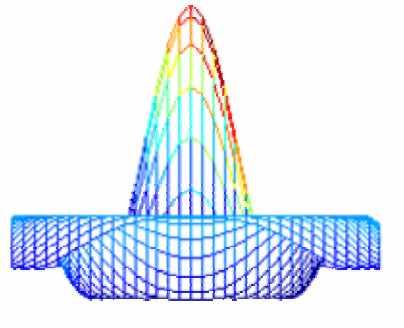


کاربرد کامپیوتر در مهندسی

مصطفی همت‌آبادی

پائیز ۱۳۸۳

بخش یک MATLAB

فصل ۱ کلیات و اصول

مقدمات

واژه MATLAB

کلمه فوق، سر- واژه‌ی عبارت MATrix LABoratory است، که در این متن به‌شکل MATLAB یا متلب می‌آید. برای اجرای مثال‌ها و تمرین‌های این متن از MATLAB 6.5, Release 13 استفاده شده، و استفاده از آن (یا ویراست‌های بالاتر) به دانشجویان توصیه می‌شود.

سیمای MATLAB

MATLAB زبانی است که کاربرد کامپیوتر در مهندسی را با کارائی بالا تضمین کرده و امکانات محاسباتی، تصویری، و برنامه‌نویسی را در محیطی آسان و آشنا فراهم می‌کند. کارائی MATLAB در مقوله‌هائی نظیر: محاسبات ریاضی، دسترسی به/ و آنالیز داده‌ها، مُدل‌سازی و شبیه‌سازی، گرافیک، و تولید نرم‌افزار (حتی برای محیط ویندوز) به اثبات رسیده است. این زبان با توجه به نظرات کاربران دانشگاهی و صنعتی دست‌خوش بازنگری‌های زیادی شده و اکنون به زبان استاندارد جهت آموزش‌های مقدماتی و عالی و ابزار پژوهش و توسعه در صنایع تبدیل شده است.

MATLAB جعبه‌ابزارهائی برای کاربردهای خاص در اختیار قرار می‌دهد، که از جمله آن‌ها جعبه‌ابزار ریاضیات، کنترل، شبکه‌های عصبی، بازرگانی، . . . می‌باشند. جعبه ابزارها با زبان متلب و به صورت مجموعه‌ای از ام- فایل‌ها گسترش یافته‌اند و برای هر کاربر در زمینه تخصصی‌اش کاربرد و اهمیت زیاد دارند. امکان ساخت جعبه‌ابزارهای جدید و شخصی نیز برای کاربران پیشرفته فراهم است.

در این نرم‌افزار هر متغیر به عنوان یک ماتریس یا یک بردار (بردار ماتریس تک سطری یا تک ستونی است) شناخته می‌شود. لذا تعدادی مقدار را یک جا می‌توان به یک متغیرِ تک- نام (بدون نیاز به اعلام قبلی تعداد اعضاء) نسبت داد. این ابتکار ما را از مقدار دهی به تک تک عناصر آرایه که در زبان‌های برنامه نویسی معمولاً از طریق حلقه‌های تکرار انجام می‌شود بی‌نیاز می‌کند. از طرف دیگر برای رسم ترسیمات، یک متغیر برداری می‌تواند به تنهائی مقادیر روی یک محور را پوشش دهد. هم‌چنین می‌توان توابعی با چندین خروجی (خروجی برداری) تعریف کرد.

متلب دارای پنج ویژگی شایان ذکر است:

۱- پنجره‌ی واسط کاربر Integrated Development Environment (IDE) بسیار دل‌پذیر و دست‌یافتنی که از امتیازات برنامه‌نویسی متنی و گرافیکی هر دو استفاده می‌کند. این واسط کاربر شامل پنجره‌های: فرمان، دیرکتوری جاری، تاریخچه فرمان، فضای کار، . . . است. پنجره‌ی فرمان دستورات را به صورت کنسول یا خط فرمان، مشابه برنامه‌نویسی DOS، دریافت و اجرا می‌کند. پنجره فضای کار کلیه متغیرهای فضای کار را با مشخصات آن‌ها نمایش می‌دهد. پنجره‌های واسط کاربر دینامیک بوده و ممکن است خود شامل پنجره‌های فرعی باشند.

۲- کتاب‌خانه‌ی عظیمی از توابع مقدماتی و پیشرفته.

۳- زبان قوی هم برای فرامین کوتاه و یک‌بار مصرف و هم برای برنامه‌نویسی بزرگ و کاربردی.

٤- روش‌های متعدد ترسیمات دوبعدی و سه‌بعدی.

۵- واسط میان‌برنامه‌ای Application Program Interface (API) که امکان فراخوانی برنامه‌های متلب از زبان‌های C و Fortran ، تبدیل فایل‌های متلب (ام- فایل‌ها) به زبان C ، و استفاده از موتور محاسباتی متلب در این زبان‌ها را فراهم می‌کند. با اسفاده از تبدیلات فوق می‌توان برنامه‌های نوشته شده

در محیط متلب را به صورت اجرائیِ کنسولی (قابل اجرا از خط فرمان سیستم عامل) درآورد. هم‌چنین امکان تهیه فایل‌های اجرائی با واسط گرافیکی برای ویندوز وجود دارد.

باید به پنج مورد فوق که توسط شرکت Mathworks (سازنده متلب) به عنوان برجستگی‌های متلب ذکر شده‌اند، دارا بودن راهنمای جامع و کاملِ کنارِ دست و وجود مدارک راهنمای قابل چاپ به صورت PDF را افزود.

روش این کتاب

روش کار ما در این متن یادگیری- با- مثال Learn by Examples است. ازاین‌رو اجرای متلب، هم‌زمان با مطالعه متن اکیداً توصیه می‌شود. در مثال‌های کاربردی نوع کاربرد را ذکر کرده‌ایم. در کلیه مثال‌ها، دستورات فوری که روی پنجره فرمان نوشته می‌شوند، بعداز علامت << (نشانه سطر فرمان Command Prompt) آمده‌اند، و در انتهای هر دستور فوری بایستی کلید <Enter> زده شود. نتایجی که روی پنجره فرمان ظاهر می‌شوند بدون این علامت << آمده‌اند. اگر در پایان یک دستور علامت سمی‌کالن (نقطه- ویرگول) قرار دهیم نتیجه اجرا پس از زدن کلید <Enter> در پنجره فرمان ظاهر نمی‌شود، وگرنه نتیجه را در پنجره فرمان خواهیم دید. علامت % برای توضیحات می‌آید و آن‌چه بعداز آن نوشته شود جزو دستورات منظور نمی‌شود.

خروجی‌های گرافیکی در پنجره گرافیک ظاهر می‌شوند. سطرهائی که در ادیتور برنامه‌نویسی به صورت ام- فایل نوشته می‌شوند بدون این علامت << آمده‌اند.

تمرین‌های پایان دروس از نکات برجسته متن گل‌چین شده، نقش خودآزما را به عهده دارند، و انجام آن‌ها ضروری است.

دورچین ها

در این متن برای تفکیک قسمت‌ها و ایجاد خوانائی بیشتر از دورچین‌های زیر استفاده شده است:

نتیجه‌ی عملیات پنجره فرمان یا یک برنامه

راهنما:

- نکات آموزشی داخل مثال‌ها یا جزئیاتی که در متن ذکر نشده‌اند

دقت کنید:

مواردی که نیاز به تأکید بیشتر دارند و جلب توجه به موارد مهم

تمرین

مؤلفه های اصلی متلب

ماتریس Matrix

A که یک ماتریس ۲×۳ است نمونه‌ای از یک ماتریس دو بُعدیِ m × n عددی است که در جبر این‌گونه نوشته می‌شود:

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

مثال:

ماتریس فوق را در متلب تعریف و بُعد آن را با تابع ndims () تعیین کنید.

حل:

```
>> A = [1 2 3; 4 5 6]
A = 1 2 3
    4 5 6
>> n = ndims(A)
   n = 2
```

راهنما:

- عمده‌ی دستورات متلب در پنجره فرمان یا در ام- فایل می‌آیند. (شرح ام- فایل بعداً می‌آید). علامت >> نشانه سطر فرمان Command Prompt است. هر دستور در پنجره فرمان با فشردن کلید <Enter> انجام می‌شود.
- دستور اول را می‌توان با شکستن سطر به صورت زیر نوشت:

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

مثال:

عنصر ردیف۲ ستون۳ و عضوِ (عنصر) ردیف۱ ستون۲ ماتریس A را چاپ کنید.

حل:

```
>> a = A(2,3)
   a = 6
>> A(1,2)
   ans = 2
```

راهنما:

- متلب مابین حروف بزرگ و کوچک انگلیسی فرق می‌گذارد. به عبارت دیگراین زبان Case Sensitive است.
- همیشه آخرین مقدار حساب شده در متغیر داخلی ans قرار می‌گیرد، مگر این‌که قبلاً برای آن متغیری تعریف شده باشد.

مثال:

ستون‌های ۲ و ۳ از ردیف‌های ۱ و ۲ ماتریس A را در یک ماتریس به نام A1 قرار دهید.

حل:

```
>> A1 = A(1:2,2:3)
A1 = 2     3
     5     6
```

راهنما:

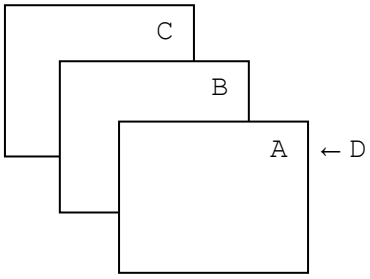
برای استخراج قسمتی از یک ماتریس ابتدا ردیف‌ها و سپس ستون‌های مستخرجه از آن‌ردیف‌ها را (به‌سبک فوق) می-نویسیم.

در مورد ماتریس‌ها بدانیم

- ماتریس سه بُعدی نیز وجود دارد.
- ماتریس عددی با اعداد مختلط نیز وجود دارد.
- ماتریس کاراکتری، ماتریس نمادین و ماتریس سلولی (با عناصری از نوع‌های مختلف) نیز وجود دارند.
- ماتریسی که دارای یک ردیف باشد، بردار ردیفی، و ماتریسی که دارای یک ستون باشد بردار ستونی نام دارند. در متلب بردارها کاربرد زیادی دارند. معمولاً کلمه بردار به تنهائی برای بردار ردیفی به کار می‌رود.
- متلب ترجیح می‌دهد با ماتریس یا بردار کار کند، لذا یک عدد (یا سکالر) را، ماتریس عددیِ 1×1 نام می‌دهد.

ماتریس سه بعدی (فضائی)

یک ماتریس سه بعدیِ $m \times n \times p$ دارای p صفحه است که هر صفحه یک ماتریس دو بعدی $m \times n$ است. انتخاب مقادیر m, n, p دلخواه است، اما باید دقت کرد که n و m در تمامی صفحات یکسان باشند. در شکل زیر شمای فضائی ماتریس D نشان داده شده که در آن p = 3 است:



صفحات ۱، ۲، و ۳ ماتریس فضائی D به این صورت تعریف می‌شوند:

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

A و B و C ماتریس‌های دو بعدی هستند که قبلاً تعریف شده‌اند.

راهنما:

ماتریس‌های دو بعدی که صفحات ماتریس سه بعدی را تشکیل می‌دهند باید هم‌ردیف و هم‌ستون باشند.

مثال:

یک ماتریس فضائی با سه صفحه تولید کنید. تعداد ردیف و ستونِ هر صفحه، و تعداد صفحه‌های آنرا با تابع داخلی size() نمایش دهید. بُعد آن را با استفاده از تابع داخلی ndims() تعیین کنید.

حل:

```
>> A = [2 -5; 16 7.6; 13.3 -9];
>> B = [-2 -5.2; 6 7.3; 5 1.4 ];
>> C = [-2 2 ; 7 3; 1.55 1.4 ];
>> D(:,:,1) = A;
>> D(:,:,2) = B;
>> D(:,:,3) = C
```

```
D(:,:,1) =
    2.0000   -5.0000
   16.0000    7.6000
   13.3000   -9.0000
D(:,:,2) =
   -2.0000   -5.2000
    6.0000    7.3000
    5.0000    1.4000
```

```
D(:,:,3) =
   -2.0000    2.0000
    7.0000    3.0000
    1.5500    1.4000
>> size(D)
   ans =   3     2     3
>> ndims(D)
   ans = 3
>> length(size(D))
   ans = 3
```

راهنما:

- `size()` تعداد ردیف و تعداد ستون هر صفحه، و تعداد صفحه‌های ماتریس فضائی را نشان می‌دهد.
- `ndims()` طول یا تعداد عناصر `size()` را می‌دهد، و معادل است با `length(size())`.

بردار

بردار، یک ماتریس تک سطری (ردیفی) یا تک ستونی است که دارای تعدادی عضو (معمولاً مقادیر عددی) بوده و تحت یک نام واحد شناخته می‌شود. نام بردار و مقدار آن همزمان در خط فرمان نوشته می‌شوند و برخلاف زبان‌های دیگر نیاز به اعلام قبلی نام و تعداد اعضاء وجود ندارد. مفهوم بردار در متلب ما را از مقدار دهی به تک تک عناصر آرایه که در زبان‌های برنامه نویسی معمولاً از طریق حلقه‌های تکرار انجام می‌شود بی‌نیاز می‌کند. از طرف دیگر برای رسم ترسیمات، یک متغیر برداری می‌تواند به تنهائی مقادیر روی یک محور را پوشش دهد.

مثال:

یک بردار ردیفی ٤ عضوی (٤ عنصری) تولید کنید. اندازه، طول و بُعد آن‌را با توابع داخلی `size()` و `length()` و `ndims()` چاپ کنید.

حل:

```
>> rv = [2 -5 16 7.6];
>> size(rv)
   ans = 1     4
>> L = length(rv)
   L = 4
>> m = ndims(rv)
   m = 2
```

راهنما:

- اگر در پایان یک سطر علامت سمی‌کالن (نقطه- ویرگول) قرار دهیم نتیجه اجرا در پنجره فرمان ظاهر نمی‌شود.
- توابع `ndims()`, `size()`, `length()` و نظائرشان را که در سنجش کمیت ماتریس‌ها به کار می‌روند، در این متن توابع اندازه‌گیر نامیده‌ایم.
- تابع `length()` معادل `max(size())` و تابع `ndims()` معادل `length(size())` هستند.

دقت کنید:

- اندازه `size` و نوع `type` هر آرایه در پنجره `Workspace` نشان داده می‌شود.
- در پنجره `Workspace` هر بردار ردیفی یک ماتریس دو بعدی `1×n` (تک‌ردیفی) و هر بردار ستونی یک ماتریس دو بعدی `m×1` (تک‌ستونی) است. پس یکی از ابعاد هر نوع بردار یک است.

فصل ۲ برخی تعاریف مقدماتی

آرایه، عملیات آرایه ای

همان‌گونه که در سایر زبان‌های برنامه‌نویسی تعریف می‌شود، آرایه یک نام کلی برای انواع ماتریس، بردار و کلاً مجموعه‌های یک تا چندین عنصری است. این عناصر در تعدادی ردیف، ستون، و صفحه قرار گرفته‌اند. در این متن کلمه آرایه اعم از کلمات ماتریس، بردار، و متغیر به کار می‌رود، و کلمات متغیر و آرایه نیز به جای هم به کار می‌روند.

در متلب متغیرِ تک‌مقداری آرایه یک عنصری نامیده می‌شود.

هم‌چنین عملیاتی که بر روی تک‌تک عناصر آرایه انجام شود، عملیات آرایه‌ای نام دارد که در مقابل عملیات ماتریسی قرار می‌گیرد (شرح بیشتر بعداً می‌آید).

ترسیمات

کار ترسیم توابع داخلی متلب و توابع تعریف شده توسط کاربر بسیار آسان است. مثال زیر اصول کار را توضیح می‌دهد، شرح بیشتر در دروس بعد می‌آید.

مثال:

معادله یک سهمی را رسم کنید.

حل:

ابتدا یک آرایه شامل مقادیر x تعریف می‌کنیم:

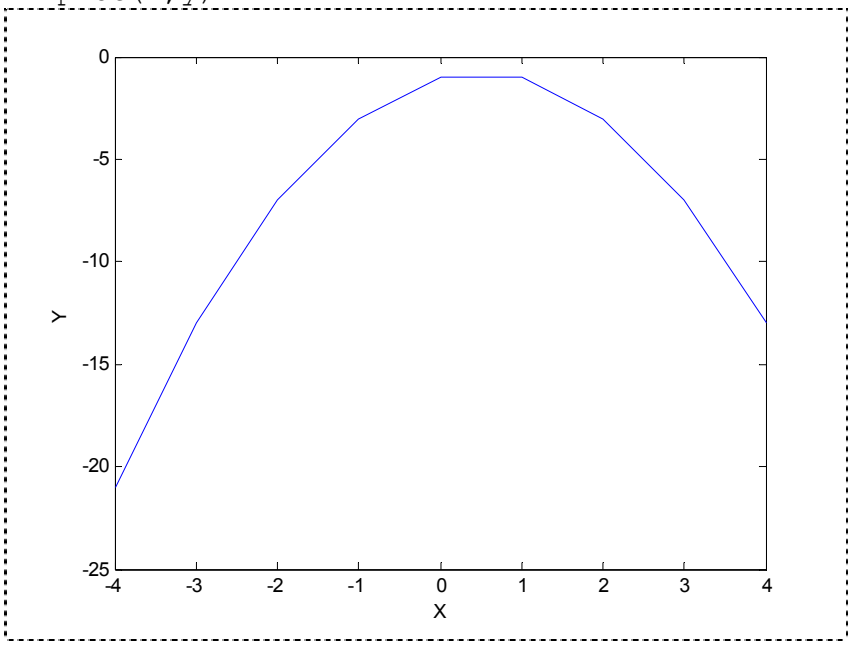
```
>> x = [-4 -3 -2 -1 0 1 2 3 4];
```

سپس تابع مورد نظر را مینویسیم:

```
>> y = -x.^2 + x - 1;
```

و با یک دستور ساده y را بر حسب x رسم می‌کنیم:

```
>> plot(x,y)
```



راهنما:

برای به توان رساندن تک‌تک عناصر آرایه (توان آرایه‌ای) از علامت ^ . استفاده می‌شود. شرح بیشتر عملیات آرایه‌ای بعداً می‌آید.

دقت کنید:

دستور plot(x,y) سطح x-y را به تعداد عناصر موجود در آرایه‌های x و y نقطه‌گذاری کرده سپس نقاط را به هم وصل می‌کند، لذا برای داشتن منحنی بدون شکست باید نقاط بیشتری را به x و نتیجتاً به y نسبت داد.

توابع کتاب خانه ای

یکی از قابلیت‌های مهم متلب داشتن تعداد معتنابهی توابع توساخت built in functions یا توابع داخلی یا توابع کتاب‌خانه‌ای است، که نیازهای گوناگون انواع کاربران را عمدتاً برآورده می‌سازند. معرفی چند تابع در زیر آمده:

تولید عدد تصادفی

تابع rand(m,n,p) یک ماتریس تصادفی p صفحه‌ای با صفحاتی دارای m ردیف و n ستون تولید می‌کند.

مثال:

یک ماتریس تصادفی دو صفحه‌ای با صفحات ۲×۳ تولید کنید.

حل:

```
>> B = rand(2,3,2)
B(:,:,1) = 0.9218    0.1763    0.9355
           0.7382    0.4057    0.9169
B(:,:,2) = 0.4103    0.0579    0.8132
           0.8936    0.3529    0.0099
```

مثال:

یک بردار عددی با ٦ عنصر تصادفی تولید کنید.

حل:

```
>> d1 = rand(1,6)
d1 = 0.9901    0.7889    0.4387    0.4983    0.2140    0.6435
```

دقت کنید:

در مثال فوق ماتریسی با p = 1 (ماتریس دو بعدی، تک صفحه) m = 1 (تک ردیف)
n = 6 (6 ستون) تولید می‌شود.

مثال:

یک بردار عددی با ٦ عنصر تصادفی تولید کنید، که بزرگای هر عضو آن بین ۱ تا ۱۰ باشد.

حل:

```
>> rd = round(9 * rand(1, 6))+ 1
rd = 4    10     8     5     8     3
```

راهنما:

برای ایجاد N عدد تصادفی بین MinN و MaxN از فرمول زیر استفاده می‌کنیم:

```
rd = round((MaxN - MinN) * rand(1, N))+ MinN
```

تولید آرایه با تکرار آرایه دیگر

تابع B = repmat(A,M,N) آرایه A را M بار افقی و N بار عمودی تکرار کرده و در آرایه B قرار می‌دهد.

مثال:

یک بردار با ده عضو متناوب صفر ویک تولید کنید.

حل:

```
>> A = [1 0];
>> B = repmat(A, 1, 5)
B = 1     0     1     0     1     0     1     0     1     0
```

ماتریس جادوئی

دستور magic (n) یک مربع جادوئی n*n می‌سازد. خاصیت مربع جادوئی این است که حاصل‌جمع عناصر آن در طولِ ردیف، قطر، و ستون برابرند.

مثال:

یک مربع جادوئی ۲×۲ بسازید.

حل:

```
>> Mg = magic(2)
Mg = 1     3
     4     2
```

مثال:

یک ماتریس ٤×٤ به نام Mg با تکرار ماتریس جادوئی ۲×۲ بسازید.

حل:

```
Mg = repmat(magic(2),2,2)
Mg = 1     3     1     3
     4     2     4     2
     1     3     1     3
     4     2     4     2
```

آرایه منطقی

تابع logical () ، یک بردار با اعضاء عددی را به یک بردار با اعضاء صفر و یکِ منطقی تبدیل می‌کند. منطقی بودن یک آرایه با تابع islogical () امتحان می‌شود. متغیر منطقی فقط دو مقدار درستی (منطق یک) و نادرستی (منطق صفر) را می‌گیرد، که از لحاظ نوع type با 0 و 1 عددی متفاوت هستند. مقادیر منطقی یک بایت از حافظه را اشغال می‌کنند، در حالی‌که مقادیر عددی (از نوع double که پیش‌فرض متلب است) هشت بایت جا می‌گیرند.

مثال:

تعداد بایت اشغالی عددِ یک و منطقِ یک را در حافظه، روی پنجره فضای کار Workspace مشاهده کنید.

حل:

```
>> ad = 1
ad = 1
>> ag = logical(1)
ag = 1
```

Workspace

Name	Size	Bytes	Class
ad	1x1	8	double array
ag	1x1	1	logical array

مثال:

از بردار B یک بردار منطقی ایجاد کنید. منطقی بودن B و BL را با تابع islogical () امتحان کنید.

حل:

```
>> B = [1 0 1 0 1 0 1 0 1 0];
>> BL = logical(B)
BL = 1     0     1     0     1     0     1     0     1     0
>> islogical(B)
   ans = 0
>> islogical(BL)
   ans = 1
```

راهنما:

- اگرچه آرایه‌های عددی با اعضاء غیر از صفر و یک را می‌توان به آرایه منطقی تبدیل کرد، اما توصیه می‌شود فقط آرایه‌هائی با اعضاء صفر و یکِ عددی به آرایه منطقی تبدیل شوند.
- توابعی که صحّت یا سُقمِ امری را امتحان می‌کنند، در صورت صحت، منطق یک و در صورت کذب، منطق صفر برمی‌گردانند. معمولاً این گونه توابع با `is...` شروع می‌شوند.

حذف بعضی عناصر آرایه

اگر بردار منطقی را اندیس بردار عددی قرار دهیم (اندیس‌گذاری منطقی `Logical Indexing`) فقط عناصر متناظر با یک‌های منطقی باقی می‌مانند.

مثال:

بردار `Li = [2 4 5 6 8 10 11 12]` و یک بردار منطقی به نام `Lg` با صفرهای متناظر با عناصر زوج `Li` تولید کنید. با استفاده از بردار `Lg` عناصر غیر زوج `Li` را حذف کنید.

حل:

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 0 1 1 1 0 1]);
>> LiEv = Li(Lg) % Logical Indexing
LiEv = 2     4     6     8    10    12
```

مثال:

سه عنصر اول بردار `Li` را با استفاده از بردار `Lg` نگه‌داری و بقیه عناصر آن را حذف کنید.

حل:

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 1]);
>> LiEv = Li(Lg)
LiEv = 2     4     5
```

راهنما:

عناصر بردار اندیس می‌تواند از عناصر بردار عددی کم‌تر باشد، اما بیش‌تر نمی‌تواند باشد.

ضرب بردار عددی در بردار منطقی

حاصل یک عمل ریاضی بین دو نوع مختلف از نوع برتر خواهد بود (همانند C++).

مثال:

دو بردار `Li` و `Lg` را در هم ضرب و نوع حاصل‌ضرب را تحقیق کنید.

حل:

```
>> Li = [2 4 5 6 8 10 11 12];
>> Lg = logical([1 1 0 1 1 1 0 1]);
>> Lip = Li .* Lg
   Lip =  2     4     0     6     8    10     0    12
>> islogical(Lip)
   ans = 0
>> isnumeric(Lip)
   ans = 1
```

راهنما:

- برای ضرب عنصر به عنصر، علامت ضرب آرایه‌ای `.*` به کار می‌رود.
- برای ضرب عنصر به عنصر، دو آرایه باید متساوی‌العنصر باشند.
- نوع عددی از نوع منطقی برتر است (چون بایت بیشتری را اشغال می‌کند؟).

تمرین

۱- یک بردار عددی به نام Rv با ۵ عنصر تعریف کنید.

۲- توابع `ndims()`, `size()`, `length` را در مورد بردارتمرین ۱ اجرا کنید

۳- یک بردار منطقی به نام L1 با ۵ عنصر بسازید.

٤- بردار منطقی تمرین ۳ را: الف) اندیس Rv قرار دهید. ب) در Rv ضرب کنید.

۵- * دو ماتریس عددی ۶×۳ به نام‌های M1, M2 با مربع جادوئی بسازید.

٦- توابع `ndims()`, `size()`, `length` را در مورد ماتریس‌های تمرین ۵ اجرا کنید.

۷- با ماتریس‌های تمرین ٤ یک ماتریس سه بعدی را به نام Qm تعریف کنید و نمایش دهید.

۸- توابع `ndims()`, `size()`, `length` را در مورد ماتریس تمرین ۷ اجرا کنید.

۹- یک بردار عددیِ ستونی به نام Rh با ۵ عنصر تعریف کنید. تابع `y = Rh.^2` را رسم کنید.

۱۰- توابع `ndims()`, `size()`, `length` را در مورد بردارتمرین ۹ اجرا کنید.

۱۱- نوع، اندازه، و تعداد بایت آرایه‌های تولید شده را در پنجره Workspace ملاحظه و بررسی کنید.

فصل ۳ متغیرها

آشنائی با متغیرهای متلب

در متلب هر آرایه یک متغیر محسوب می‌شود. متغیر به معنای متعارف برنامه نویسی آرایه‌ی تک عنصری است.

بعضی از پیش- تعریف ها

بعضی از متغیرهای داخلی متلب در جدول زیر آمده‌اند:

نام	تعریف	مقدار (بر روی کامپیوتر من)
ans	متغیر داخلی که آخرین مقدار حساب شده را نگه می‌دارد	آخرین مقدار حساب شده
eps	فاصله بین 1 و بزرگ‌ترین عدد اعشاری بعد از آن	pow2(1,-52) or 2^(-52) 2.220446049250313e-016
realmax	بزرگ‌ترین عدد ممکن بر روی کامپیوتر شما	1.797693134862316e+308
realmin	کوچک‌ترین عدد ممکن بر روی کامپیوتر شما	2.225073858507201e-308
pi	عدد پی	4*atan(1) or imag(log(-1)) 3.141592653589793e+000
Inf, inf	Infinity	نتیجه تقسیم عدد بر صفر
NaN,nan	Not a Number	تقسیم inf بر inf و صفر بر صفر

تابع pow2(m,n) معادل است با $m \times 2^n$.

مثال:

با فرمت مناسب مقدار eps - 1 را نمایش دهید.

حل:

```
>> format long e
>> a = 1 - eps
a = 9.999999999999998e-001
```

راهنما:

دستور format long e خروجی را با تعداد ارقام زیاد و به صورت نمائی چاپ می‌کند.

مثال:

با فرمت مناسب مقدار eps و realmin + eps را نمایش دهید.

حل:

```
>> eps
ans = 2.220446049250313e-016
>> realmin + eps
ans = 2.220446049250313e-016
```

دقت کنید:

هر دو مقدار مساوی هستند، لذا نتیجه می‌گیریم که کوچک‌ترین عددی که ارزش محاسباتی دارد eps است.

مثال:

مقدار عدد پی را با نوشتن خودش و نوشتن فرمول‌ها چاپ کنید.

حل:

```
>> pi
ans = 3.141592653589793e+000
>> 4*atan(1)
ans = 3.141592653589793e+000
>> imag(log(-1))
ans = 3.141592653589793e+000
```

فضای حافظه Workspace

مشخصه‌های نام، نوع (عددی، منطقی، نمادین، رشته‌ای، ...)، اندازه (تعداد ردیف و ستون)، و تعداد بایتِ متغیر(آرایه)های تعریف شده، در پنجره Workspace نمایش داده می‌شوند. لذا توصیه می‌شود همواره ار انتخاب View از منیوی اصلی، پنجره Workspace را تیک بزنید تا بتوانید مشخصه‌های متغیرهای خود را در فضای حافظه مشاهده نمائید. اگر به روی یک متغیر در پنجره Workspace کلیک-کلیک کنید، دست‌رسی بیش‌تری به‌آن پیدا کرده، حتی می‌توانید آن را ادیت کنید (امتحان کنید).

اعلام نوع متغیر

اصولاً متلب نیازی به اعلام یا معرفیِ declaration متغیرها قبل از کاربرد آن‌ها ندارد (بر خلاف C++)، بلکه به هنگام مقدار دهی به یک متغیر، نوع آن خودبه‌خود تعیین می‌شود.

میزان دقت متغیرهای عددی

پیش‌فرض متلب این گونه است که هر متغیر عددی را به هنگام ورود (یا معرفی) آن، یعنی اعلام و مقدار دهیِ هم‌زمان، با دقت افزوده (مضاعف) double نگه‌داری می‌کند. هر عدد با دقت افزوده double ۸ بایت از حافظه و با دقت محدود single ٤ بایت از حافظه را می‌گیرد. چند نوع عدد صحیح integer نیز در متلب وجود دارند. هر عدد صحیحِ int16 شانزده بیت یا دو بایت حافظه اشغال می‌کند. هم‌چنین توابعی برای تغییر نوع متغیرهای عددی داریم، برای اطلاع بیشتر از انواع متغیرها help datatypes را اجرا کنید.

مثال:

متغیر عددی x = 45.678894673823456789 را وارد کرده و با دو فرمت بلند و کوتاه نمایش دهید

حل:

```
>> format long
>> x = 45.678894673823456789
x = 45.67889467382346
>> format short
>> x
x = 45.6789
```

راهنما:

- فرمت بلند تا ١٤ رقم بعد از نقطه- اعشار را نشان می‌دهد.
- فرمت کوتاه عدد را به ٤ رقم بعد از ممیز گرد کرده و نشان می‌دهد.

مثال:

متغیر x مثال فوق را با دقت محدود در y قرار داده و با هر دو فرمت فوق مشاهده کنید.

حل:

```
>> format long
>> y = single(x)
   y = 45.67889404296875
>> format
>> y
   y = 45.6789
```

راهنما:

اجرای format با اجرای format short معادل است

مثال:

متغیر x مثال فوق را به صورت عدد صحیح در z قرار داده و z را مشاهده کنید.

حل:

```
>> z = int16(x)
z = 45
```

راهنما:

تابع int16() قسمت صحیح عدد اعشاری را برمی‌گرداند.

مثال:

عدد گنگِ π را با هر دو فرمت فوق و فرمت کسری format rational نمایش دهید.

حل:

```
>> format long
>> pi
   ans = 3.14159265358979
>> format
>> pi
   ans = 3.1416
format rational
>> pi
   ans = 355/113
```

راهنما:

pi یک تابع توساخت built-in است، که در داخل متلب با دقت افزوده وجود دارد.

دقت کنید:

به تعداد بایت متغیرها در پنجره Workspace .

دستورهائی در مورد متغیرها

نام دستور	نتیجه اجرا
`who`	نام متغیرها را می‌دهد
`whos`	نام و اطلاعات بیشتری از مشخصه‌های متغیرها می‌دهد
`clear var`	یک متغیر را پاک می‌کند
`clear`	کلیه متغیرهای تعریف شده را پاک می‌کند
`clear all`	حافظه را کلاً پاک می‌کند
`pack`	حافظه را منظم و جمع و جور می‌کند

چند راهنمائی نوشتاری

نوشتن چند دستور در یک سطر

مثال:

```
>> x = 3, y = 34; z = y ^ x
x = 3
z = 39304
```

دقت کنید:

چون در مقابل y سمی‌کالن (نقطه- ویرگول) آمده مقدار آن نشان داده نشده است.

شکستن یک دستور در چند سطر

مثال:

```
>> rv1 = [1.4 5.436 -17 ...
       3.2 -12.28 0]
```

قطع اجرا

هرگاه اجرای دستورات در پنجره فرمان طولانی شد، در صورت تمایل برای قطع اجرا <Ctrl+C> را بزنید.

ضبط دستورات قبلی

دستورات قبلی که در پنجره فرمان نوشته شده‌اند، حتی در اجراهای بعدی متلب باقی می‌مانند، و تکرار آن‌ها با کلیدهای ↑ و ↓ انجام میشود. برای اجرای یک سطر تکراری کلید <Entar> و برای حذف یک سطر تکراری نادل‌خواه کلید <Esc> یا در ابتدای سطر کلید <Ctrl+K> را بزنید. پنجره Command History هم فرمان‌های قبلی را نگه می‌دارد. اگر سطر فرمانی در هریک از پنجره‌های فرمان یا تاریخچه فرمان یا ادیتور ام- فایل مارک شود، با راست کلیک و انتخاب Evaluate Selection آن دستور اجرا خواهد شد.

تمرین

۱- با نحوه نمایش `format long`، و با تکرار تعیین کنید `n` چه عدد صحیحی باشد که رقم یک در اعداد بعداز ممیز حاصلِ عبارت `1 + n*eps` ظاهر شود (به این شکل نشان داده شود `1.00000000000001`)

۲- متغیر `x = 42235.62567889404296875` را با تابع `single(x)` به تابع یکا تبدیل و با فرمت بلند، کوتاه و فرمت بانک `format bank` نمایش دهید.

۳- توابع `realmax` و `realmin` را امتحان کنید. عبارات `type realmax` و `type realmin` را اجرا و بررسی کنید.

٤- متغیر `x` تمرین ۲ را تقسیم بر صفر کرده برابر `y` قرار دهید.

٥- متغیر `t` را مساوی صفر تعریف کرده `f = sin(t)/t` را به دست آورید.

٦- عدد گنگِ `e` را با فرمول `e = exp(1)` (طرف راست، e^1 است) به دست آورده با هر دو فرمت کوتاه و بلند و فرمت کسری `format rational` نمایش دهید. **دقت کنید:** `e` یک متغیر داخلی نیست، بلکه توسط خودمان تعریف شده است.

۷- دستورات `who, whos, clear var, clear` را برای متغیرهای تعریف شده خودتان امتحان کنید.