

کاربرد کامپیوتر در مهندسی

مصطفی همت آبادی

پائیز ۱۳۸۳

MATLAB بخش یک

بخش یک MATLAB II

فصل ۱ کلیات و اصول

۱. مقدمات

۱. واژه MATLAB

۱. سیمای MATLAB

۲. روش این کتاب

۲. دورچین ها

۲. مؤلفه های اصلی متلب

۲. ماتریس Matrix

۴. ماتریس سه بعدی (فضائی)

۵. بردار

فصل ۲ برخی تعاریف مقدماتی

۶. آرایه، عملیات آرایه ای

۶. ترسیمات

۷. توابع کتاب خانه ای

۷. تولید عدد تصادفی

۷. تولید آرایه با تکرار آرایه دیگر

۸. ماتریس جادویی

۸. آرایه منطقی

۹. حذف بعضی عناصر آرایه

۹. ضرب بردار عددی در بردار منطقی

۱۰. تمرین

فصل ۳ متغیرها

۱۱. آشنائی با متغیرهای متلب

۱۱. بعضی از پیش-تعریف ها

۱۲. فضای حافظه Workspace

۱۲. اعلام نوع متغیر

۱۲. میزان دقت متغیرهای عددی

۱۴دستورهای در مورد متغیرها
۱۴چند راهنمایی نوشتاری
۱۴نوشتن چند دستور در یک سطر
۱۴شکستن یک دستور در چند سطر
۱۴قطع اجرا
۱۴ضبط دستورات قبلی

تمرین

فصل ۴ دستورها و توابع ورودی خروجی ۱۶

۱۶دریافت ورودی
۱۶دستور input()
۱۶ارسال خروجی
۱۶ارسال خروجی به صفحه نمایش، دستور disp(var)
۱۷ارسال خروجی به صفحه نمایش، دستور fprintf()
۱۸دستور فرمت format
۱۸مثال: نمایش اعداد در مبنای شانزده
۱۸ضرب در نمایش اعداد
۱۹ارسال خروجی به فایل و درگاه
۱۹مثال: ایجاد یک فایل متن TXT
۱۹مثال: باز کردن درگاه port

توابع زمانی ۲۰

۲۰مثال: تعیین فاصله زمانی
۲۰مثال: نمایش تاریخ روز و ساعت
۲۰مثال: نمایش تاریخ

دستورات پنجره فرمان ۲۰

۲۰دستور more
۲۰نگه داری دستورات پنجره فرمان
۲۱دستورات سیستم عامل

دریافت اطلاع و راهنمایی از مدارک راهنمای متلب ۲۱

۲۱تمرین
----	------------

فصل ۵ توابع و عملیات ریاضی ۲۲

عملگرها ۲۲

عملگرهای آرایه ای ۲۲

عملگرهای ماتریسی ۲۲

اولویت عملگرها ۲۲

بعضی از توابع ریاضی ۲۳

اعدد مختلط ۲۳

مثال: نمایش هندسی عدد مختلط ۲۳

ماتریس با اعداد مختلط ۲۴

توابع مختلط کتاب خانه ای ۲۴

توابع تبدیل مختصات ۲۵

مثال: امپدانس معادل ۲۵

توابع خاص ۲۵

تمرین ۲۶

فصل ۶ بردارها ۲۷

تعریف بردار Vector Definition ۲۷

عملگر کالن ۲۷

تابع linspace() ۲۷

کاربرد یک بردار در تعریف بردار دیگر ۲۷

بردار تهی ۲۷

ترازینهاد Transpose ۲۸

اندیس اعضاء بردار A Vector Member Index ۲۸

بردار و رسم منحنی ۲۹

دستور plot(t,x) ۲۹

مثال: پرتابه عمودی ۲۹

افزودن توضیحات روی منحنی ۲۹

چند منحنی در یک صفحه مختصات ۳۰

مثال: مدولاتور دامنه ۳۰

- مثال: توان مصرفی مقاومت ۳۱
- توابع تحلیل داده ها ۳۲
- مثال: مدار معادل چند مقاومت موازی ۳۲
- مثال: تحلیل آماری نمرات دانشجویان ۳۲
- تمرین ۳۴

فصل ۷ برنامه نویسی ۳۵

- تشکیل ام- فایل ۳۵
- مثال: سرمایه نهائی بر حسب سرمایه اولیه ۳۵
- مثال: مشخصه پراب اسیلوسکوپ ۳۵
- مثال: ترسیم پارامتریک دایره ۳۷
- مثال: آنالیز پرتابه projectile با زاویه پرتاب ۳۹
- تمرین ۴۰

فصل ۸ بردارهای منطقی ۴۱

- اندیس و مقدار عناصر بردار ۴۱
- یافتن محل عناصری با مقدار معین ۴۱
- یافتن اندیس عناصر مورد نظر با تابع find() ۴۱
- مثال: رسم سرعت پرتابه و یافتن مینیم آن ۴۲
- مثال: استفاده از پنجره Data Statistics ۴۳
- یافتن مقدار عنصر با داشتن اندیس آن ۴۳
- تحقیق منطقی بودن یک بردار با تابع islogical() ۴۴
- تبدیل بردار عددی به منطقی با تابع logical() ۴۴
- استخراج بردار از بردار دیگر ۴۴
- چند مثال دیگر از کاربردهای بردار منطقی ۴۵
- مثال: یکسوساز نیمه موج (گراف ناپیوسته) ۴۵
- مثال: تقسیم صفر بر صفر ۴۵
- مثال: تقسیم بر صفر ۴۶
- تمرین ۴۷

فصل ۹ ماتریس ها ۴۸

۴۸Elementary Matrices پایه
۴۸identity matrix ماتریس یگانی
۴۸zeros() و ones() ماتریس های
۴۸تغییر مقدار عناصر ماتریس
۴۹قرار دادن یک ماتریس در ماتریس دیگر
۴۹مثال: ضبط ماتریس در فایل متن txt
۵۰مثال: ضبط ماتریس در فایل باینری mat با دستور save
۵۰حذف ردیف یا ستون
۵۱ماتریس های نمونه
۵۱ماتریس پاسکال
۵۱عمل گرهای ماتریسی
۵۱ضرب ماتریسی
۵۲توان ماتریسی
۵۳بعضی از توابع ماتریسی
۵۳چند تابع معمول
۵۳استخراج ماتریس از ماتریس دیگر
۵۳توابعی که به روی مجموعه عناصر ماتریس عمل میکنند
۵۳یک بحث آماری
۵۳نمودار ستونی ماتریس
۵۳مثال: نمودار ستونی میزان ریزش باران هر سال
۵۴نمودار منحنی ماتریس
۵۴پنجره Data Statistics
۵۴مثال: میزان ریزش باران هر سال، نمودار منحنی
۵۵تمرین

فصل ۱۰ مباحثی پیرامون گرافیک

۵۶رسم آسان با ezplot()
۵۶رسم توابع آشکار explicit functions
۵۸رسم توابع ضمنی
۵۹تابع داخلی fplot() و توابع ریاضی سریع التغییر

حل معادلات غیر آنالیتیک	۶۰
حل ترسیمی	۶۰
حل با تابع کتاب خانه ای <code>fzero()</code>	۶۱
برچسب گذاری با ماوس	۶۱
توری روی گراف از پنجره فرمان	۶۲
دو محور با مقیاس های عمودی متفاوت	۶۲
تعیین محدوده محورها	۶۳
ترسیم قطبی	۶۳
گیره های گرافیک <code>graphics handles</code>	۶۳
مؤلفه های RGB	۶۵
تمرین	۶۵

فصل ۱۱ توابع دیگر گرافیکی (۱) ۶۶

ترسیم ۳ بعدی	۶۶
تابع <code>plot3()</code>	۶۶
تابع <code>comet3()</code>	۶۷
سطوح فضائی	۶۷
دستور <code>meshgrid(a,b)</code>	۶۷
دستور <code>mesh(X,Y,Z)</code>	۶۸
رسم کُره با <code>sphere</code>	۶۹
مثال: رسم کُره و ایجاد افکت های تصویری	۶۹
مثال: رسم قله ها با تابع نمونه <code>peaks</code>	۶۹
دستور <code>mesh(Z)</code> ، نمایش سه بعدی یک ماتریس	۷۰
زاویه دید یک تصویر	۷۱
مثال: کلاه مکزیکی معروف MATLAB	۷۱
تصویربرداری با <code>getframe</code>	۷۲
نگه داری تصاویر با <code>moviein()</code>	۷۲
باز نمایش فیلم بادستور <code>movie()</code>	۷۲
تمرین	۷۳

فصل ۱۲ توابع دیگر گرافیکی (۲) ۷۴

۷۴ سایر روش های ترسیم سه بعدی

۷۴ surf() رسم سطح

۷۴ contour() منحنیهای نمایش گر ارتفاع

۷۵ meshc(), surfc() نمایش هم زمان حجم و کنتور آن

۷۵ تغییرات روی قسمتی از سطح ترسیم سه بعدی

۷۶ compass() نمودار عقربه ای

۷۷ bar() نمودار ستونی

۷۷ ezcontourf() رسم آسان کنتور

۷۸ fill() رنگی کردن با

۷۸ hist() پیشینه نگار

۷۹ pie() نمودار دایره

۷۹ **تمرین**

فصل ۱۳ ساختارهای تصمیم و تکرار ۸۰

۸۰ ساختارهای تصمیم و عوامل آن

۸۰ Relational Operators (مقایسه ای، رابطه ای)

۸۰ Logical Operators عملگرهای منطقی

۸۰ if بلوک

۸۱ switch بلوک

۸۱ **ساختارهای تکرار**

۸۱ for حلقه

۸۲ مثال: تعیین فاکتوریل عدد صحیح مثبت

۸۳ مثال: به دست آوردن طول زمان محاسبه

۸۳ بردار به جای حلقه for

۸۳ حلقه while و دستور break

۸۳ مثال: تعیین اعداد اول

۸۴ **منیو**

۸۵ **تمرین**

فصل ۱۴ توابع کاربر - تعریف و توابع چندجمله‌ای ۸۶

تابع خط فرمان inline function ۸۶

ام - فایل تابعی function M-file ۸۷

راه بُرد نیوتن ۸۷

دستورهای roots() و poly() ۸۸

مثال: تعیین ریشه های معادله با دستور roots() ۸۹

مثال: تعیین ضرائب چند جمله ای با دستور poly() ۸۹

ام - فایل تابعی با چند آرگومان خروجی ۸۹

تقریب جبری یک منحنی ۹۰

تابع polyfit(x,y,n) ۹۰

تابع polyval() ۹۰

مثال: به دست آوردن یک معادله چند جمله ای برای یک منحنی نامشخص (غیر آنالیتیک) ۹۰

مقایسه دو منحنی با استفاده از ginput ۹۱

تمرین ۹۱

فصل ۱۵ مباحثی پیرامون توابع ۹۲

تابع بدون مقدار برگشتی (بدون آرگومان خروجی) ۹۲

زیر تابع subfunction ۹۳

کامپایلر متلب MATLAB Compiler ۹۴

تبدیل ام - فایل به پرونده پی - کد pcode file ۹۴

تولید برنامه C با کامپایلر ۹۴

تبدیل برنامه گرافیکی به زبان C++ ۹۴

سازنده اکسل Excel Builder ۹۵

خلاصه دستورات کامپایلر ۹۵

گیره تابع function handle ۹۶

تابع تابع ۹۷

مثال: تابع humps(x) تابع نمونه متلب ۹۷

خودفراخوانی یک تابع Function Recursivity ۹۷

مثال: محاسبه فاکتوریل از طریق خود فراخوانی ۹۸

فصل ۱۶ مباحثی پیرامون رشته ها ۹۹

۹۹ رشته به مثابه آرایه (بردار) ۹۹

۹۹ بلوک try...catch ۹۹

۱۰۰ مثال: تابع مرتب سازی داخلی متلب sort() ۱۰۰

۱۰۰ تابع eval() ۱۰۰

۱۰۰ دستور lasterr ۱۰۰

۱۰۱ توابع single() و double() و عدد اسکی یک کاراکتر ۱۰۱

۱۰۲ مقایسه رشته ها ۱۰۲

۱۰۲ تابع strcmp(s1,s2) ۱۰۲

۱۰۲ عملگرهای مقایسه ای برای رشته ها ۱۰۲

۱۰۳ تعیین فرمت برای نمایش رشته با fprintf() ۱۰۳

۱۰۳ رشته $m \times n$ ۱۰۳

۱۰۳ نگه داری رشته در یک متغیر با sprintf() ۱۰۳

۱۰۵ تمرین ۱۰۵

فصل ۱۷ سیگنال، سیستم، فیلتر ۱۰۶

۱۰۶ تبدیلات فوریه ۱۰۶

۱۰۶ تبدیل فوریه گسسته fft() ۱۰۶

۱۰۶ تبدیل فوریه گسسته وارون ifft(t) ۱۰۶

۱۰۶ مثال: تبدیل فوریه گسسته ی تابع زمانی دندان ای (دندانه ارّه ای) ۱۰۶

۱۰۸ مثال: تبدیل بردار حوزه فرکانسی به بردار حوزه زمان برای شکل دندان ای (دندانه ارّه ای) ۱۰۸

۱۰۸ مثال: تبدیلات فوریه برای منحنی مربعی ۱۰۸

۱۱۰ توابع سیستم ها ۱۱۰

۱۱۰ دستور ایجاد تابع تبدیل زمان پیوسته tf ۱۱۰

۱۱۰ ترسیم bode() برای یک سیستم ۱۱۰

۱۱۰ واکنش پله ای و واکنش ایمپالسی step(), impulse() ۱۱۰

۱۱۰ دیاگرام نایکوئیست Nyquist Diagram ۱۱۰

۱۱۰ مثال: انتگراتور ۱۱۰

- ۱۱۱.....مثال: اسپلاتور.....
- ۱۱۲.....تغییر تابع تبدیل به فرم فضای حالت $tf2ss$
- ۱۱۲.....دستور $rlocus$ برای رسم مکان هندسی ریشه ها.....
- ۱۱۳.....سیستم فیدبک منفی با دستور $feedback()$
- ۱۱۳.....مثال: سیستم کنترل سرعت.....
- ۱۱۳.....مدل زمان گسسته **Discrete-Time Models**.....
- ۱۱۴.....مثال: تابع تبدیل زمان گسسته مدار $Lead$
- ۱۱۵.....فیلترها.....
- ۱۱۵.....فیلتر Butterworth و دستور $butter()$
- ۱۱۵.....مثال: فیلتر آنالوگ پائین گذر باترورث.....
- ۱۱۶.....فیلتر دیجیتال.....
- ۱۱۶.....مثال: فیلتر پائین گذر دیجیتال.....
- ۱۱۷.....**تمرین**.....

فصل ۱۸ ریاضیات نمادین Symbolic Math

- ۱۱۸.....انواع متغیرها.....
- ۱۱۸.....آرایه یا متغیر عددی با دقت افزوده $double array$
- ۱۱۸.....آرایه یا متغیر کاراکتری.....
- ۱۱۸.....شیء یا متغیر نمادین.....
- ۱۱۸.....انواع دیگر متغیرها.....
- ۱۱۸.....نمایش متغیرها در پنجره $workspace$
- ۱۱۹.....عملیات ریاضی و نوع متغیرها.....
- ۱۱۹.....روش معرفی متغیر نمادین.....
- ۱۲۰.....عملیات ریاضی بر روی متغیرهای نمادین.....
- ۱۲۰.....ریشه دوم.....
- ۱۲۰.....توان.....
- ۱۲۱.....مشتق.....
- ۱۲۱.....انتگرال.....
- ۱۲۱.....انتگرال محدود.....
- ۱۲۱.....تبدیل به کسرهای جزئی با دستور $residue()$

۱۲۱مثال: ریشه و قطب یک تابع تبدیل
۱۲۲تبدیل عبارت جبری به کسر متعارفی گویا
۱۲۲متغیر مستقل نمادین
۱۲۳جای گزینی عدد نمادین در متغیر نمادین
۱۲۳یافتن متغیرهای نمادین
۱۲۳نمایش اعداد نمادین
۱۲۳اعداد مختلط نمادین
۱۲۴مزدوج یک عدد مختلط
۱۲۴توابع نمادین
۱۲۴معرفی یک تابع کلی
۱۲۴جای گزینی یک عبارت به جای x (subs = substitution)
۱۲۵تابع نمادین مختلط
۱۲۵حد تابع
۱۲۶تابع ام-فایلی نمادین
۱۲۶سری ها
۱۲۶توابع آسان ساز
۱۲۶تابع pretty()
۱۲۷توابع collect() و expand()
۱۲۷فاکتورگیری factor()
۱۲۸ساده کردن با simplify()
۱۲۸ساده کردن با simple()
۱۲۸رسم تابع نمادین با ezplot()
۱۲۹ماتریس های نمادین
۱۲۹ماتریس با عناصری از متغیرهای نمادین
۱۳۰ماتریس با عناصری از توابع نمادین
۱۳۰حل معادلات
۱۳۰معادله چند جمله ای
۱۳۱دستگاه معادلات نمادین
۱۳۲حل دستگاه معادلات در متلب

۱۳۲معادلات دیفرانسیل
۱۳۲معادله دیفرانسیل یک مجهولی
۱۳۳دستگاه معادلات دیفرانسیل
۱۳۳دریافت راهنما در مورد ریاضیات نمادین
۱۳۴تمرین

فصل ۱۹ واسط گرافیکی کاربر ۱۳۵

۱۳۵واسط گرافیکی کاربر (GUI) graphical user interface
۱۳۶property inspector شاخصه یاب
۱۳۶Name or String شاخصه عنوان
۱۳۶Tag شاخصه برچسب
۱۳۷Callback Function توابع فراخوان
۱۳۷برنامه نویسی
۱۳۸مثال: شبیه سازی مدولاسیون دامنه
۱۴۱تمرین

بخش دو EXCEL ۱۴۲

فصل ۲۰ نکاتی پیرامون صفحه گسترده ۱۴۳

۱۴۳spreadsheet cells سلولهای صفحه گسترده
۱۴۳برچسب پیش فرض یک سلول
۱۴۳برچسب گذاری دل خواه
۱۴۶Formula Bar میله فرمول
۱۴۶رجوع نسبی و رجوع مطلق به سلول
۱۴۷روش رجوع نسبی و رجوع مطلق به سلول
۱۴۷نامگذاری

فصل ۲۱ توابع کتاب خانه ای و نمودارها ۱۵۰

۱۵۰پر کردن سری سلول ها
۱۵۰چند تابع کتاب خانه ای
۱۵۱نمودارها

۱۵۱	نمودار ستونی
۱۵۱	ترسیم منحنی
۱۵۲	جبری سازی منحنی
۱۵۳	هیستوگرام

فصل ۲۲ تابع کاربر - تعریف ۱۵۵

۱۵۵	وی - بی - آ در اکسل VBA in Excel
۱۵۵	برنامه نویسی
۱۵۶	استفاده از تابع VBA در صفحه گسترده
۱۵۸	ضبط ماکرو

فصل ۲۳ ابزارهای محاسباتی ۱۵۹

۱۵۹	Goal Seek
۱۶۰	Solver
۱۶۳	Data Table جدول داده

مقدمات

واژه MATLAB

کلمه فوق، سر- واژه‌ی عبارت MATrix LABoratory است، که در این متن به شکل 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 مقوله‌هایی نظیر: محاسبات ریاضی، دسترسی به/ و آنالیز داده‌ها، مدل‌سازی 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 جعبه‌ابزارهایی 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 متغیر 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 تنهایی 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.
- ۴- 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 اجرائی 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 کامل کنار 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. از این رو 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، بعد از 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 بعد از آن نوشته شود جزو دستورات منظور نمی‌شود.

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

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

دورچین‌ها

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

مؤلفه‌های اصلی متلب

ماتریس Matrix

A که یک ماتریس 2×3 است نمونه‌ای از یک ماتریس دو بُعدی $m \times n$ 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:

- عملدهی 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 ردیف ۲ ستون ۳ و عضو (عنصر) ردیف ۱ ستون ۲ 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 دیگر این 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:

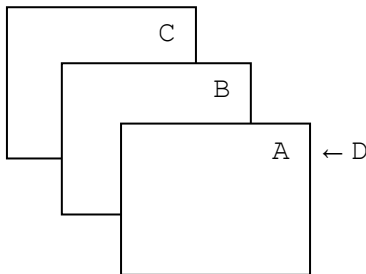
برای استخراج قسمتی از یک ماتریس ابتدا ردیف‌ها و سپس ستون‌های مستخرجه از آن ردیف‌ها را (به سبک فوق) می‌نویسیم.

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 تنهایی برای بردار ردیفی به کار می‌رود.
- متلب ترجیح می‌دهد با ماتریس یا بردار کار کند، لذا یک عدد (یا اسکالر) را، ماتریس عددی 1×1 نام می‌دهد.

ماتریس سه بُعدی (فضائی)

یک ماتریس سه بُعدی $m \times n \times p$ دارای p صفحه است که هر صفحه یک ماتریس دو بُعدی $m \times n$ است. انتخاب مقادیر m ، n ، p دلخواه است، اما باید دقت کرد که m و n 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:

$$\begin{aligned} D(:,:,1) &= A \\ D(:,:,2) &= B \\ D(:,:,3) &= C \end{aligned}$$

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

راهنما:

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

مثال:

یک ماتریس فضائی با سه صفحه تولید کنید. تعداد ردیف و ستون هر صفحه، و تعداد صفحه‌های آن را 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 تنهایی مقادیر روی یک محور را پوشش دهد.

مثال:

یک بردار ردیفی ۴ عضوی (۴ 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 length(), size(), ndims() 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.

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 متغیر 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:

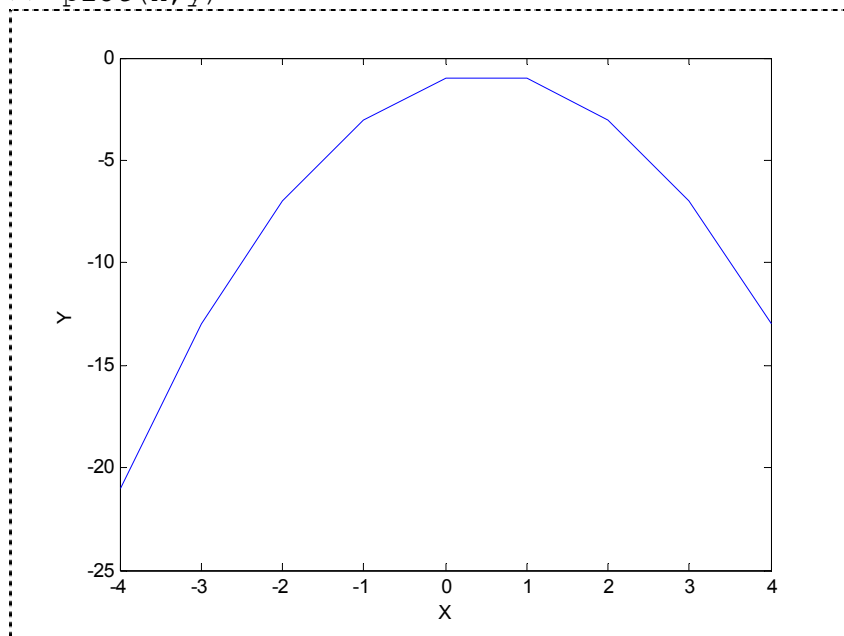
```
>> x = [-4 -3 -2 -1 0 1 2 3 4];
```

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 $^{\wedge}$ استفاده می‌شود. شرح بیشتر عملیات آرایه‌ای بعداً می‌آید.

دقت کنید:

دستور $\text{plot}(x, y)$ 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 معتناهی 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 2×3 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 ۶ 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) $n = 6$ (ستون) تولید می‌شود.

مثال:

یک بردار عددی با ۶ 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 و یک 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 $\text{magic}(n)$ یک مربع جادوئی $n \times n$ می‌سازد. خاصیت مربع جادوئی این است که حاصل جمع عناصر آن در طول ردیف، قطر، و ستون برابرند.

مثال:

یک مربع جادوئی 2×2 بسازید.

حل:

```
>> Mg = magic(2)
```

```
Mg = 1 3
     4 2
```

مثال:

یک ماتریس 4×4 به نام Mg با تکرار ماتریس جادوئی 2×2 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 $\text{logical}()$ ، یک بردار با اعضاء عددی را به یک بردار با اعضاء صفر و یک منطقی تبدیل می‌کند. منطقی بودن یک آرایه با تابع $\text{islogical}()$ امتحان می‌شود. متغیر منطقی فقط دو مقدار درستی (منطق یک) و نادرستی (منطق صفر) را می‌گیرد، که از لحاظ نوع با 0 و 1 عددی متفاوت هستند. مقادیر منطقی یک بایت از حافظه را اشغال می‌کنند، در حالی که مقادیر عددی (از نوع double که پیش‌فرض متلب است) هشت بایت جا می‌گیرند.

مثال:

تعداد بایت اشغالی عدد یک و منطق 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 $\text{islogical}()$ 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 آرایه‌هایی با اعضاء صفر و یک عددی به آرایه منطقی تبدیل شوند.
- توابعی که صحت یا سقم امری را امتحان می‌کنند، در صورت صحت، منطقی یک و در صورت کذب، منطقی 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 حاصل ضرب 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 متساوی‌العناصر 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 `length()`، `size()`، `ndims()` را در مورد بردار تمرین ۱ اجرا کنید
- ۳- یک بردار منطقی به نام L1 با ۵ عنصر بسازید.
- ۴- بردار منطقی تمرین ۳ را: الف) اندیس Rv قرار دهید. ب) در Rv ضرب کنید.
- ۵- * دو ماتریس عددی ۳×۶ به نام‌های M1، M2 با مربع جادویی بسازید.
- ۶- توابع `length()`، `size()`، `ndims()` را در مورد ماتریس‌های تمرین ۵ اجرا کنید.
- ۷- با ماتریس‌های تمرین ۴ یک ماتریس سه بعدی را به نام Qm تعریف کنید و نمایش دهید.
- ۸- توابع `length()`، `size()`، `ndims()` را در مورد ماتریس تمرین ۷ اجرا کنید.
- ۹- یک بردار عددی ستونی به نام Rh با ۵ عنصر تعریف کنید. تابع $\gamma = Rh.^2$ را رسم کنید.
- ۱۰- توابع `length()`، `size()`، `ndims()` را در مورد بردار تمرین ۹ اجرا کنید.
- ۱۱- نوع، اندازه، و تعداد بایت آرایه‌های تولید شده را در پنجره Workspace ملاحظه و بررسی کنید.

آشنایی با متغیرهای متلب

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

بعضی از پیش-تعریف‌ها

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	<code>pow2(1,-52)</code> 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	$4 \cdot \text{atan}(1)$ or $\text{imag}(\log(-1))$ 3.141592653589793e+000
Inf, inf	Infinity	نتیجه تقسیم عدد بر صفر
NaN, nan	Not a Number	تقسیم <code>inf</code> بر <code>inf</code> 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 نمایی چاپ می‌کند.

مثال:

با فرمت مناسب مقدار `eps` و `eps + realmin` 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 بایت متغیر (آرایه‌های تعریف شده، در پنجره Workspace نمایش داده می‌شوند. لذا توصیه می‌شود همواره از انتخاب View از منوی اصلی، پنجره Workspace را تیک بزنید تا بتوانید مشخصه‌های متغیرهای خود را در فضای حافظه مشاهده نمایید. اگر به روی یک متغیر در پنجره Workspace کلیک-کلیک کنید، دسترسی بیش‌تری به آن پیدا کرده، حتی می‌توانید آن را ادیت کنید (امتحان کنید).

اعلام نوع متغیر

اصولاً متلب نیازی به اعلام یا معرفی 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 دهی هم‌زمان، با دقت افزوده (مضاعف) double نگه‌داری می‌کند. هر عدد با دقت افزوده double ۸ بایت از حافظه و با دقت محدود single ۴ بایت از حافظه را می‌گیرد. چند نوع عدد صحیح integer نیز در متلب وجود دارند. هر عدد صحیح 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 ۱۴ رقم بعد از نقطه-اعشار را نشان می‌دهد.
- فرمت کوتاه عدد را به ۴ 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 برمی گرداند.

مثال:

عدد گنگ 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 تو ساخت built-in است، که در داخل متلب با دقت افزوده وجود دارد.

دقت کنید:

به تعداد بایت متغیرها در پنجره Workspace.

دستورهای در مورد متغیرها

نام دستور	نتیجه اجرا
who	نام متغیرها را می دهد
whos	نام و اطلاعات بیشتری از مشخصه های متغیرها می دهد
clear var	یک متغیر را پاک می کند
clear	کلید متغیرهای تعریف شده را پاک می کند
clear all	حافظه را کلاً پاک می کند
pack	حافظه را منظم و جمع و جور می کند

چند راهنمایی نوشتاری

نوشتن چند دستور در یک سطر

مثال:

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

```
x = 3
```

```
z = 39304
```

دقت کنید:

چون در مقابل γ سمی کالن (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 شده اند، حتی در اجراهای بعدی متلب باقی می ماندند، و تکرار آن ها با کلیدهای \uparrow و \downarrow انجام میشود. برای اجرای یک سطر تکراری کلید <Enter> و برای حذف یک سطر تکراری نادلخواه کلید <Esc> یا در ابتدای سطر کلید <Ctrl+K> را بزنید. پنجره Command History هم فرمان های قبلی را نگه می دارد. اگر سطر فرمانی در هریک از پنجره های 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 بعد از ممیز حاصل عبارت $1 + n * \text{eps}$ ظاهر شود (به این شکل نشان داده شود
 1.0000000000000001)
- 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 `realmin` و `realmax` را امتحان کنید. عبارات `type realmin` و `type realmax` را اجرا و بررسی کنید.
- ۴- متغیر x تمرین ۲ را تقسیم بر صفر کرده برابر y قرار دهید.
- ۵- متغیر t را مساوی صفر تعریف کرده $f = \sin(t)/t$ را به دست آورید.
- ۶- عدد گنگ 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.