت که صفحه پاک نمی‌شود

  توجه داریم که پاک شدن صفحه بمعنای پاک شدن متغیرها نیست
- ☑ دستور who لیستی از نام متغیرهایی را که تعریف کرده‌ایم را به نمایش می‌گذارد
- ☑ دستور whos لیست متغیرها را با اطلاعات جامعتری نشان می‌دهد
- ☑ دستور whos a اطلاعاتی در مورد متغیر ذکر شده (در اینجا a) می‌دهد
- ☑ در آغاز خط فرمان متلب یک help خلاصه وسریع می‌باشد



علاوه بر کاربردهای فوق می‌توانیم برنامه‌ها و فایل‌های مختلف را با استفاده از دستورهای صفحه فرمان متلب فراخوانی کنیم

۱.۳.۱ استفاده از help در صفحه فرمان متلب

- ☑ `help ...` : (برای مثال help sort >>) می‌توانیم اطلاعاتی در مورد تابع ذکر شده (در اینجا sort) بدست آوریم

- ☑ `doc ...` : (برای مثال doc sin) نوشتن این عبارت موجب بازکردن اسناد متنی مربوط به تابع ذکر شده (در اینجا sin) می‌شود که البته در عبارت help sin هم پس از پایان توضیحات لینک doc sin وجود دارد. همچنین لینک توابع با کاربرد مرتبط و نزدیک با تابع سینوس نیز در انتهای فایل متنی باز شده موجود است.
- ☑ `lookfor ...` : (برای مثال lookfor sort) روشی دیگر برای جستجو (lookfor برای توابع خاص وتخصصی مفید است)

۱.۴ صفحه‌ی Editor

در این صفحه برنامه‌ها و توابع متلب نوشته می‌شوند و با پسوند m ذخیره می‌گردند

- ☑  اجرای برنامه
- ☑  بازکردن یک فایل جدید
- ☑  بازکردن پوشه‌ی مربوط به فایل جاری
- ☑  تهیه‌ی یک گزارش با فرمت Html از m فایل جاری
- ☑  ایجاد یک سلول
- ☑ نشان دادن همه‌ی سلول‌ها

- ☑  اجرای سلول انتخاب شده از برنامه
- ☑  انتخاب و اجرای سلول بعدی
- ☑  نمایش همه‌ی توابع موجود در برنامه

برای اجرای برنامه می‌توانیم از کلید F5 استفاده کنیم

برای اجرای قسمتی از برنامه، پس از انتخاب قسمت مورد نظر کلید F9 را می‌زنیم

برای ایجاد یک عبارت توضیحی از علامت % استفاده می‌کنیم (عبارت های بعد از این علامت تا آخر خط تنها ارزش توضیحی دارند)

برای ایجاد یک متن توضیحی می‌توانیم علاوه بر کمک‌گیری از روش فوق از آکولاد همراه با علامت درصد نیز استفاده کنیم بدین صورت که در ابتدای متن توضیحی(و یا قبل از متن توضیحی) از عبارت `%{` استفاده می‌کنیم و پس از پایان توضیحات عبارت `%}` را وارد می‌کنیم (دستورهای اجرایی با رنگ سبز نشان داده می‌شوند)

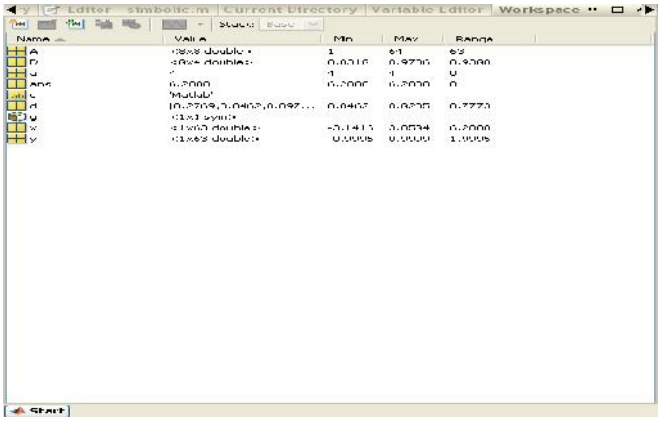
با تایپ %% و یک خط فاصله بعد از آن می‌توانیم یک سلول تولید کنیم. سلول‌ها برای خوانایی بیشتر برنامه مفید هستند. از Ctrl+Enter می‌توان برای محاسبه سلول انتخاب شده استفاده کرد

برای نوشتن ادامه برنامه‌ی خط جاری در خط بعد از ... استفاده می‌کنیم

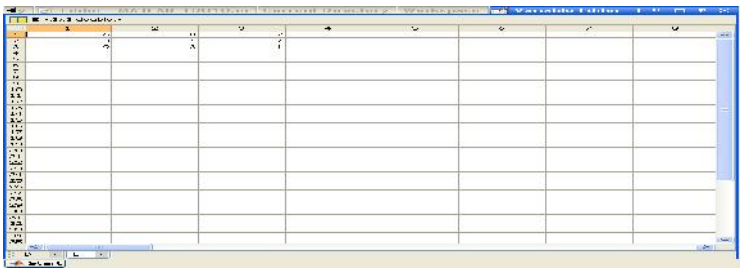
```
>> x=6;
>> y=(x+8+...
+12)/10

y =
    2.6000
```

۱.۵ صفحه‌ی Work Space

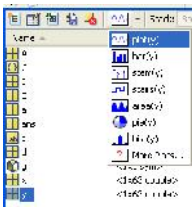
با استفاده از این صفحه می‌توانیم همه‌ی متغیرهای تعریف شده را از زمان اجرای متلب ببینیم (متغیرهایی که پاک نشده باشند) در این صفحه نام، سایز و اطلاعاتی دیگر در مورد متغیرها آمده است



- ✓ وقتی یک فایل را فراخوانی می‌کنیم و یا بین قسمت‌های مختلف متلب ارتباط برقرار می-کنیم (برای مثال بین simulinkو m فایل) داده‌ها همه در Work Space ذخیره خواهند شد
- ✓ با دوبار کلیک پیاپی روی یک آیکن زرد یک متغیر صفحه‌ی Variable Editor برای آن متغیر باز می‌شود در این صفحه براحتی می‌توان انواع متغیرها را ویرایش نمود



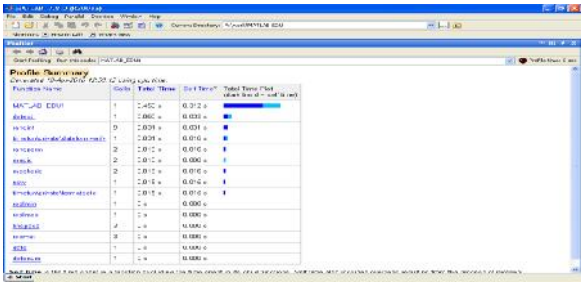
☑ با انتخاب یک متغیر و استفاده از  می‌توان انواع رسم ها را برای متغیر مورد نظر انجام داد



با راست کلیک روی یک یا چند متغیر انتخاب شده می‌توان انواع رسم‌ها را انجام داد

Profiler ۱.۶

یک ابزار بسیار ارزشمند برای بررسی عملکرد دستورها از لحاظ زمانی است. و برای آنالیز سرعت الگوریتم‌ها و روش‌ها استفاده می‌شود با کلیک بر روی  و یا تایپ عبارت profile viewer (و یا روش‌های دیگر) می‌توانید به این صفحه دسترسی داشته باشید با واردکردن نام فایل موردنظر برای تحلیل متلب یک تحلیل کامل از سرعت برنامه و سرعت تک‌تک خطوط برنامه به شما ارائه خواهد کرد



Help ۱.۷

یکی از مهمترین صفحات متلب help متلب است

- ☑ با تایپ یک عبارت در کادر جستجوی متلب  می‌توانیم توابع و یا صفحاتی را بعنوان نتیجه داشته باشیم

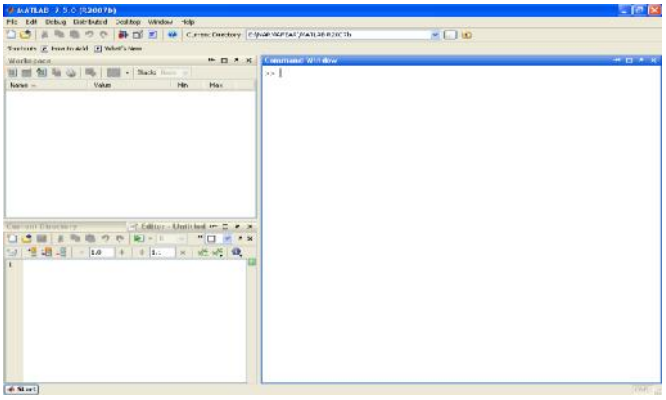
فصل 2

۲ آرایه ها

۲.۱ صفحه فرمان متلب

برای شروع کار با متلب مقداری آشنایی اولیه با صفحه فرمان لازم است در یک فصل جداگانه با صفحات متلب بصورت کامل آشنا خواهیم شد. در اینجا در حد مفدماتی و ساده با صفحه فرمان متلب آشنا می‌شویم و فعلا همه‌ی کارهایمان را در صفحه فرمان انجام می‌دهیم تا کم‌کم بصورت کاملتر با متلب و صفحات مختلف آن آشنا شویم

پس از بارگذاری متلب صفحه‌ی زیر برای کاربر نمایش داده می‌شود



صفحه‌ای که پیش‌روی شماست ممکن است مقداری با پیکربندی فوق تفاوت داشته باشد .قسمتی که بر روی نوار عنوان آن عبارت Command Window نوشته شده است صفحه فرمان متلب یا همان command window نام دارد. اگر متلب شما این صفحه را پس از بارگذاری نداشت آن را از نوار ابزار گزینه‌ی desktop را انتخاب کنید و سپس عبارت command window را تیک‌دار کنید

با در دست داشتن صفحه فرمان اکنون می‌توانیم برنامه نویسی در متلب را آغاز کنیم

۲.۲ تعریف متغیرها

در متلب تعریف و مقدار دهی متغیرها عموما بصورت همزمان صورت می‌گیرد.

نکات قابل توجه در نامگذاری متغیرها :

- ☑ متلب به حروف بزرگ و یا کوچک حساس است بدین معنی که برای مثال a وA دو متغیر متفاوتند
- ☑ نام متغیر ها با حرف شروع می‌شوند a1 یک نامگذاری صحیح و 2a یک نامگذاری غلط می‌باشد
- ☑ نام متغیرها شامل حروف اعداد و زیرخط (آندرلاین) می‌باشد a_w23 صحیح و w%2 ویا d.2 غلط هستند .
- ☑ بهتر است نام متغیرها را از نام توابع پرکاربرد، پارامترها تعریف شده‌ی متلب و کلمات کلیدی متلب انتخاب نکنیم. در صورت عدم رعایت این موضوع خطاهایی خواهیم داشت که بخصوص در مواردی که در متلب تازه‌کار هستیم براحتی قادر به رفع آنها نخواهیم بود. برای مثال نام متغیر را sin نگذاریم تا با تابع سینوس اشتباه نشود. نام متغیر را pi نگذاریم تا با عدد π تعریف شده در متلب جایگزین نگردد. و همینطور نام متغیر را از کلمات کلیدی متلب مثل if و یا for انتخاب نکنیم. بزودی توضیح جامعتری در این موارد داده خواهد شد

کار خود را با تعریف چند متغیر عددی ساده شروع می‌کنیم (بعد از تعریف متغیر در خط فرمان کلید Enter را فشار دهید

```
>> a=4
a =
     4
```

```
>> d=.2
d =
    0.2000

>> g=2.3658
g =
  2.3658
```

درصورتیکه در انتهای یک دستور متلب از نماد سمیکالن (;) استفاده کنیم متغیر تعریف و مقداردهی می‌شود ولی نمایش داده نمی‌شود

```
>> s=5.69;
>> f_14=36;
```

می‌توانیم چندین دستور را در یک سطر وارد کنیم بدین منظور بین دستورها علامت کاما (,) و یا سمیکالن می‌گذاریم. در صورت استفاده از کاما نتیجه دستور تایپ شده نمایش داده می‌شود ولی اگر از سمیکالن استفاده کنیم نمایش نتیجه حذف خواهد شد.

```
>> d_58=2.36;e=12.365;

>> a=2,b=56,c=5,
a =
     2
b =
    56
c =
     5
```

در ادامه توضیحی در مورد نمایش اعداد در متلب می‌دهیم. برای مثال عدد بیست میلیارد را در متلب تایپ کنید

```
>> 20000000000
ans =
  2.0000e+010
```

نتیجه‌را بدین صورت بخوانید 2×10^{10} . با همین نمایش نیز می‌توانیم اعداد را تعریف کنیم

```
>> 2.36e4        ⇨      2.36×10^4
ans =
       23600
```

می‌توانیم توان را بصورت منفی نیز تعریف کنیم

```
>> 6.39e-3         ⇨      6.39×10^-3
ans =
    0.0064
```

در متلب چندین پارامتر مقداردهی اولیه شده‌اندکه کاربردهای زیادی دارند برای مثال به چند نمونه‌ی مهم پرکاربرد از آنها اشاره می‌کنیم

```
>> pi              →  π عدد
ans =
    3.1416

>> i               →     √−1  عدد مختلط
ans =
        0 + 1.0000i

>> j               →     √−1  عدد مختلط
ans =
        0 + 1.0000i

>> eps             →     ϵ عدد بسیار کوچک  اپسیلون

ans =
  2.2204e-016

>> inf             →          ∞عدد مثبت بی نهایت
ans =
   Inf


>> nan             →  NaN به معنای اینکه یک عدد نیست    ( Not-a-Number)
```

```
ans =
   NaN

>> 0/0
ans =
   NaN
```

برای مثال 0/0 یک nan می‌باشد

۲.۲.۱ کلمات کلیدی متلب

تعداد کلمات کلیدی متلب زیاد نیست و این کلمات شامل if ، end، for، else، while و... می‌باشند متلب خود این کلمات را تشخیص می‌دهد و در صورتی که این کلمات توسط کاربر تایپ گردد رنگ نمایش کلمه کلیدی به رنگ آبی تغییر می‌کند. برای داشتن لیستی از کلمات کلیدی می‌توانیم دستور iskeyword را در خط فرمان متلب تایپ کنیم.

```
>> iskeyword
```

کلمات کلیدی متلب

```
ans =

    'break'
    'case'
    'catch'
    'classdef'
    'continue'
    'else'
    'elseif'
    'end'
    'for'
    'function'
    'global'
    'if'
    'otherwise'
    'parfor'
    'persistent'
    'return'
    'spmd'
    'switch'
    'try'
    'while'
```

۲.۲.۲ جملات توضیحی

برای اینکه بخواهیم توضیحاتی را در قسمتی از برنامه اضافه کنیم می‌توانیم از نماد ٪ استفاده کنیم در این صورت عبارتهای بعد از این نماد در همان سطر به رنگ سبز در آمده و غیر اجرایی خواهند بود و تنها ارزش نمایشی و توضیحی خواهند داشت

```
>> a2=2.145; % one patameter
```

۲.۳ اعداد مختلط

با تعریف عدد مختلط i وj می‌توانیم اعداد را بصورت مختلط تعریف کنیم. برای این منظور یک عدد مختلط با قسمت حقیقی ۲ و قسمت موهومی ۳ را بصورت 2+3i و یا 2+i*3 نمایش می‌دهیم توجه داریم که عبارت 2+i3 بمعنای عدد ۲ بعلاوه‌ی متغیر i3 است که می‌تواند هر مقداری داشته باشد و یا تعریف نشده باشد در حالی که 3i با توجه به قواعد نامگذاری متغیرها نمی‌تواند نام یک متغیر باشد و از این بابت مشکلی وجود ندارد خطایی که می‌تواند در اینجا رخ دهد این است که ممکن است ما i را در برنامه مقدار داده باشیم در این صورت در کار با عبارت 3i هیچ مشکلی نخواهیم داشت ولی اگر از عبارتی مثل عبارت 2+i*3 استفاده کنیم نتیجه آنچه که انتظار داریم نخواهد بود برای جلوگیری از وقوع این اشتباه بهتر است اگر از فرم a+b*i استفاده می‌کنیم بجای i وj از عبارت های معادل 1i و 1j استفاده کنیم یعنی: 2+3*1i

```
>> 2+3i
ans =
   2.0000 + 3.0000i

>> .45+3.8i
ans =
   0.4500 + 3.8000i

>> 4+3*i
ans =
```

```
   4.0000 + 3.0000i

>> 4+i3
??? Undefined function or variable 'i3'.
→     i3 تعریف نشده است (تابع یا متعیر)

>> i=10;4+3*i
ans =
    34

>> i=10;4+3*1i
ans =
.0000i
```

۲.۴ آرایه ها

نکات:

- ☑ آغاز آرایه با علامت براکت باز [
- ☑ فاصله‌ی عناصر با , و یا فاصله
- ☑ پایان آرایه با علامت براکت بسته]

در این صورت یک ماتریس سطری خواهیم داشت که به طور معادل آن را بردار و یا آرایه‌ی عددی (آرایه) می‌نامیم، برای آشنایی بیشتر مثال‌های زیر را دنبال کنید

```
>> a=[2,5,8]
a =
     2     5     8

>> b=[2  -9  1.36  5.2   6.9]
b =
    2.0000   -9.0000    1.3600    5.2000    6.9000

>> d=[]      →  ماتریس تهی
d =
```

```
     []
```

آرایه‌ها را می‌توانیم بصورت عنصر به عنصر نیز تعریف کنیم این موضوع را با ذکر یک مثال توضیح می‌دهیم

```
>> c(1)=5;c(2)=8;c(3)=10;
c =
   5     8    10
```

- ☑ اندیس آرایه‌ها در متلب مخالف صفر بوده و صحیح است یعنی اندیس صفر نداریم همچنین اندیس هایی مثل ۱.۲ و۲- و ...

- ☑ هرگاه تعدادی از عناصر ماتریس را تعریف نشده رها کنیم این عناصر خودبخود صفر تعریف می‌شوند

```
>> f(1)=5;f(2)=8;f(6)=-5;
f =
     5     8     0     0     0    -5
```

برای تعریف آرایه‌ای از اعداد با نظم خاص در متلب می‌توانیم از راههای ساده‌تری استفاده کنیم

```
>> c1=1:5              → اعداد ۱ تا ۵
c1 =
     1     2     3     4     5
```

```
>> c2=0:6              → اعداد ۰ تا ۶
c2 =
     0     1     2     3     4     5     6

>> c3= -6 : 9                → اعداد ۶- تا ۹
ans =
```

```
  Columns 1 through 11          →      ستون ۱ تا ۱۱
  -6   -5   -4   -3   -2   -1    0     1     2     3     4

  Columns 12 through 16         →        ستون ۱۲ تا ۱۶
     5     6     7     8     9
```

برای توضیحی در مورد نمایش برداری فوق به این نکته توجه داریم که نمی‌توان تعداد زیادی عدد را در یک سطر نشان داد و در این مورد در متلب با توجه به اندازه‌ی صفحه فرمان محدودیت داریم. از آنجا که بردار یک ماتریس سطری است لذا هر عدد در بردار را می‌توان با ذکر شماره ستون آن مشخص کرد.

همچنین می‌توانیم گام تغییر اعداد را یک عدد دلخواه تعیین کنیم حالت کلی این تعریف بدین صورت است:

n:mکه بمعنای اعداد از n تا m است حالت دیگر تعریف n:d:m می‌باشد که d گام تعریف را نشان می‌دهد با مثال‌های زیر موضوع کاملا روشن می‌شود. در ضمن برای این تعریف می‌توان پرانتز() و یا علامت [] را استفاده کرد و یا بدون این دو دستور را نوشت.[۱] $n:m \equiv (n:m) \equiv [n:m]$

```
>> d1=0: 2 : 10    →              اعداد صفر تا ده با گام دو
d1 =
     0     2     4     6     8    10
```

```
>> d2=2: .2 :3     →              اعداد ۲ تا ۳ با گام ۲.
d2 =
  2.0000    2.2000    2.4000    2.6000    2.8000    3.0000
```

[۱] نماد≡را بخوانیم: معادل است با

```
>>d3=0:-1:-3        →          اعداد صفر تا ۳-
d3 =
     0     -1     -2     -3
```

برای حالتی که عدد دوم کوچکتر از عدد اول می‌باشد گام تغییر حتما باید منفی ذکر گردد برای مثال نتیجه‌ی عبارت 0:-3 ماتریس تهی خواهد بود

```
>> 0:-3
ans =
   Empty matrix: 1-by-0   →  ماتریس تهی ۱ در صفر
```

ذکر نشدن گام نیز بمعنای گام ۱ می‌باشد یعنی برای نمونه دو دستور ۱:۳ و ۱:۱:۳ معادل هستند

- `linspace()`: اگر بخواهیم بازه‌ای دلخواه را بصورت خطی به چندین قسمت مساوی تقسیم کنیم از این تابع استفاده می‌کنیم و فرم فراخوانی آن بصورت linspace(a,b,N) می‌باشد که در نتیجه آن یک بردار سطری با عنصر اول a و عنصر آخر b ،فاصله عناصر یکسان و طول بردار N خواهیم داشت.

```
>> linspace(4,9,6)
ans =
     4     5     6     7     8     9

>> linspace(1,2,6)
ans =
   1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

۲.۵ آرایه‌های خاص

- `zeros(1,n)` (آرایه‌ی تمام صفر) برای ساختن برداری با عناصر تمام صفر می‌توانیم از دستور zeros استفاده کنیم بدین صورت که نتیجه‌ی zeros(1,n) یک آرایه‌ی n تایی (۱×n) تمام صفر می‌دهد

```
>> zeros(1,5)
ans =
     0     0     0     0     0

>> zeros(1,3)
ans =
     0     0     0
```

- `ones(1,n)` : (آرایه‌ی تمام یک) برای ساختن برداری با همه‌ی عناصر برابر یک می‌توانیم از دستور ones استفاده کنیم بدین صورت که ones(1,n) یک آرایه‌ی n تایی (۱×n) صفر می‌دهد

```
>> ones(1,6)
ans =
     1     1     1     1     1     1
```

آرایه‌ای ۶×۱ با تمام عناصر ۵ بسازید

```
>> 5*ones(1,6)
ans =
     5     5     5     5     5     5
```

این توابع برای ساختن آرایه‌های خاص کاربرد زیادی دارند.

۲.۶ اعداد و آرایه‌ی تصادفی:

در متلب بمنظور تولید اعداد ، آرایه‌ها و ماتریسهای تصادفی توابع مختلفی در نظر گرفته شده است که هر یک از این توابع بروش متفاوت این اعداد را تولید می‌کنند و کاربردهای خاص خود را دارند در زیر چند نمونه از این توابع را معرفی می‌کنیم:

- `rand(1,n)` : این تابع بمنظور تولید اعداد تصادفی با توزیع یکنواخت در بازه‌ی [0 1] استفاده می‌شود

تابع rand را در اینجا بصورت rand(1,n) فراخوانی می‌کنیم حاصل یک آرایه‌ی ۱×n است که هر عنصرآرایه عددی تصادفی بین صفر و یک است برای مثال:

```
a = rand(1,4)

    0.8147    0.9058    0.1270    0.9134
```

هر عنصرآرایه عددی تصادفی بین صفر و یک است

برای اینکه اعداد تصادفی در بازه بزرگتری توزیع شوند می‌توانیم تابع rand() را در عددی ضرب کنیم برای مثال 5* rand(1,4) در بازه‌ی [0 5] قرار دارد

```
>> a=8*rand(1,3)
a =
    6.4022    1.1351    3.3741
```

- ☑ در حالت کلی اگر u یک عدد تصادفی در بازه‌ی [0 1] باشد a+(b-a)*u در بازه‌ی [a b] قرار دارد

```
>> u=rand(1,6);
```

```
>>u1=2+(5-2)*u            → u1∈[a  b] a=2 b=5
u1 =

   4.4704    4.0845    2.9513    4.8507    2.1033    3.3162
```

عناصر u1 تصادفی و در بازه‌ی [2 5] قرار دارند

- `randn(1,n)`: این تابع بمنظور تولید اعداد تصادفی با توزیع نرمال(میانگین صفر و واریانس یک) استفاده می‌شود این تابع را مشابه تابع rand() بصورت randn(1,n) فراخوانی می‌کنیم حاصل یک آرایه‌ی n×۱ است که هر عنصرآرایه عددی تصادفی با توزیع یاد شده می‌باشد (این تابع عموما در شبیه سازی نویز گاوسی کاربرد دارد)

```
>> a=randn(1,4)
a =
   -0.2414    0.3192    0.3129   -0.8649
```

ملاحظه می‌کنیم که اعداد منفی نیز در این فراخوانی ظاهر می‌شوند. با روشی مشابه روشی که برای rand() گفته شد می‌توانیم واریانس و میانگین توزیع را تغییر دهیم.

- تابع randint() : ازاین تابع برای تولید اعداد تصادفی صحیح استفاده می‌کنیم و به چندین فرم می‌توانیم این تابع را فراخوانی کنیم

- `randint(1,n)` : تولید رشته‌ی تصادفی متشکل از عناصر ۰و۱با طول n (در اینجا فراخوانی شبیه rand() است)

```
>> a=randint(1,5)
a =
     1     0     1     0     1
```

- `randint(1,n,m)` :تولید اعداد تصادفی صحیح کمتر از m (0 1 2 … m-1): برای این منظور عدد m باید بعنوان پارامتر سوم به ورودی‌های تابع افزوده گردد بنابراین randint(1,n,m) یک بردار تصادفی با طول n و عناصر صحیح کمتر از m بدست می‌دهد. یعنی n عنصر بین 0 و m-1

```
>> a=randint(1,5,8)
a =
     0     2     1     6     2

>> a=randint(1,5,80)
a =
    73    12    66    43    79
```

- `randint(1,n,[a  b])` :تولید اعداد تصادفی در بازه

(a a+1 a+2 …b-1 b) <== [a b]

```
>> a=randint(1,5,[-2 5])
a =
    -2     1    -2     5    -2

>> a=randint(1,5,[20 25])
a =
    24    24    25    20    22
```

- `randperm(n)`: این تابع یک تابع تصادفی نیست بلکه برای چیدن اعداد صحیح به فرم تصادفی مورد استفاده قرار می‌گیرد برای مثال randperm(n) اعداد ۱تا n را بترتیب تصادفی می‌چیند

```
>> randperm(8)
ans =
     3     7     2     1     6     5     8     4
```

دستورهای بسیار زیاد دیگری نیز برای تولید اعداد تصادفی وجود دارند که عموما مربوط به توزیع های مختلف می‌باشند برای مثال توزیع T ویا توزیع پواسن و یا انواع دیگر توزیعها ، ولی دستورهایی که در بالا مطرح شد در مسائل و موضوعات عمومی و در بسیاری از مباحث تخصصی می‌توانند نیاز ما را برطرف کنند.

۲.۷ فراخوانی و مقدار دهی آرایه‌ها:

در کدنویسی برنامه‌های مختلف فراخوانی و مقداردهی به فراوانی مورد استفاده قرار می‌گیرد و توانایی متلب در فراخوانی، مقداردهی و انجام عملیات جبری بر روی زیرماتریس‌ها و زیر آرایه‌ها از توانمندی-های بالای متلب بشمار می‌رود

برای یک متغیر آرایه‌ای که در متلب تعریف شده است اگر نام آن متغیر را در صفحه فرمان تایپ کنیم مقادیر آن نشان داده می‌شود و یا اگر تنها عنصر خاصی از آرایه را فراخوانی کنیم تنها این عنصر نمایش داده می‌شود:

```
>>a(3)
ans =
    20
>>a(0)      →   در متلب اندیس باید صحیح باشد  و اندیس صفر نداریم
??? Attempted to access a(0); index must be a positive
integer or logical.
```

☑ در متلب اندیس همواره یک عدد صحیح است و اندیس صفر نداریم

می‌توانیم چندین مقدار را همزمان فراخوانی کنیم

```
>>a(1:2)                          →عناصر ۱ تا ۲
ans =
    10    15
```

```
>>a(1:3)
ans =
    10    15    20
```

```
>>a(2:5)
ans =
    15    20    25    30

>>a(2:2:8)
ans =
    15    25    35    45

>>a(5:-1:2)
ans =
    30    25    20    15

>>a(end)                    → a عنصر آخر
ans =
    50

>>a(end-3)
ans =
    35

>>a(3:end)                  → عنصر ۳ تا آخر
ans =
    20    25    30    35    40    45    50

>>a(end:-1:1)               → با این عبارت می‌توانیم آرایه را از آخر به اول بنویسیم
ans =

    50    45    40    35    30    25    20    15    10

>>a(:)                      نمایش ستونی a
ans =

    10
    15
    20
    25
    30
    35
    40
```

```
    45
    50
```

- ☑ end را می‌توانیم بعنوان اندیس ماتریس فراخوانی کنیم که نمایشگر آخرین عنصر آرایه می-باشد
- ☑ فرم نمایش a(:) نمایش ستونی را نتیجه می‌دهد
- ☑ می‌توانیم بجای اندیس یک بردار داشته باشیم و بدینصورت مجموعه‌ای از عناصر آرایه را فراخوانی کنیم

```
>>c=[1:5];d=[2 6 3 5];
>> a([2 4])
ans =
    15    25

>> a([1 5 2 ])
ans =
    10    30    15

>> a([1 5 2 5 8 2 4 2 2 4])
ans =
    10    30    15    30    45    15    25    15    15    25

>> a(c(1))
ans =
    10

>> a(c)
ans =
    10    15    20    25    30

>> a(d)
ans =
    15    35    20    30

>> a([c d 2])
ans =
    10    15    20    25    30    15    35    20    30    15
```

۲.۷.۱ مقداردهی آرایه ها:

پس از توانایی فراخوانی مقداردهی کار سختی نیست

```
>> a=[1 2 3 4];
>> b=[10 20 30 40];

>> a(1)=20                    →  مقدار دهی یک عنصر از آرایه
a =
    20     2     3     4

>> a(1:3)=[100 100 100]       →  مقدار دهی همزمان چند عنصر
a =
   100   100   100     4

>> a([2 5])=[77 77]
a =
   100    77   100     4    77

>> x=a
x =
   100    77   100     4    77

>> x(1)=[]    →  حذف یک عنصر
x =
    77   100     4    77

>> x=a;x(1:3)=[]     →  حذف چند عنصر  a=[100 77 100 4 77]
x =
     4    77

>> a(1:2:end)=[]      →  حذف عناصر ۱ و ۳ و ۵
a =
    77     4
```

فصل 3

۳ ماتریس ها

۳.۱ ماتریس در MATLAB

اهمیت ماتریس در متلب از نامگذاری متلب که از MAtrix LABoratory گرفته شده است روشن است . اغلب متغیرها ماتریس هستند برای مثال یک عدد معمولی یک ماتریس ۱× ۱ است آرایه‌های سطری و ستونی وضعیت مشابهی دارند یک سیگنال صحبت و یا یک عکس همه یک ماتریس هستند. یک رشته‌ی متنی یک ماتریس با عناصر کاراکتر است

۳.۱.۱ تعریف ماتریس:

در تعریف ماتریس نکات زیر حائز اهمیت‌اند

- ☑ آغاز ماتریس با علامت براکت باز [
- ☑ فاصله‌ی عناصر یک سطر با کاما و یا فاصله
- ☑ تفکیک ستون‌ها با سمی‌کالن
- ☑ پایان آرایه با علامت براکت بسته]

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

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

بنابراین آرایه‌ی سطری را نیز می‌توان بصورت زیر تعریف کرد:

```
>> A=[1;2;3;4]
A =
     1
     2
     3
     4
```

ماتریس ها را می‌توانیم بصورت عنصر به عنصر نیز تعریف کنیم این موضوع را با یک مثال توضیح می‌دهیم

```
>>A(1,1)=1;A(1,2)=2;A(2,1)=3;A(2,2)=4;

>>A     →   نمایش A
A =
     1     2
     3     4
```

هرگاه تعدادی از عناصر ماتریس را تعریف نشده رها کنیم این عناصر خودبخود صفر تعریف می‌شوند

```
>>B(1,1)=2;B(2,1)=5;B(2,4)=8;

>>B
B =
     2     0     0     0
     5     0     0     8
```

۳.۱.۲ ماتریسهای خاص

این مبحث کاملا مشابه مبحث آرایه‌های سطری است با این تفاوت که بجای عدد۱ نشانگر تعداد سطر در توابع یاد شده است (ones(),zeros(),rand(),…) عدد دلخواه m را می‌گذاریم لذا یک ماتریس m×n خواهیم داشت.

```
>> ones(3,6)
ans =
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1

>> rand(2,4)
ans =
```

```
    0.6221    0.5132    0.0760    0.1233
    0.3510    0.4018    0.2399    0.1839
```

- `eye(n)` : (ماتریس یکه‌ی واحد) از این تابع برای ساختن ماتریس n×n با عناصر قطر اصلی ۱ استفاده می‌شود

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

- `magic(n)` : دسته ماتریس‌های تعریف شده‌ی دیگری که در کتاب‌ها و مثال‌های help متلب دیده می‌شوند ماتریس‌های جادویی می‌باشند که یک ماتریس n×n بوده و عناصر آن اعداد 1 تا n^2 می‌باشند که به‌ترتیبی در ماتریس چیده می‌شوند که جمع همه‌ی سطرها و همچنین همه‌ی ستون‌ها یکسان باشد، این ماتریسها بصورت magic(n) در متلب قابل فراخوانی هستند و اغلب در مثال‌ها برای این منظور که زحمت تعریف یک ماتریس را به خود ندهیم از آنها استفاده می‌کنیم.

```
>> magic(3)      →        جمع همه عناصر هر یک از سطرها و همچنین ستون‌ها در اینجا ۱۵ است
ans =
     8     1     6
     3     5     7
     4     9     2
```

```
>> magic(4)      →        جمع همه عناصر هر یک از سطرها و همچنین ستون‌ها در اینجا ۳۴ است

ans =
    16     2     3    13
     5    11    10     8
     9     7     6    12
```

```
     4    14    15     1
```

۳.۱.۳ تعریف ماتریس با استفاده ازماتریسهای تعریف شده و معلوم

در تعریف آرایه‌ها و یا ماتریس‌ها می‌توانیم از آرایه‌ها و یا ماتریس‌های تعریف شده استفاده کنیم مثالهای زیر موضوع را روشن‌تر می‌کند

```
>> a=1:4 ;
>> b=[2 6];
>> c=[1 2  a  b];

>> A=[a ; a ; 1 6 8 9]
A =
     1     2     3     4        → a
     1     2     3     4        → a
     1     6     8     9

>> D=[A,A,A]
D =
 1    2    3    4    1    2    3    4    1    2    3    4
 1    2    3    4    1    2    3    4    1    2    3    4
 1    6    8    9    1    6    8    9    1    6    8    9
 -------  A  -----   -----  A  -----    ------    A   ---

>>G=[A ; A]
G =
     1     2     3     4         |
     1     2     3     4        A
     1     6     8     9         |
     1     2     3     4         |
     1     2     3     4        A
     1     6     8     9         |

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

```
 1     6     8     9     1     6     8     9     1     6     8     9
```

- `cat(dim,A,B)` : این دستور کاری مشابه در عبارت‌های بالا نوشتیم انجام می‌دهد بدین صورت که cat(1,A,B) ماتریسهای A وB را بصورت ستونی کنار هم قرار می‌دهد و cat(2,A,B) ماتریسهای AوB را بصورت سطری در کنار هم قرار می‌دهد cat(1,A,B)≡[A;B] وcat(2,A,B)≡[A,B]

- `repmat(A,n,m)` : نتیجه معادل ماتریس ones(n , m) با عناصر A بجای عناصر یک است از این دستور بمنظور تکرار یک آرایه و یا ماتریس به تعداد دلخواه استفاده می‌کنیم که برای مثال در ایجاد قطار پالس کاربرد دارد (تکرار یک پالس به تعداد n بار (repmat(c,1,n)

برای مثال repmat(A,2,3) بصورت زیر است :

☑ $repmat(A,2,3)=\begin{pmatrix} A & A & A \\ A & A & A \end{pmatrix}$

☑ $repmat(A,1,4)\equiv(A \quad A \quad A \quad A)$

```
>> c=[1 4 2];
>> repmat(c,2,3)

ans =
     1     4     2     1     4     2     1     4     2
     1     4     2     1     4     2     1     4     2
```

۳.۲ فراخوانی و مقدار دهی ماتریسها

```
>>B=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]
B =
     1     2     3     4
     5     6     7     8
```

```
     9    10    11    12
    13    14    15    16

>>B(1,1)
ans =
     1

>>B(2,3)
ans =
     7

>>B(3,end)              →سطر ۳  ستون آخر
ans =
    12

>>B(end,end)            →سطر آخر  ستون آخر
ans =
    16

>>B(end,end-2)
ans =
    14

>>B(1:3,end)             →سطر ۱ تا ۳  ستون آخر
ans =
     4
     8
    12

>>B(3,2:4)               →  سطر ۳  ستون ۲تا۴
ans =
    10    11    12

>>B(2,[2 3])             →سطر ۲  ستون۲و۳
ans =
     6     7

>>B(2,1:end)              →سطر ۲  ستون۱ تا  آخر
ans =
     5     6     7     8

>>B(2,:)                  →سطر ۲  ستون همه  →  سطر ۲
ans =
```

```
     5     6     7     8


>>B(1,:)                  سطر ۱→
ans =
     1     2     3     4

>>B(:,3)                  ستون۳→
ans =
     3
     7
    11
    15

>>B(:,2)                    ستون ۲→
ans =
     2
     6
    10
    14

>>B(1:2,:)                  → سطر ۱تا ۲
ans =
     1     2     3     4
     5     6     7     8

>>B([3 2],:)                  →      سطر ۲و۳
ans =
     9    10    11    12
     5     6     7     8

>>B(:,[2 3])                  →  ستون ۲و۳
ans =
     2     3
     6     7
    10    11
    14    15

>> B(2,4)=300
B =
     1     2     3     4
     5     6     7   300
     9    10    11    12
    13    14    15    16
```

```
>> B(2,:)=[100 100 100 100]   → مقداردهی یک سطر
B =
     1     2     3     4
   100   100   100   100
     9    10    11    12
    13    14    15    16

>> c=[50 50 80 80];

>> B(2,:)=c
B =
     1     2     3     4
    50    50    80    80
     9    10    11    12
    13    14    15    16

>> B(:,2)=[4;4;4;4]
B =
     1     4     3     4
    50     4    80    80
     9     4    11    12
    13     4    15    16

>> B(:,2)=250                    → همه‌ی عناصر ستون ۲ برابر ۲۵۰
B =
     1   250     3     4
    50   250    80    80
     9   250    11    12
    13   250    15    16
```

☑ برای توضیحی در مورد فراخوانی سطرها و یا ستون‌های یک ماتریس توجه کنید که عبارتی مثل A(1,:) شماره‌ی سطر را قید می‌کند و ستون را آزاد می‌گذارد یعنی شماره سطر یک وشماره ستون همه یا بعبارت دیگر همان سطر 1 ویا عبارتی مثل A(:,3) ستون را قید کرده است و سطر آزاد است که بمعنای ستون 3 است، عبارتی مثل A([2 3],:) سطرهای دو و سه را مشخص می‌کند.

۳.۲.۱ فراخوانی زیر ماتریس

```
>> B=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]
B =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> B([1 2],[3 4])    →    سطر۱ ستون ۳و۴ و سطر ۲ ستون ۳و۴
ans =
     3     4
     7     8

>> B([2 1],[2 4])
ans =
     6     8
     2     4

>> B(:)    →    نمایش ستونی ماتریس  ستون‌ها زیر هم چیده می‌شوند
ans =
     1
     5
     9
    13
     2
     6
    10
    14
     3
     7
    11
    15
     4
     8
    12
    16

>> B(1)    →    فراخوانی تک اندیسی برای ماتریس
ans =
     1

>> B(2)
```

```
ans =
     5

>> B(7)
ans =
    10
```

۳.۲.۲ ❋تبدیل اندیس ها

در اینجا این توضیح را می‌آوریم که دستورها، بخش‌ها و یا فصل‌هایی که با ❋ علامت‌گذاری شده‌اند برای مطالعه تکمیلی بوده و خواننده می‌تواند از مطالعه‌ی این قسمت‌ها صرف‌نظر کند و حذف آنها ایرادی به یادگیری مطالب بعدی وارد نمی‌آورد.

- `[i , j] = ind2sub( [m , n] , index)` :تبدیل نمایش اندیسی به نمایش معمولی، سطر و ستونی برای یک ماتریس با ابعاد m×n

```
>> B=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16;17 18 19 20]
B =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
    17    18    19    20

>> B(3)
ans =
     9

>> [i,j]=ind2sub(size(B),3)
i =
     3
j =
     1

>> B(i,j)
ans =
```

```
     9

>> B(10)
ans =
    18

>> [i,j]=ind2sub(size(B),10)
i =
     5
j =
     2

>> B(i,j)
ans =
    18

>> B(8)
ans =
    10

>> [i,j]=ind2sub(size(B),8)
i =
     3
j =
     2

>> B(i,j)
ans =
    10
```

۳.۲.۳ ❋دستور reshape()

- `reshape(A,n,m)` : ماتریس A را تبدیل به یک ماتریس n×m می‌کند برای این منظور تعداد عناصر A باید به تعداد n×m باشد در غیر این صورت با خطای متلب مواجه خواهیم شد

```
>> C=[1 2;3 4]
C =
```

```
     1     2
     3     4

>> reshape(C,1,4)
ans =
     1     3     2     4

>> D=[1 2 3;4 5 6]
D =
     1     2     3
     4     5     6

>> reshape(D,1,6)
ans =
     1     4     2     5     3     6
```

→ ترتیب عناصر ستون‌های ماتریس اصلی است : عنصر اول ستون اول عنصر دوم ستون اول ... عنصر اول ستون دوم

```
>> B=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16];

>> reshape(B,1,16)

ans =
  1  5  9  13  2  6  10  14  3  7  11  15  4  8  12  16

>> reshape(B,2,8)
ans =
     1     9     2    10     3    11     4    12
     5    13     6    14     7    15     8    16
```

→ عناصر بترتیب در ستونها چیده می‌شوند

```
>> reshape(B,3,4)

??? Error using ==> reshape
To RESHAPE the number of elements must not change.
```

→ تعداد عناصر ماتریس B(4×4=16) با 3×4=12 نمی‌خواند

```
>> reshape(B,8,2)
ans =
     1     3
```

```
     5     7
     9    11
    13    15
     2     4
     6     8
    10    12
    14    16
```

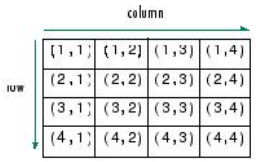
۳.۳ ماتریسهای سه بعدی و یا بالاتر

برای یک ماتریس از لحاظ تعریف برای بعد هیچ محدودیتی وجود ندارد اما عموم ماتریسهایی که ما با آنها سروکار داریم ماتریسهای دو بعدی هستند. در مباحث معمول کمتر به یک ماتریس با بعد بالاتر نیاز پیدا خواهیم کرد مگر حالت خاصی که بخواهیم تعدادی ماتریس را با نام یکسان ذخیره کنیم اما در بحث‌های مقداری پیشرفته‌تر ماتریسهای با مرتبه ۳ مواردی هستند به آشنایی و کار با آنها نیاز داریم.

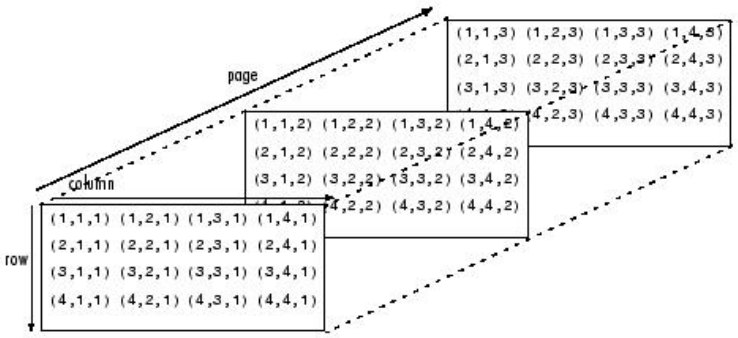
۳.۳.۱ ❋تعریف و فراخوانی ماتریس مرتبه ۳

یک ماتریس مرتبه سه را می‌توانیم بصورت عنصر به عنصر تعریف کنیم و یا اینکه با ماتریس‌های با سایز یکسان که در بعد سوم کنار هم چیده می‌شوند آن را ایجاد کنیم. برای راحتی می‌توانیم چنین تصور کنیم که چندین صفحه در فضا به موازات هم چیده می‌شوند. شکل‌های help متلب در این زمینه بسیار گویا است و ما همین شکل‌ها را در ادامه می‌آوریم

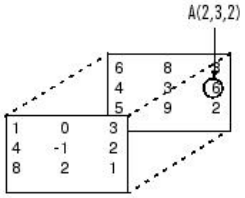
ماتریس دوبعدی



ماتریس سه بعدی



برای نمونه به نحوه‌ی تعریف جایگاه عناصر در ماتریس سه بعدی زیر توجه کنید




```
A(:,:,1) =

     1     0     3
     4    -1     2
     8     2     1

A(:,:,2) =

     6     8     3
     4     3     6
     5     9     2
```

مثال‌های زیر چند نمونه از تعریف ماتریس سه‌بعدی هستند

```
>> A(1,1,1)=5;A(1,1,2)=8;A(1,2,1)=10;A(1,2,2)=2;

>> A
A(:,:,1) =
     5    10

A(:,:,2) =
     8     2

>> I=ones(3,4,2)
```

```
I(:,:,1) =

     1     1     1     1
     1     1     1     1
     1     1     1     1

I(:,:,2) =

     1     1     1     1
     1     1     1     1
     1     1     1     1
```

۳.۳.۲ فراخوانی و مقداردهی ماتریس‌های سه‌بعدی

برای آشنایی با فراخوانی در ماتریس‌های سه‌بعدی مثالهای زیر را دنبال کنید

```
>> U=rand(3,4,2)

U(:,:,1) =

    0.1681    0.8099    0.0309    0.4104
    0.0526    0.9793    0.6531    0.1698
    0.5893    0.4631    0.1770    0.3615


U(:,:,2) =

    0.9446    0.9381    0.6455    0.3062
    0.2142    0.6617    0.8530    0.5814
    0.3662    0.9839    0.1337    0.6031



>> U(1,2,2)

ans =
```

```
    0.9381


>> U(3,4,2)

ans =
    0.6031



>> U(2,3,2)
ans =
    0.8530


>> U(2,:,2)

ans =
    0.2142    0.6617    0.8530    0.5814


>> U(1,:,2)

ans =
    0.9446    0.9381    0.6455    0.3062

>> U(:,1,2)

ans =
    0.9446
    0.2142
    0.3662

>> U(:,3,1)

ans =
    0.0309
    0.6531
    0.1770

>> U(1,3,:)

ans(:,:,1) =
```

```
    0.0309

ans(:,:,2) =
    0.6455

>> U(:,:,1)

ans =
    0.1681    0.8099    0.0309    0.4104
    0.0526    0.9793    0.6531    0.1698
    0.5893    0.4631    0.1770    0.3615
```

پس از آشنایی با فراخوانی، مقداردهی ماتریس‌های سه‌بعدی دیگر کار دشواری نیست

۳.۳.۳ ذخیره چند ماتریس در یک ماتریس سه بعدی

یکی از مواردی که عموما در کار با ماتریس‌های سه‌بعدی به آن برخورد می‌کنیم ذخیره‌ی چند ماتریس دوبعدی در یک ماتریس سه‌بعدی است

برای مثال فرض کنید می‌خواهیم سه ماتریس ۵×۴ ، A و BوC را که بصورت زیر تعریف شده‌اند را در یک ماتریس سه‌بعدی ذخیره کنیم

```
>> A=zeros(4,5);
>> B=ones(4,5);
>> C=5*ones(4,5);
```

با روش زیر این ماتریس‌ها را در ماتریس سه‌بعدی ذخیره می‌کنیم

```
>> D(:,:,1)=A;
>> D(:,:,2)=B;
>> D(:,:,3)=C;

>> whos
  Name      Size             Bytes  Class     Attributes
```

```
  A         4x5                 160  double
  B         4x5                 160  double
  C         4x5                 160  double
  D         4x5x3               480  double
```

D یک ماتریس سه بعدی است

```
>> D(:,:,1)

ans =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

همان A است

```
>> D(:,:,3)

ans =
     5     5     5     5     5
     5     5     5     5     5
     5     5     5     5     5
     5     5     5     5     5
```

همان C است

```
>> D(1,:,:)

ans(:,:,1) =
     0     0     0     0     0

ans(:,:,2) =
     1     1     1     1     1

ans(:,:,3) =
     5     5     5     5     5
```

ترکیبی از ماتریس‌هاست

عکس این کار نیز ذخیره‌ی عناصر یک ماتریس سه‌بعدی در چند ماتریس دوبعدی است، برای مثال فرض کنید همین ماتریس D را می‌خواهیم در چند ماتریس دوبعدی ذخیره کنیم

```
>> A1=D(:,:,1);
>> A2=D(:,:,2);
>> A3=D(:,:,3);
```

```
>> A2
A2 =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

اگر بخواهیم عبارت را بصورت زیر بنویسیم

```
>> A1=D(1,:,:)

A1(:,:,1) =
     0     0     0     0     0

A1(:,:,2) =
     1     1     1     1     1

A1(:,:,3) =
     5     5     5     5     5
```

با توجه به اینکه D(1,:,:) یک ماتریس سه‌بعدی است، A1 نیز یک ماتریس سه بعدی خواهد شد که با آنچه ما می‌خواستیم متفاوت خواهد بود

گهگاهی نیز در ذخیره ماتریس‌ها در یک ماتریس سه‌بعدی ممکن است با نوشتن دستور بصورت اشتباه به خطای زیر برخورد کنیم

```
>> D(4,:,:)=A
??? Subscripted assignment dimension mismatch.
```

که در این موارد خطا به علت یکسان نبودن ابعاد طرفین تساوی رخ داده است

صورت صحیح عبارت بالا بشکل زیر است که قبلا ذکر شد

```
>> D(:,:,4)=A
```

فصل 4

۴ عملیات جبری مقدماتی

۴.۱ مقدمه

ساده‌ترین اعمال جبری جمع و تفریق هستند. تنها نکته‌ای که وجود دارد این است که در مورد بردارها و یا ماتریس‌ها در صورت جمع یا تفریق هردو ماتریس (و یا بردار) باید طول یکسانی داشته باشند. استثنا درمورد جمع آرایه‌ها و یا ماتریس‌ها با یک عدد است که در این صورت همه‌ی عناصر با عدد جمع می‌شوند. ضرب ماتریسی نیز عمل آشنایی است

```
>> 2+2

ans =
     4

>> 2-2

ans =
     0

>> 2*2

ans =
     4

>> 2/2

ans =
     1

>> 10/5

ans =
     2

>> 10\5     → ۵/۱۰ تقسیم از راست معادل

ans =
    0.5000
```

```
>> 2^10

ans =
        1024

>> 2^0.5    →    √2 معادل

ans =
    1.4142

>> sqrt(2)
→  تابع رادیکال است  sqrt()

ans =
    1.4142

>> 2^-1

ans =
    0.5000

>>A=[1 2 3;4 5 6; 7 8 9];B=[10 20 30;40 50 60; 70 80 90];

>>a=[1 2 3];b=[10 20 30];

>>a+b

ans =
    11    22    33

>>A+B

ans =
    11    22    33
    44    55    66
    77    88    99

>>a+6

ans =
     7     8     9


>>B+.2
```

```
ans =
   10.2000   20.2000   30.2000
   40.2000   50.2000   60.2000
   70.2000   80.2000   90.2000

>> a/2
ans =
    0.5000    1.0000    1.5000

>> 2*a
ans =
     2     4     6

>> 10*A
ans =
    10    20    30
    40    50    60
    70    80    90

>> A/5
ans =
    0.2000    0.4000    0.6000
    0.8000    1.0000    1.2000
    1.4000    1.6000    1.8000

>> A+B
ans =
    11    22    33
    44    55    66
    77    88    99

>> A*A   → ضرب ماتریسی
ans =
    30    36    42
    66    81    96
   102   126   150

>> A^2   → A^2 ≡A*A
ans =
    30    36    42
    66    81    96
   102   126   150
```

```
>> A^3     → A^3 ≡A*A*A
ans =
         468         576         684
        1062        1305        1548
        1656        2034        2412

>> C=[1 2;3 4];

>> C^-1
ans =
   -2.0000    1.0000
    1.5000   -0.5000
```

→معادل معکوس ماتریسA است (در این خصوص در زیرفصل مربوطه صحبت خواهد شد)

```
>> a*A     → ضرب بردار در ماتریس
ans =
    30    36    42

>> A*B
ans =
         300         360         420
         660         810         960
        1020        1260        1500
```

☑ در جمع (و یا ضرب ، تفریق و تقسیم) آرایه یا ماتریس با یک عدد همه عناصر با آن عدد جمع (و یا ضرب ، تفریق ، و تقسیم) می‌شوند

۴.۱.۱ ترانهاد

عملگر `'` عملگر ترانهاده می‌باشد و برای یک ماتریس با عناصر حقیقی ترانهاده‌ی ماتریس را بدست می‌دهد ترانهاد یک بردار سطری یک بردار ستونی و همینطور ترانهاد یک بردار ستونی بردار سطری است. اگر عناصر ماتریس مختلط باشند عملگر فوق علاوه بر ترانهادگیری بجای عناصر ماتریس مزدوج مختلط آنها را نیز حساب می‌کند. برای اجتناب از این امر از عملگر `.'` استفاده می‌کنیم که در هر دو صورت عناصر حقیقی و یا مختلط فقط ترانهاد را بدست می‌دهد.

```
>> a=1:4;

>> b=[1-3i 1+8i 5-6i 7+7i];
```

```
>> a
a =
     1     2     3     4

>> a'
ans =
     1
     2
     3
     4
```

```
>> transpose(a)     →     تابع ترانهاد که معادل عملگر می‌باشد
ans =
     1
     2
     3
     4
```

```
>> a.'
ans =
     1
     2
     3
     4
```

→ دو عملگر پرایم و دات پرایم برای آرایه‌های حقیقی معادل‌اند

```
>> b=[1-3i ;1+8i; 5-6i ;7+7i]
b =
   1.0000 - 3.0000i
   1.0000 + 8.0000i
   5.0000 - 6.0000i
   7.0000 + 7.0000i
```

```
>> b'     → ترانهد علاوه‌ی مزدوج مختلط
ans =
1.0000  +  3.0000i     1.0000  -  8.0000i     5.0000  +  6.0000i
7.0000 - 7.0000i
```

```
>> b.'    → فقط ترانهاد
ans =
1.0000  -  3.0000i     1.0000  +  8.0000i     5.0000  -  6.0000i
7.0000 + 7.0000i

>> A=randint(4,4,[-10 10])    → یک ماتریس دلخواه
A =
    -5     4    -1    -7
   -10    -4    -2     0
    -8     9     6    -1
     7   -10     6     3

>> A'
ans =
    -5   -10    -8     7
     4    -4     9   -10
    -1    -2     6     6
    -7     0    -1     3

>> a*a'
ans =
    30

>> b*b'
ans =
    3000
```

۴.۲ ضرب ، تقسیم و توان عنصر به عنصر

وقتی دو ماتریس را با هم جمع می‌کنیم هر عنصر با عنصر متناظر خود جمع می‌شود اگر بخواهیم چنین موردی نیز برای ضرب داشته باشیم یعنی اینکه برای مثال در ضرب دو ماتریس تنها عناصر متناظر، در هم ضرب شوند می‌توانیم از ضرب عنصر به عنصر تعریف شده در متلب استفاده کنیم که با نماد `*.` (می‌توانیم بخوانیم دات ضرب) نشان داده می‌شود. شبیه این عمل را برای تقسیم و توان نیز داریم `/.` (می‌توانیم بخوانیم دات تقسیم) و `^.` (می‌توانیم بخوانیم دات توان)

```
>> a=1:4
a =
     1     2     3     4
```

```
>> b=[10 20 30 40];
b =
    10    20    30    40

>> a.*b
ans =
    10    40    90   160

>> a*b      → ابعاد متناسب نیست
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> a./b
ans =
    0.1000    0.1000    0.1000    0.1000


>> b./a
ans =
    10    10    10    10


>> b^a      → تعریف نشده
??? Error using ==> mpower
At least one operand must be scalar.

>> b.^a
ans =
     10         400       27000     2560000

>> b^2
??? Error using ==> mpower
Matrix must be square.

>> b.^2
ans =
     100         400         900        1600
```

```
>> b.^0.5
ans =
    3.1623    4.4721    5.4772    6.3246

>> 2.^a
ans =
     2     4     8    16

>> 2.^b
ans =
  1.0e+012 *     → همه‌ی عناصر باید در این عدد ضرب شوند
    0.0000    0.0000    0.0011    1.0995

>> 3.^a
ans =
     3     9    27    81

>> 2.^[0:5]
ans =
     1     2     4     8    16    32

>> [0:5].^2
ans =
     0     1     4     9    16    25


>> A=randint(4,4,[-10 10])
A =
     4     3    10     5
     5    -7    -3    -5
    -5    -8     2     0
     4     0    -6     4


>> B=magic(4)
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
>> A.*B
ans =
    64     6    30    65
    25   -77   -30   -40
   -45   -56    12     0
    16     0   -90     4

>> A./B
ans =
    0.2500    1.5000    3.3333    0.3846
    1.0000   -0.6364   -0.3000   -0.6250
   -0.5556   -1.1429    0.3333         0
    1.0000         0   -0.4000    4.0000

>> A.^2
ans =
    16     9   100    25
    25    49     9    25
    25    64     4     0
    16     0    36    16

>> B.^0.5            → است sqrt(B)  معادل
ans =
    4.0000    1.4142    1.7321    3.6056
    2.2361    3.3166    3.1623    2.8284
    3.0000    2.6458    2.4495    3.4641
    2.0        3.7417    3.8730    1.0000
```

فصل ۵

۵ توابع مقدماتی

۵.۱ توابع مثلثاتی

در این فصل سعی در معرفی تعدادی از پرکاربردترین توابع موجود در متلب داریم که البته در فصول فبل با تعدادی از آنها آشنا شده‌ایم

```
>> a=[pi/2 .2  pi/5 -.2]
a =
    1.5708    0.2000    0.6283   -0.2000

>> A=[pi/2 pi;-pi/3 pi/6 ]
A =
    1.5708    3.1416
   -1.0472    0.5236

>> rad2deg(pi)
ans =
   180

>> pi*(180/pi)
ans =
   180

>> deg2rad(90)
ans =
    1.5708

>> 90*(pi/180)
ans =
    1.5708

>> sin(pi)
ans =
  1.2246e-016


>> sin(pi/4)
ans =
    0.7071
```

```
>> sin(a)
ans =
    1.0000    0.1987    0.5878   -0.1987

>> sin([0:.2:1]*pi)
ans =
     0    0.5878    0.9511    0.9511    0.5878    0.0000

>> sin(A)
ans =
    1.0000    0.0000
   -0.8660    0.5000

>> cos(pi/2)
ans =
  6.1232e-017      → با تقریب خوبی صفر است

>> tan(pi/6)
ans =
    0.5774

>> cot(pi/4)
ans =
    1.0000
```

۵.۱.۱ توابع مثلثاتی با آرگومان ورودی بر حسب درجه

```
>> sind(30)
ans =
    0.5000
```

```
>> sin(30*(pi/180))    → معادل کار sind
```

```
ans =
    0.5000

>> sind(90)
ans =
     1

>> cosd(180)
ans =
    -1
```

۵.۱.۲ معکوس توابع مثلثاتی

```
>> asin(.5)
ans =
    0.5236

>> acos(0)
ans =
    1.5708

>> asin(3)
ans =
   1.5708 - 1.7627i

>> atan(2)
ans =
    1.1071

>> acot(1)
ans =
    0.7854

>> asin(3)      → برای اعداد حقیقی سینوس بین ۱- و ۱ است
ans =
   1.5708 - 1.7627i
```

۵.۱.۳ معکوس توابع مثلثاتی با خروجی درجه

```
>> asind(.5)
ans =
   30.0000

>> asin(.5)*(180/pi)
```
→ معادل کار asind را انجام می‌دهد
```
ans =
   30.0000

>> asind(1/sqrt(2))
ans =
   45.0000

>> asind(1)
ans =
    90

>> atand(1)
ans =
    45
```

۵.۲ توابع نمایی و لگاریتمی

- `exp(x)` : مقدار e^x را محاسبه می‌کند

```
>> a=[pi/2 .2  pi/5 -.2]
a =
    1.5708    0.2000    0.6283   -0.2000
```

```
>> A=[pi/2 pi;-pi/3 pi/6 ]
A =
    1.5708    3.1416
   -1.0472    0.5236

>> exp(3)
ans =
   20.0855


>> e^3
??? Undefined function or variable 'e'.
```

→ این فرم نوشتار برای تابع e^x صحیح نیست و بسته به اینکه e چه باشد جواب متفاوت خواهد بود

```
>> e=10;e^3       → 10^3
ans =
        1000

>> exp(-3)
ans =
    0.0498


>> exp(-2:4)
ans =
 0.1353  0.3679   1.0000  2.7183  7.3891  20.0855  54.5982


>> exp(a)
ans =
    4.8105    1.2214    1.8745    0.8187


>> exp(A)
ans =
    4.8105   23.1407
    0.3509    1.6881

>> 2^4
ans =
    16
```

```
>> 2.^A
ans =
    2.9707    8.8250
    0.4839    1.4375
```

۵.۲.۱ توابع لگاریتمی

- `log(x)` : لگاریتم طبیعی Ln(x)
- `log10(x)` : لگاریتم مبنای ۱۰
- `log2(x)` : لگاریتم مبنای ۲

```
>> log10(5)
ans =
    0.6990

>> log10(10)
ans =
     1

>> log10(10^4)
ans =
     4

*>> log10(-2)
ans =
   0.3010 + 1.3644i

>> log(10)
ans =
    2.3026

>> log(3)
ans =
    1.0986
```

```
>> log(exp(8))
ans =
     8

>> log2(5)
ans =
    2.3219

>> log2(2^4)
ans =
     4
```

۵.۲.۲ توابع هیپربولیک

```
>> sinh(.2)
ans =
    0.2013

>> tanh(.6)
ans =
    0.5370

>> asinh(5)
ans =
    2.3124

>> atanh(.8)
ans =
    1.0986
```

۵.۳ توابع نظریه اعداد

۵.۳.۱ جزء صحیح و گرد کردن

توابع معروف برای قسمت صحیح اعداد و گردکردن اعداد توابع زیراند

- `floor(x)` : جزء صحیح x
- `ceil(x)` : گرد کردن به سمت عدد صحیح بزرگتر
- `fix(x)` : قسمت صحیح
- `round(x)` : گرد کردن بسمت عدد صحیح نزدیکتر

```
>> floor(2.6)
ans =
     2

>> floor(-2.6)
ans =
    -3

>> fix(-2.6)
ans =
    -2

>> ceil(1.1)
ans =
     2

>> round(1.2)
ans =
     1

>> round(1.8)
ans =
     2
```

- `rem(a,b)` : باقیمانده تقسیم a به b

```
>> rem(15,7)
ans =
     1

>> rem(15.3,7)
ans =
    1.3000
```

۵.۳.۲ توابع مختلف دیگر از نظریه اعداد

- `primes(n)` : اعداد اول کوچکتر از n را لیست می‌کند
- `isprime(a)` : برسی می‌کند که آیا a اول است (۱) یا خیر (۰)
- `factorial(n)` : n!
- `factor(a)` : a را به عوامل اول تجزیه می‌کند
- `[num , den] = rat(a)` : عدد a را بصورت کسری تبدیل می‌کند
- `lcm(a,b)` : ک.م.م
- `gcm(a,b)` : ب.م.م

- `nchoosek(n,k)` : $\binom{n}{k}$ انتخاب k شی از n شی
- `perms(x)` : تمام جایگشت‌های درایه‌های x

```
>> primes(10)
ans =
     2     3     5     7

>> primes(20)
ans =
     2     3     5     7    11    13    17    19
```

```
>> isprime(20)
ans =
     0

>> isprime(17)
ans =

     1

>> factorial(4)
ans =
    24


>> factor(50)
ans =
     2     5     5

>> factor(16)
ans =
     2     2     2     2

>> [num,den]=rat(.365)
num =
    73
den =
   200

>> num/den
ans =
    0.3650

>> [num,den]=rat(pi)
num =
   355
den =
   113

>> num/den
ans =
    3.1416
```

```
>> lcm(6,8)
ans =
    24

>> gcd(6,8)
ans =
     2

>> perms([2 3 7])
ans =
     7     3     2
     7     2     3
     3     7     2
     3     2     7
     2     3     7
     2     7     3

>> n=7;k=3;
>> nchoosek(n,k)
ans =
    35


>> factorial(n)/(factorial(k)*factorial(n-k))
ans =
    35
```

۵.۴ *چند تابع پرکاربرد دیگر

۵.۴.۱ *تابع سینک

$sinc(x) = \frac{\sin(\pi x)}{\pi x}$: `sinc(x)*`

تابع sinc از توابع معروف در ریاضیات مهندسی و برق است

```
>> sinc(0)
ans =
     1
```

```
>> sinc(1)    → sinc(n)=0      n عدد صحیح
ans =
  3.8982e-017

>> sinc(2)
ans =
 -3.8982e-017

>> sinc(.1)
ans =
    0.9836
```

۵.۵ توزیع نرمال

توزیع نرمال یکی از مواردی است که در محاسبات خود زیاد بدان بر می‌خوریم تابع چگالی توزیع احتمال (pdf) این توزیع بصورت زیر است

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

که در آن μ میانگین و σ واریانس توزیع می‌باشند

تابع توزیع تجمعی با این مفهوم که: احتمال اینکه مقدار متغیر تصادفی X (که دارای توزیع نرمال است) از مقدار x کمتر باشد. بصورت زیر می‌باشد

$$P(X < x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \, dx$$

این توزیع تجمعی برای حالت میانگین صفر و واریانس یک به Q-function معروف است

احتمال اینکه مقدار متغیر تصادفی X بین x1 و x2 باشد نیز از رابطه زیر بدست می‌آید

$$P(X \in [x1 \ x2]) = Q(x2) - Q(x1)$$

۵.۵.۱ تولید رشته‌ای از اعداد با توزیع نرمال

- `normrdn(mu,sigma)` : randn(1) یک عدد تصادفی با توزیع نرمال میانگین mu و واریانس sigma تولید می‌کند. حال اگر بخواهیم یک ماتریس n×m از اعداد تصادفی با توزیع نرمال داشته باشیم از normrnd(mu,signa,n,m) استفاده می‌کنیم

- `normcdf(x,mu, sigma)` : احتمال اینکه مقدار متغیر تصادفی با توزیع نرمال با میانگین mu و واریانس sigma از x کمتر باشد

```
>> mu=0;sigma=1;
>> n=normrnd(mu,sigma,1)
n =
    0.5377

>> n=normrnd(mu,sigma,1)
n =
    1.8339

>> n=normrnd(mu,sigma,1,5)
→  آرایه ۵*۱  از اعداد تصادفی با توزیع نرمال و میانگین و واریانس مشخص
n =
   -2.2588    0.8622    0.3188   -1.3077   -0.4336


>> P = normcdf(1,mu,sigma)→ احتمال اینکهx  کمتر از ۱ باشد
P =
    0.8413

>> P = normcdf(3,mu,sigma)→  احتمال اینکهx  کمتر از ۳ باشد
P =
    0.9987

>> P = normcdf([1 3],mu,sigma)
P =
    0.8413    0.9987
>> P(2)-P(1)    →  احتمال اینکه x  بین ۱ و ۳ باشد
```

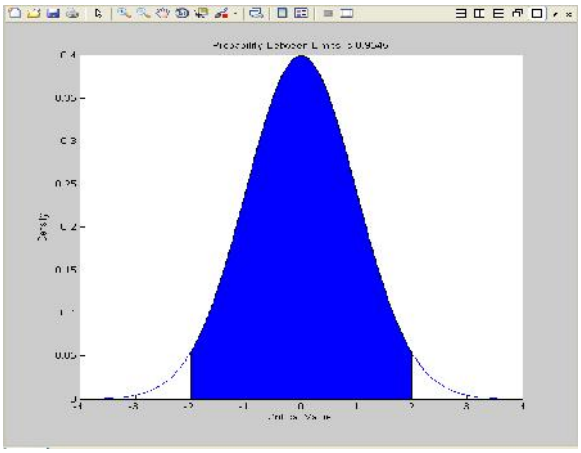
```
ans =
    0.1573
```

`normspec([x1 x2],mu,sigma)` : احتمال اینکه متغیر تصادفی با میانگین mu و واریانس sigma بین x1 و x2 باشد همراه با نمودار توزیع‌تجمعی بسیار گویا

احتمال اینکه یک عدد تصادفی با توزیع نرمال استاندارد بین ۲- و ۲ باشد چقدر است؟

```
>> mu=0;sigma=1;
>> P = normcdf([-2 2],mu,sigma)
P =
    0.0228    0.9772

>> P(2)-P(1)
ans =
    0.9545
```

```
>> mu=0;sigma=1;
>> p=normspec([-2 2],mu,sigma)
p =
    0.9545
```



- `x = norminv(p,mu,sigma)` : مقدار x را حساب می‌کند که بازای آن توزیع تجمعی برابر p شود یعنی: $x : P(X < x) = p$

  این تابع ، تابع معکوس توزیع تجمعی است.

- محل ۹۰٪ توزیع تجمعی نرمال میانگین صفر و واریانس یک در کجا قرار دارد؟

```
>> x=norminv(.9,0,1)
x =
    1.2816
```

- بازه‌ای متقارن حول صفر که ۹۵٪ مقادیر توزیع نرمال استاندارد در آن قرار دارد را پیدا کنید

```
>> p=[.025 .975]
p =
    0.0250    0.9750

>> p(2)-p(1)
ans =
    0.9500

>> x = norminv(p,0,1)
x =
   -1.9600    1.9600
```

فصل ۶

۶ توابع پرکاربرد آرایه‌ای و ماتریسی

۶.۱ اندازه و کمینه‌ی ماتریس

- `length(a)` : طول بردار a را بدست می‌دهد
- `size(A)` : سایز ماتریس را نشان می‌دهد
- `numel(A)` : نتیجه عددی است که برابر تعداد درایه‌های A است

☑ وقتی می‌گوییم اندازه‌ی یک ماتریس 3×2 است یعنی تعداد سطرها ۲ و تعداد ستونها ۳ است از آنجا که تعداد سطرها را در ابتدا و تعداد ستونها را در مرتبه دوم ذکر می‌کنیم این قرارداد را ذکر می‌کنیم بعد اول سطر ، و بعد دوم ستون باشد ممکن است ابعاد دیگری هم داشته باشیم. این قرارداد در بسیاری از موارد در فراخوانی دستورها می‌تواند به ما کمک کند.

```
>> A=zeros(5,8);
>> B=rand(12,4);
>> c=ones(1,7);
>> d=ones(4,1);

>> length(c)
ans =
     7

>> length(d)
ans =
     4

>> size(A)
ans =
     5     8

>> size(B)
ans =
    12     4
```

```
>> [m,n]=size(A)
m =
     5
n =
     8

>> [m,n]=size(c)
m =
     1
n =
     7

>> x=size(A)     →x یک بردار سطری به طول دو است
x =
     5     8

>> m*n
ans =
     7

>> numel(A)     → number of elements
ans =
    40

>> numel(B)
ans =
    48

>> numel(c)
ans =
     7

>> size(A,1)      → تعداد عناصر بعد ۱ (تعداد سطر ها)
ans =
     5

>> size(A,2)       → تعداد عناصر بعد ۲ (تعداد ستون ها)
ans =
     8

>> size(B,1)
ans =
    12
```

```
>> size(B,2)
ans =
     4
```

- `min(a)`: برای حالتی که a یک بردار باشد این دستور کمینه مقدار این بردار را بدست می‌دهد و در صورتی که x یک ماتریس باشد تابع min مقدار کمینه‌ی هر یک از ستون ها را حساب می‌کند و خروجی یک بردار سطری خواهد بود که هر عنصر نشان دهنده‌ی مقدار کمینه‌ی ستون متناظر است. این دستور صورتهای دیگر فراخوانی دارد که در مثال‌ها آمده است.
- `max(a)`: با کارکردی مشابه دستور min(a) مقدار بیشینه را بدست می‌دهد

```
>> min(1,4)     → مقدار کمینه از بین ۱ و ۴
ans =
     1

>> max(1,4)
ans =
     4

>> a1=7;a2=4;max(a1,a2)
ans =
     7

>> a = [1 2 3 4 5 6 7];
>> max(a,4)      → مقدار بیشینه بین درایه‌های a و ۴
ans =
     4     4     4     4     5     6     7

>> min(a,4)
ans =
     1     2     3     4     4     4     4

>> min(a,2)
```

```
ans =
     1     2     2     2     2     2     2

>> a=[1 5 20 -5 -6 8 -25 9 10 3]
a =
    1     5    20    -5    -6     8   -25     9    10     3
```

```
>> b = randint(1,10,20)    → یک آرایه‌ی تصادفی
b =
  16    18     2    18    12     1     5    10    19    19

>> min(a)    → مقدار کمینه‌ی a
ans =
    -6

>> min(b)
ans =
     1

>> min(a,b)
ans =
   1     5     2    -5    -6     1   -25     9    10     3
```

→ فراخوانی min با دو ورودی ، حاصل یک بردار با طول یکسان با بردارهای ورودی و عناصر کمینه‌ی بین دو ورودی

```
>> max(a,b)
ans =
  16    18    20    18    12     8     5    10    19    19

>> [me ma]=min(a)    → فراخوانی دستور min با دو خروجی مقدار کمینه و مکان عنصر کمینه
me =        → مقدار
    -6
ma =        → مکان
     5      → عنصر پنجم کمینه مقدار را دارد
```

تعریف نام متغیر های خروجی بعلت مشابهت با نام فارسی بوده و اختیاری است

```
>> [me ma]=min(b)
me =
     1
```

```
ma =
     6

>> [me ma]=max(a)
me =
    20
ma =
     3

>> [me ma]=min(a)
me =
    -6
ma =
     5

>> A=randint(4,5,50)
A =
     7    40    39    42    37
    48     7    47    46    19
    47    21    32    33    32
    24    45     1    37     8

>> min(A)     → مقدار کمینه‌ی هر یک از ستون‌ها : نتیجه در یک بردار سطری داده می‌شود
ans =
     7     7     1    33     8

>> max(A)
ans =
    48    45    47    46    37


>> min(A')'     → از این تکنیک برای محاسبه‌ی کمینه‌ی سطرها استفاده می‌کنیم
ans =
     7
     7
    21
     1

>> max(A')'
ans =
    42
    48
    47
    45
```

```
>> [me ma]=min(A)     → فراخوانی دستور min با دو خروجی برای یک ماتریس
me =
     7     7     1    33     8
ma =
     1     2     4     3     4 → شماره سطری که مقدار بیشینه در آن واقع است


>> [me ma]=max(A)
me =
    48    45    47    46    37
ma =
     2     4     2     2     1

>> min(min(A))      → مقدار کمینه یک ماتریس دو بعدی
ans =
     1

>> max(max(A))
ans =
    48
```

۶.۲ چرخش ماتریس‌ها

- `fliplr(a)` : برای یک بردار آرایه‌ها را از آخر به اول می‌چیند و برای ماتریس ستون‌ها بدین ترتیب چیده می‌شوند

- `flipud(A)` : برای یک ماتریس سطرها را از آخر به اول می‌چیند

- `minmax(a)` : کمینه و بیشینه‌ی بردار a را بدست می‌دهد در صورتی که ورودی یک ماتریس باشد کمینه و بیشینه‌ی سطرها را بدست می‌دهد.

- `rot90(A , k)` : این دستور ماتریس A را k*90 درجه در جهت مثبت مثلثاتی (پاد ساعتگرد) می‌چرخاند

```
>> a=[1 5 20 -5 -6 8 5 9 10 -8];
>> b=randint(1,10,20)
b =
   16    18     2    18    12     1     5    10    19    19

>> a=[1 5 20 -5 -6 8 5 9 10 -8]
a =
    1     5    20    -5    -6     8     5     9    10    -8

>> c=fliplr(a)    → چیدن a از آخر به اول
c =
   -8    10     9     5     8    -6    -5    20     5     1

>> a(end:-1:1)    →      fliplr کار معادل
ans =
   -8    10     9     5     8    -6    -5    20     5     1


>> A=randint(4,5,50)
A =
    35     4    47    38    22
     1    41     1    39    32
    13    34    21     9    35
     2    15    19    24    37

>> fliplr(A)              → چیدن ستون‌ها از آخر به اول
ans =
    22    38    47     4    35
    32    39     1    41     1
    35     9    21    34    13
    37    24    19    15     2

>> A(:,end:-1:1)          → معادل fliplr ماتریسی
ans =
    22    38    47     4    35
    32    39     1    41     1
    35     9    21    34    13
    37    24    19    15     2
```

```
>> flipud(A)                    → چیدن سطرها از آخر به اول
ans =
     2    15    19    24    37
    13    34    21     9    35
     1    41     1    39    32
    35     4    47    38    22


>> A(end:-1:1,:)               → معادل flipud ماتریسی
ans =
     2    15    19    24    37
    13    34    21     9    35
     1    41     1    39    32
    35     4    47    38    22

>> minmax(a)
ans =
    -8    20

>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> minmax(A)
ans =
     2    16
     5    11
     6    12
     1    15


------- یک مثال خاص برای rot90 ---------

>> A=imread('cameraman.tif');
   → فعلا دستور متلب را فقط به این صورت تایپ می‌کنیم و تنها با rot90 کار داریم

>> imshow(A)          → نتیجه را در متلب ببینید
>> B=rot90(A);        → چرخش ۹۰ درجه‌ای یک تصویر
```

```
>> imshow(B)          →   چرخش ۹۰ درجه‌ای تصویر را در متلب ببینید
```

۶.۳ چند تابع ریاضی

- `sign(a)` : sign تابع علامت است و برای مقادیر مثبت ۱ مقادیر منفی ۱- و برای صفر مقدار ۰ را برمی‌گرداند.
- `abs(a)` : abs تابع قدر مطلق است البته کارکردی دیگر برای اعداد مختلط دارد که در زیرفصل مربوطه ذکر خواهد شد.
- `sqrt(a)` : sqrt تابع رادیکال می‌باشد و می‌توان بجای آن بسادگی از a^0.5 و یا در صورت برداری و یا ماتریسی بودن ورودی از A.^0.5 استفاده کرد

```
>> sign(-15.2)
ans =
    -1

>> sign(0)
ans =
     0

>> a=-4:4
a =
    -4    -3    -2    -1     0     1     2     3     4

>> sign(a)
ans =
    -1    -1    -1    -1     0     1     1     1     1

>> abs(a)
ans =
     4     3     2     1     0     1     2     3     4

>> b=2:8
b =
```

```
     2     3     4     5     6     7     8

>> sqrt(b)
ans =
1.4142   1.7321   2.0000   2.2361   2.4495   2.6458   2.8284

>> b.^0.5
ans =
1.4142   1.7321   2.0000   2.2361   2.4495   2.6458   2.8284
```

- `sum(a)` : این دستور بسیار پرکاربرد است و با توانایی استفاده از آن می‌توانیم بسیاری از حلقه‌های for را از برنامه حذف کنیم، سرعت اجرای برنامه را زیاد و برنامه را خواناتر کنیم

```
>> sum(4,2)      → دستور sum  برای جمع دو عدد نیست
ans =
     4

>> a=-20:10:60
a =
   -20   -10     0    10    20    30    40    50    60

>> b=[4 -5 12 0 8 23 5 20 -12]
b =
     4    -5    12     0     8    23     5    20   -12

>> sum(a) →  جمع همه‌ی درایه‌ها
ans =
   180

>> sum(b)
ans =
    55

>> sum(abs(a))      → جمع قدر مطلق همه‌ی درایه‌ها
ans =
   240
```

```
>> sum(abs(b))
ans =
    89
```

```
>> sum(a-b)
ans =
   125

>> sum(abs(a-b))
ans =
   207

>> sum(a.^2)          →          جمع توان دو همه‌ی عناصر
ans =
        9600

>> sum((a-b).^2)
ans =
        8347

>> B=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15 ]
B =
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15

>> D=(B-5)*3
D =
   -12    -9    -6    -3     0
     3     6     9    12    15
    18    21    24    27    30

>> sum(B)       →  جمع همه‌ی ستون‌ها
ans =
    18    21    24    27    30

>> sum(B,1)
ans =
    18    21    24    27    30
```

→ جمع همه‌ی ستون‌ها (عدد ۱ بعدی است که عمل جمع در آن صورت می‌گیرد بعد سطری ۱ بعد ستونی ۲ و ...)

```
>> sum(B,3)
```

```
ans =
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
```

جمع عناصرB در بعد سوم (چون این ماتریس دو بعد بیشتر ندارد جمع روی بعد سوم خود ماتریس را نتیجه می‌دهد)

```
>> sum(4,3)   → مشابه مورد بالا
ans =
     4

>> sum(B,2)   → جمع همه‌ی سطرها
ans =
    15
    40
    65

>> sum(D,1)
ans =
     9    18    27    36    45

>> sum(D,2)
ans =
   -30
    45
   120

>> sum(sum(B))   → جمع همه‌ی عناصر
ans =
   120

>> sum(sum(D))
   135

>> sum(sum(abs(D)))
ans =
   195

>> sum(sum(abs(B-D)))
ans =
   113
```

- `prod(a)` : حاصلضرب درایه‌ها را حساب می‌کند و اگر ورودی ماتریس باشد نتیجه حاصلضرب ستون‌ها است.

```
>> a=2:6
a =
     2     3     4     5     6
>> prod(a)
ans =
   720

>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

>> prod(A)              → حاصلضرب ستون‌ها
ans =
    96    45    84

>> prod(size(A))        → عموما از این عبارت بجای دستور numel استفاده می‌شود
ans =
     9

>> numel(A)
ans =
     9
```

۶.۳.۱ *جمع و ضرب تجمعی

- `cumsum(a)`: جمع تجمعی را حساب می‌کند و نتیجه یک بردار است این دستور در بیشتر در محاسبه‌ی توزیع تجمعی از روی توزیع احتمال کاربرد دارد.

- `cumprod(a)`: عملکرد مشابه cumsum در حوزه‌ی ضرب

```
>> a=1:5
a =
     1     2     3     4     5

>> b=-4:3:8
b =
    -4    -1     2     5     8

>> cumsum(a)    → cumulative sum
ans =
     1     3     6    10    15

>> cumsum(b)
ans =
    -4    -5    -3     2    10

>> cumprod(a)
ans =
     1     2     6    24   120

>> cumprod(b)
ans =
    -4     4     8    40   320
```

۶.۳.۲ توابعی از آمار ریاضی

- `mean(a)`: میانگین بردار a را بدست می‌دهد . اگر ورودی یک ماتریس باشد نتیجه میانگین ستون‌ها است
- `median(a)`: میانه‌ی a را نتیجه می‌دهد و بیشتر یک دستور آماری است
- `var(a)`: واریانس یک بردار را بدست می‌دهد معادل عبارت ریاضی σ_a . اگر ورودی یک ماتریس باشد نتیجه واریانس ستون‌ها است

- `std(a)` : پراکندگی استاندارد آرایه‌ی ورودی را بدست می‌دهد که معادل عبارت ریاضی σ_a^2 است اگر ورودی یک ماتریس باشد نتیجه پراکندگی استاندارد ستون‌ها است

```
>> a=[5 8 -3 25 12 -8 30 0 3]
a =
     5     8    -3    25    12    -8    30     0     3

>> b=[15 -13 0 5 1 -5 10 10 3]
b =
    15   -13     0     5     1    -5    10    10     3

>> mean(a)
ans =
     8

>> mean(b)
ans =
    2.8889

>> var(a)
ans =
   158

>> std(a)
ans =
   12.5698

>> std(a)^2
ans =
   158

>> I=(sum(a.^2-mean(a)^2))/(length(a)-1)
I =
   158
```

- * `norm(a)` : نرم یک آرایه را حساب می‌کند که برای آرایه بصورت $\sqrt{\sum a_i^2}$ تعریف می‌شود در حالت کلی توان می‌تواند یک عدد دلخواه باشد $\sqrt[p]{\sum a_i^p}$ که برای این منظور تابع نرم را با دو ورودی فراخوانی می‌کنیم norm(a , p)

```
>> a=10:5:40
a =
    10    15    20    25    30    35    40

>> norm(a)
ans =
   71.2390

>> sqrt(sum(a.^2))
ans =
   71.2390

>> norm(a,10)
ans =
   41.1556

>> (sum(a.^10))^0.1
ans =
   41.1556
```

۶.۴ مرتب کردن آرایه‌ها

- `sort(a)` : این دستور آرایه را به ترتیب از مقدار کوچک به بزرگ می‌چیند

- `find()` : این دستور برای جستجو و پیداکردن مقادیر خاص در آرایه‌ها و ماتریس‌ها بکار می‌رود و استفاده از آن پیش‌نیاز عبارتهای شرطی را می‌طلبد و در اینجا در حد آشنایی ساده مطرح می‌شود و بحث کامل تا کار با آرایه‌های شرطی به تعویق می‌افتد

```
>> a=[13 17 -15 17 5 -17 -9  2 19 19]
a =
    13    17   -15    17     5   -17    -9     2    19    19


>> a1=sort(a)
a1 =
   -17   -15    -9     2     5    13    17    17    19    19


>>[a1 ind]=sort(a)     → مرتب کردن صعودی علاوه‌ی اندیس عناصر
a1 =
   -17   -15    -9     2     5    13    17    17    19    19
ind =
     6     3     7     8     5     1     2     4     9    10

>> sort(a,'ascend')     → صعودی
ans =
   -17   -15    -9     2     5    13    17    17    19    19

>> sort(a,'descend')    → نزولی
ans =
    19    19    17    17    13     5     2    -9   -15   -17

>> a1(end:-1:1)         → یک روش دیگر برای مرتب‌کردن نزولی
ans =
    19    19    17    17    13     5     2    -9   -15   -17

>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> sort(A)
ans =                          → ستون‌ها بصورت نزولی مرتب شده‌اند
     4     2     3     1
     5     7     6     8
     9    11    10    12
    16    14    15    13
```

```
>> sort(A,1)                    →    ستون ها بصورت نزولی مرتب شود

ans =

     4     2     3     1
     5     7     6     8
     9    11    10    12
    16    14    15    13

>> sort(A,2)                    →    سطرها بترتیب نزولی مرتب شوند

ans =

     2     3    13    16
     5     8    10    11
     6     7     9    12
     1     4    14    15


>> ind=find(a==2)
ind =                    →  خروجی اندیس جایی است که در آن  a  برابر ۲ است
     8
→          در اینجا از عبارت == استفاده می‌کنیم توضیح در عبارتهای شرطی خواهد آمد

>> a(ind)
ans =
     2

>> ind=find(a==17)
ind =
     2     4

>> a(ind)
ans =
    17    17
```

۶.۵ توابع اعداد مختلط

- `conj(a)` : مزدوج مختلط عدد را میدهد. بدین منظور می‌توان عملگر `'` را برای عدد و عملگر `.'` را برای آرایه و یا ماتریس استفاده کرد
- `real(a)` : قسمت حقیقی عدد مختلط
- `imag(x)` : قسمت موهومی عدد مختلط
- `abs(a)` : اندازه‌ی عدد مختلط
- `angle(a)` : فاز عدد مختلط بر حسب رادیان

```
>> a=4-3i
a =
   4.0000 - 3.0000i


>> b=5-8i
b =
   5.0000 - 8.0000i


>> d=[1-3i 4-5i -2-5i 4-8i]
d =
1.0000  -  3.0000i     4.0000  -  5.0000i   -2.0000  -  5.0000i
4.0000 - 8.0000i


>> a'
ans =
   4.0000 + 3.0000i
```

```
>> conj(a)
ans =
   4.0000 + 3.0000i

>> conj(d)
ans =
1.0000  +  3.0000i     4.0000  +  5.0000i   -2.0000  +  5.0000i
4.0000 + 8.0000i

>> d'.'
ans =
1.0000  +  3.0000i     4.0000  +  5.0000i   -2.0000  +  5.0000i
4.0000 + 8.0000i

>> real(a)
ans =
     4

>> real(d)
ans =
     1     4    -2     4

>> imag(a)
ans =
    -3

>> imag(d)
ans =
    -3    -5    -5    -8

>> abs(a)
ans =
     5

>> abs(d)
ans =
    3.1623    6.4031    5.3852    8.9443

>> angle(a)
ans =
   -0.6435
```

```
>> angle(d)
ans =
   -1.2490   -0.8961   -1.9513   -1.1071

>> 180/pi*angle(d)    → زاویه بر حسب درجه
ans =
  -71.5651  -51.3402 -111.8014  -63.4349
```

☑ دستور ()sort برای اعداد مختلط اندازه‌ی عدد مختلط را معیار قرار می‌دهد و بر این اساس اعداد را بترتیب صعودی (در اندازه) مرتب می‌کند در صورت یکسان بودن اندازه عددی که فاز بزرگتری دارد بزرگ حساب می‌شود

```
>> q=[-9 1+1i 8 3-5i 1-1i 3+5i  6]
q =
  Columns 1 through 4
-9.0000   1.0000 + 1.0000i   8.0000    3.0000 - 5.0000i

  Columns 5 through 7
1.0000 -1.0000i   3.0000 + 5.0000i   6.0000

>> w=sort(q)
w =
  Columns 1 through 3
1.0000 - 1.0000i   1.0000 + 1.0000i   3.0000 - 5.0000i

  Columns 4 through 7
3.0000 + 5.0000i      6.0000       8.0000       -9.0000

>> abs(w)     → اندازه بترتیب صعودی است
ans =
1.4142   1.4142   5.8310   5.8310   6.0000   8.0000  9.0000

>> angle(w)
ans =
-0.7854   0.7854  -1.0304   1.0304      0     0    3.1416
```

۶.۶ عبارتهای جبری

در این بخش پس از یادگیری و آشنایی با توابع مختلف می‌خواهیم صورت متلبی عبارت‌های جبری را که عموما به آنها برخورد می‌کنیم را بنویسیم. برای آشنایی بیشتر مثالهای زیر را دنبال کنید

فرض کنید x در حالت کلی یک آرایه‌ی برداری بصورت باشد $x = [x1\ x2\ \dots\ xN]$

صورت متلبی عبارتهای جبری زیر را بنویسید (جواب در سمت راست نوشته شده است)

$\sin(x)$	`>> sin(x)`
$\cos(x)$	`>> cos(x)`
$\sin^{-1}(x)$	`>> asin(x)`
$\sinh(x)$	`>> sinh(x)`
$\sinh^{-1}(x)$	`>> asinh(x)`
$\tanh^{-1}(x)$	`>> atanh(x)`
e^{x}	`>> exp(x)`
$\log(x)$	`>> log10(x)`
$\ln(x)$	`>> log(x)`
$()*()$	`>> ().*()` ⟸ حالت کلی
e^{-x^2}	`>> sin(x)./x`
$x\sin(x)$	`>> exp(-x.^2)`

$x \sin(\frac{}{x^2})$	`>> x.*sin(x)`
—	`>> x.*sin(x.^2)`
$\frac{\sin(}{x}$	`>> ()./()` ⟾ حالت کلی
$\frac{}{x^2 - 13}$	`>> (x.^3-3*x+4)./(x.^2-13)`
$\frac{x_2 \sin(x) - \cos(}{x^3 - 3x}$	`>> ((x.^2).*sin(x) - cos(x)) ... ./(x.^3 - 3*x)`
$x\, e^{\frac{-x^2}{x+3}}$	`>> x.*exp(-(x.^2)./(x+3) )`
$x - \frac{}{x^2 - 1}$	`>> x - x./( x.^2 - 1 )`
	`>> x- x/( x.^2 - 1 )` ⟾ ایراد دستوری ندارد ولی صحیح نیست
$e_x - x_2$	`>> exp(x)-x.^2`
$\frac{}{e^x + e^{-x}}$	`>> ( exp(x)- x.^2 )./( exp(x)+exp(-x) )`
$\frac{}{e^x + e^{-x}}$	`>> ( exp(x)- x.^2 )./( exp(x)+exp(-x) )`

- ✓ اگر a وb دو بردار با طول یکسان باشند `a./b` تقسیم آشنای عنصر به عنصر است نکته‌ی قابل توجه این است که `a/b` نیز در متلب تعریف شده است `a/b = a*b'/(b*b')` و در صورتیکه از عملگر `/` بجای `./` استفاده کنیم ایراد نگارشی وجود ندارد ولی عبارت محاسبه شده با عبارت مورد نظر ما متفاوت است لذا لازم است در نوشتن عبارت تقسیم عنصر به عنصر دو بردار دقت کافی داشته باشیم

۶.۶.۱ یادآوری چند نکته در مورد صفحه فرمان (command window)

در اینجا چند نکته برای آشنایی اولیه با صفحه فرمان ذکر می‌کنیم . بعدا در یک فصل مجزا بصورت مفصلتر به بررسی این صفحه و صفحات مختلف متلب خواهیم پرداخت

- ☑ ابتدای دستور متلب علامت << قرار دارد
- ☑ اگر در ابتدای خط فرمان جاری کلید بالاپیمای ↑ صفحه کلید را فشار دهیم به دستور قبل می‌رسیم
- ☑ اگر عبارتی را تایپ و سپس کلید بالاپیما ↑ را فشار دهیم دستور قبلی متلب که با این عبارت آغاز می‌شود در خط فرمان جاری ظاهر خواهد شد مثال a=↑ و یا sort↑ (علامت ↑ در اینجا بمعنای فشردن کلید بالاپیما است)
- ☑ کلید Esc خط فرمان جاری را پاک می‌کند
- ☑ دستور clc موجب پاک شدن صفحه فرمان می‌شود
- ☑ پاک شدن صفحه بمعنای پاک شدن متغیرها نیست

فصل 7

۷ جبر ماتریسی

۷.۱ جبر ماتریسی

در این فصل ما با عملیات مختلف ماتریسی در متلب آشنا می‌شویم

- `A*B` : ضرب ماتریسی دو ماتریس A و B
- `A.'` : ترانهاده A
- `det(A)` : دترمینان A
- `inv(A)` : معکوس A
- `diag(A)` : قطر اصلی A

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2


>> A.'
ans =
     8     3     4
     1     5     9
     6     7     2

>> diag(A)
ans =
     8
     5
     2
```

```
>> diag(diag(A))
ans =
     8     0     0
     0     5     0
     0     0     2

>> det(A)                  →  |A|
ans =
  -360

>> det(A)*det(inv(A))     →  |A^-1| = 1/|A^-1|
ans =
     1

>> inv(A)
ans =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028


>> A^-1    →    همان معکوس Aرا نتیجه می‌دهد
ans =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028

>> inv(A)*A
ans =
    1.0000         0   -0.0000
         0    1.0000         0
         0    0.0000    1.0000
```

```
>> B=(A-4)
B =
     4    -3     2
    -1     1     3
     0     5    -2
```

```
>> B/A          → B/A ≡ B*A^-1
ans =
    0.7333   -0.2667   -0.2667
   -0.2667    0.7333   -0.2667
   -0.2667   -0.2667    0.7333

>> B*inv(A)
ans =
    0.7333   -0.2667   -0.2667
   -0.2667    0.7333   -0.2667
   -0.2667   -0.2667    0.7333
```

۷.۱.۱ ❋ضرب کرونکر

`kron(A,B)` : ضرب کرونکر B,A *

ضرب کرونکر را می‌توانیم بدین صورت بیان کنیم که در آن هر درایه‌ی ماتریس A در همه‌ی ماتریس B ضرب می‌شود. و توجه داریم که با توجه به تعریف این ضرب قابلیت جابجایی ندارد. در بعضی از مباحث پیشرفته استفاده از این دستور می‌تواند بر سرعت و خوانایی برنامه بیافزاید. برای مثال A یک ماتریس 3×2 باشد ضرب کرونکر B,A بصورت زیر است

$$A=\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \qquad A\otimes B=\begin{bmatrix} A_{11}*B & A_{12}*B & A_{13}*B \\ A_{21}*B & A_{22}*B & A_{23}*B \end{bmatrix}$$

که B یک ماتریس است

☑ `kron(ones(n,m),A)` ماتریس A را در n سطر m ستون تکرار می‌کند و معادل دستور repmat(A,n,m) است

```
>> A=[1 1;1 1]
A =
     1     1
     1     1

>> B=[2 3]
B =
     2     3

>> kron(A,B)
ans =
     2     3     2     3
     2     3     2     3

>> kron(B,A)    → kron(A,B) ≠ kron(B,A)
ans =
     2     2     3     3
     2     2     3     3

>> C=[1 2;3 4]
C =
     1     2
     3     4

>> kron(A,C)
ans =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4


>> s=[1 5 8 -2 4]
s =
     1     5     8    -2     4


>> kron(ones(1,3),s)
ans =
 1   5   8  -2   4   1   5   8  -2   4   1   5   8  -2   4
```

→ از این تکنیک برای تکرار به تعداد دلخواه یک آرایه استفاده می‌شود

```
>> kron(s,ones(1,3))      →
از این تکنیک برای چندین تکرار یک عنصر استفاده می‌شود
ans =
 1   1   1   5   5   5   8   8   8  -2  -2  -2   4   4   4
```

`rank(A)` : رنک یک ماتریس تعداد سطرها و یا ستونهایی که از یکدیگر مستقل خطی هستند

```
>> A=[1 0;0 1]
A =
     1     0
     0     1

>> B=[1 2;4 8]
B =
     1     2
     4     8

>> rank(A)
ans =
     2

>> rank(B)
ans =
     1
```

۷.۲ دستگاه معادلات خطی

از ریاضیات مقدماتی می‌دانیم که یک دستگاه معادلات خطی را می‌توانیم بشکل ماتریسی بنویسیم

$$\begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ xn \end{bmatrix} = \begin{bmatrix} b1 \\ \vdots \\ bn \end{bmatrix} \rightarrow Ax=b$$

$x= A^{-1} * b = A\backslash b$

```
>> A=[4 3 -6;4 -2 3;2 4 7]
A =
     4     3    -6
     4    -2     3
     2     4     7

>> b=[0; 2; -8]
b =
     0
     2
    -8

>> det(A)     →  |A|≠0
ans =
  -290

>> x=inv(A)*b  → جواب معادله
x =
    0.2276
   -1.2690
   -0.4828

>> x=A\b    →  این فرم نوشتار ساده‌تر است و خود متلب نیز از آن استفاده می‌کند
x =
    0.2276
   -1.2690
   -0.4828
```

۷.۲.۱ ❋تجزیه‌ی LU

تجزیه‌ی LU یکی از روشهای قوی برای محاسبه‌ی معکوس ماتریس و حل معادله‌ی چند مجهولی ساده است که در آن، ماتریس را به دو ماتریس بالا مثلثی و پایین مثلثی تجزیه می‌کنیم که حاصلضرب آنها برابر ماتریس اصلی است چنین تفکیکی همیشه امکان پذیر نیست و شرایط خاصی را می‌طلبد یک راه حل برای اینکه به این روش مقداری عمومیت دهیم این است که سطرهای ماتریس اصلی را جابجا کرده سپس تجزیه‌ی LU را انجام دهیم.

- `[L1,U] = lu(A)` L1 یک ماتریس پایین مثلثی است که سطرهای آن جابجا شده‌اند L1*U = A

- `[L,U,P] = lu(A)` تجزیه به ماتریس‌های پایین مثلثی و بالا مثلثی در حالت کلی برای این کار باید سطرهای A جابجا شوند ماتریس P این کار را انجام می‌دهد LU = PA

- `[L,U,p] = lu(A,'vector')` p را بصورت برداری ذخیره می‌کند که در کار با ماتریس‌های با ابعاد بالا از ذخیره‌ی داده و سرعت انجام برنامه می‌تواند بسیار مفید باشد

- `triu(A)` : ماتریس A با عناصر مثلث بالا

- `tril (A)` : ماتریس A با عناصر مثلث پایین

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

>> [L1,U] = lu(A)   → سطر ۲ و ۳ مربوط به ماتریس پایین مثلثی جابجا شده‌اند
L1 =
    0.1429    1.0000         0
    0.5714    0.5000    1.0000
    1.0000         0         0

U =
    7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000

>> L1*U
ans =
     1     2     3
     4     5     6
     7     8     0
```

```
>> X = inv(U)*inv(L1)
X =
   -1.7778    0.8889   -0.1111
    1.5556   -0.7778    0.2222
   -0.1111    0.2222   -0.1111


>> [L2,U,P] = lu(A)
L2 =
    1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000

U =
    7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000

P =
     0     0     1
     1     0     0
     0     1     0


>> P*L1
ans =
    1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000

>> P*A - L2*U  → PA = L2U
ans =
     0     0     0
     0     0     0
     0     0     0

>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
```

```
>> tril(A)  → lower triangular
ans =
     8     0     0
     3     5     0
     4     9     2

>> triu(A)  → upper triangular
ans =
     8     1     6
     0     5     7
     0     0     2
```

۷.۲.۲ ❋حل معادله چند مجهولی بروش LU

$Ax=b \Rightarrow LUx = b$

$Ux= L^{-1} * b = L\backslash b$

$x= U^{-1} * (L\backslash b) = U\backslash(L\backslash b)$

```
>> A=[4 3 -6;4 -2 3;2 4 7]
A =
     4     3    -6
     4    -2     3
     2     4     7

>> b=[0; 2; -8]
b =
     0
     2
    -8

>> [L,U]=lu(A)
L =
    1.0000         0         0
    1.0000    1.0000         0
    0.5000   -0.5000    1.0000
U =
    4.0000    3.0000   -6.0000
         0   -5.0000    9.0000
```

```
         0         0   14.5000

>> x1=A^-1*b
x1 =
    0.2276
   -1.2690
   -0.4828


>> x=U\(L\b)
x =
    0.2276
   -1.2690
   -0.4828

>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

>> b=[3; 2; 8]
b =
     3
     2
     8

>> [L,U,p] = lu(A,'vector')
L =
    1.0000         0         0
    0.5000    1.0000         0
    0.3750    0.5441    1.0000
U =
    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
         0         0    5.2941
p =
     1     3     2

>> x1=A^-1*b
x1 =
    0.6639
    0.7056
   -0.5028
```

```
>> x=U\(L\(b(p)))
x =
    0.6639
    0.7056
   -0.5028
```

☑ حل معادلات خطی بروش LU حدود 2.5 برابر از روش ماتریس سریعتر است (سرعت اجرای دستورها را می‌توان با روش‌های مختلف در متلب بدست آورد که در فصل‌های آتی به آن اشاره خواهد شد)

۷.۲.۳ *عبارتهای جبری ماتریسی

در حل معادله‌ای مثل $\frac{dx}{dt} = ax$ نتیجه بسادگی بدست می‌آید $x = e^{at}$ در حالت کلی‌تر اگر x یک بردار و A یک ماتریس باشد نتیجه‌ی مشابهی داریم:

$$\frac{dx}{dt} = A\,x \;\rightarrow\; x = e^{At}$$

برای محاسباتی از این دست به توابع ماتریسی برمی‌خوریم

- $\sqrt{A}$: `sqrtm(A)` رادیکال ماتریسی یعنی عکس A*A در نتیجه خواهیم داشت
- e^{A} `expm(A)`
- $Ln(A)$ `logm(A)`

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
```

```
     4     9     2

>> B=sqrtm(A)
B =
   2.7065 + 0.0601i   0.0185 + 0.5347i   1.1480 - 0.5948i
   0.4703 + 0.0829i   2.0288 + 0.7378i   1.3739 - 0.8207i
   0.6962 - 0.1430i   1.8257 - 1.2725i   1.3511 + 1.4155i

>> D=B*B
D =
   8.0000 + 0.0000i   1.0000 + 0.0000i   6.0000 - 0.0000i
   3.0000 + 0.0000i   5.0000             7.0000 + 0.0000i
   4.0000 - 0.0000i   9.0000 - 0.0000i   2.0000 - 0.0000i

>> B=expm(A)
B =
  1.0e+006 *

    1.0898    1.0896    1.0897
    1.0896    1.0897    1.0897
    1.0896    1.0897    1.0897

>> B1=exp(A)
B1 =
  1.0e+003 *

    2.9810    0.0027    0.4034
    0.0201    0.1484    1.0966
    0.0546    8.1031    0.0074

>> A=[0 -6 -1;6 2 -16;-5 20 -10];
>> x0=[1;1;1];
>> t=2;

>> expm(t*A)*x0
ans =
    0.0035
   -0.0009
   -0.0035

>> D=logm(B)
D =
    8.0000    1.0000    6.0000
    3.0000    5.0000    7.0000
```

```
4.0 9.0000    2.0000
```

۷.۳ مقادیر ویژه و بردارهای ویژه

برای ماتریسی مثل A حل معادله‌ی $|\mathbf{A} - \mathbf{sI}| = 0$ منتج به حل معادله مشخصه ماتریس می‌شود که با حل معادله مشخصه مقادیر ویژه و از آنجا بردارهای ویژه بدست می‌آیند

- `p = poly(A)` : نمایش برداری چند جمله‌ای مشخصه ماتریس A

☑ از tf(p,1) می‌توانیم برای نمایش معمولی چندجمله‌ای برداری p استفاده کنیم این دستور فعلا برای ما تنها ارزش نمایشی دارد.

برای فرم نماشی معمولی می‌توانیم از دستور poly2sym(p) نیز بهره بگیریم ولی این دستور جز در حالتی که ضرایب صحیح باشند جواب چندان روشنی نمی‌دهد و استفاده از آن به مشروط به آشنایی با محاسبات سمبلیک است که در فصلهای بعدی توضیح داده خواهد شد و این جدای از طولانی بودن زمان اجرای آن است.

در خصوص نمایش برداری چندجمله‌ای در مبحث چندجمله‌ای‌ها بطور مبسوط بحث خواهد شد

- [V,D] = eigs(A) : مقادیر ویژه و بردارهای ویژه‌ی A ، ماتریس D یک ماتریس قطری است که عناصر قطر اصلی آن مقادیر ویژه ماتریس و ماتریس V نیز ماتریسی است که ستون‌های آن بردارهای ویژه‌ی ماتریس A می‌باشند

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

>> p=poly(A)→ نمایش برداری چند جمله‌ای ضرایب از توان بزرگتر به توان کوچک (توان صفر)
p =
    1.0000  -15.0000  -18.0000   -0.0000

>> tf(p,1)
Transfer function:
s^3 - 15 s^2 - 18 s - 2.347e-014
```

tf() یکی از دستورهای پایه‌ای متلب در کنترل می‌باشد در اینجا از آن برای نمایش چند جمله‌ای استفاده می‌کنیم

```
>> d=eig(A)
d =
   16.1168
   -1.1168
   -0.0000

>> [V,D] = eig(A)
V =
   -0.2320   -0.7858    0.4082
   -0.5253   -0.0868   -0.8165
   -0.8187    0.6123    0.4082
D =
   16.1168         0         0
         0   -1.1168         0
         0         0   -0.0000
>> d1=diag(D)
d1 =
   16.1168
   -1.1168
   -0.0000

*>> G=inv(V)*A*V    →
```

تبدیل تشابهی : ماتریس A را به فضای با پایه‌های بردارهای ویژه منتقل می‌کند

```
G =
   16.1168    0.0000    0.0000
   -0.0000   -1.1168    0.0000
    0.0000    0.0000    0.0000

*>> sum(sum(abs(G-D)))     → G=D
ans =
  1.4780e-014
```