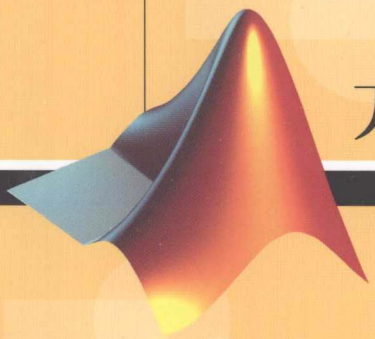

方程建模与MATLAB软件

司守奎　孙玺菁　王兴平　周　刚　编著

清华大学出版社

第1章

MATLAB的基本使用方法

1.1 MATLAB概述

1.1.1 MATLAB发展史

MATLAB是由两个英文单词Matrix和Laboratory的前三个字母组成. MATLAB诞生于20世纪70年代后期的美国新墨西哥大学计算机系主任Clever Moler教授之手. Moler教授为了减轻学生编程的负担,用Fortran语言编写了最早的MATLAB. 1984年由Little、Moler、Steve Bangert合作成立的MathWorks公司正式把MATLAB推向市场. 到20世纪90年代,MATLAB已成为国际控制界的标准控制软件. 从7.2版本开始,版本编号以年份来命名,每年3月份推出的用a表示,9月份推出的则以b表示,例如R2014b代表2014年9月推出的版本,R2015a指2015年3月份推出的版本.

MATLAB的符号运算功能是借助于其他符号运算内核完成的,R2008a之前的版本以Maple为内核,R2008b之后的版本以MuPad为内核,不同内核对符号运算的具体支持可能存在差别.

MATLAB的主要版本历程如表1-1所示.

表1-1 MATLAB主要版本历程

版　本	发行版本编号	发布时间
MATLAB1.0	N/A	1984
MATLAB2	N/A	1986
MATLAB3	N/A	1987
MATLAB4.2c	R7	1994
MATLAB5.0	R8	1996
MATLAB6.0	R12	2000
MATLAB7	R14	2004
MATLAB7.2	R2006a	2006
MATLAB7.14	R2012a	2012.3
MATLAB8.0	R2012b	2012.9
MATLAB8.5	R2015a	2015.3

1.1.2 MATLAB帮助

对于任何MATLAB的使用者，都必须学会使用MATLAB的帮助系统，因为没有必要清楚地记住成千上万个不同函数的调用情况，所以MATLAB的帮助系统是学习MATLAB编程和开发最好的教科书.

1. 命令形式

MATLAB的各个函数，一般都有使用帮助和函数功能说明. 各个工具箱通常情况下也具有一个与工具箱名相同的M文件用来说明工具箱的构成内容等，这里的工具箱是指MATLAB中具有特定功能的某类函数集合，MATLAB的M文件分为两种，一种为脚本文件，一种为函数文件，下面再详细介绍. 在MATLAB命令窗口中，可以通过命令来获取这些纯文本的帮助信息.

通常能够起到帮助作用、获取帮助信息的命令有help、lookfor、which、doc、get、type等.

(1) help命令

help命令是MATLAB中最有用的命令之一. 下面介绍help的几种常见使用情况.

① 直接使用help命令

在命令窗口直接输入命令help，并回车，则在命令窗口显示MATLAB的所有工具箱信息.

以下我们用

```
help↵                    % 显示所有工具箱信息.
```

表示前面段落的内容，这里↵表示回车，%是MATLAB的注释引导符，%后面的是注释内容.

② 使用help　工具箱名

使用命令“help　工具箱名”，可以获取该工具箱的所有函数的信息. 例如

```
help optim↵              % 获得优化工具箱的基本信息和函数列表.
```

③ 使用help　函数名

使用命令“help　函数名”，就可以获得该函数的帮助信息. 例如

```
help linprog↵            % 获得优化工具箱中线性规划命令 linprog 的帮助信息.
```

(2) lookfor命令

lookfor命令在MATLAB默认路径下搜索所有M文件第一个注释行中的关键字. 通常在不确定某个函数时，仅知道该函数的功能，lookfor命令可以根据用户提供的完整或不完整的关键字，去搜索出一组与之相关的命令. 例如

```
lookfor  integral↵       % 查找所有有关积分的命令.
```

(3) which命令

which命令可以用来定位函数的位置. 通过这个位置信息，可以获取函数所属的类别. 通常，在创建一个MATLAB文件时，为了避免与系统函数等同名，就应该先用“which　文件名”搜索查找是否存在你想要保存的文件名.

另外，利用得到的位置信息可以查找一些相关联的文件的帮助信息. 例如，在编程过程中，需要计算一个微分方程在指定点的数值解，但想不起该函数名，只记得求微分方程数值解的命令 ode45，因此，采用 which ode45 定位 ode45.

```
which ode45 ↵
D:\Program Files\MATLAB\R2015a\toolbox\MATLAB\funfun\ode45.m
```

从给出的地址可以看出，ode45 命令属于 funfun 类. 于是用 help funfun 查找该类别信息，在该类别的 Input and Output functions 子类别中找到：

```
deval - Evaluates the solution of a differential equation problem.
```

然后，再通过用 help deval 获得该函数的详细帮助信息和使用方法.

(4) 超文本格式的帮助文件

在 MATLAB 中，关于一个函数的帮助信息可以用 doc 命令以超文本的方式给出. 有些函数既可以应用于数值型数据，也可以应用于符号型数据，一般来说，其对应于不同数据类型的超文本帮助文件是不同的，为了区别它们，分别采用形如“doc 函数名”和“doc sym/函数名”的命令来获取此函数的数值型和符号型的超文本帮助. 如

```
doc eigs ↵          % 获取求数值矩阵最大模特征值及对应特征向量的超文本帮助;
doc eig ↵           % 获取求数值矩阵特征值和特征向量的超文本帮助;
doc sym/eig ↵       % 获取求符号矩阵特征值和特征向量的超文本帮助.
```

2. pdf 帮助文件

可从 MathWorks 网站上下载有关的 pdf 帮助文件.

1.2 MATLAB 基础知识

1.2.1 MATLAB 变量

变量是 MATLAB 的基本元素之一. 与其他程序设计语言不同的是，MATLAB 不需要对所使用的变量进行事先声明，也不需要指定变量的类型，系统会根据该变量被赋予的值或对该变量所进行的操作来自动确定变量的类型.

1. MATLAB 变量命名

在 MATLAB 中，变量命名有如下规则：

(1) 变量名长度最大为 63 个字符，超过的字符系统忽略不计.

(2) 变量名以字母开头，且只能由字母、下画线和数字混合组成.

(3) 变量名区分大小写.

(4) 变量名不要与 MATLAB 中已有的函数名、变量名和关键字相同.

(5) 当进行复数运算时，不能使用 i 与 j，这两个字符表示复数运算中的虚数单位.

2. MATLAB 系统变量

MATLAB 中有一些特定的变量，不需要用户定义，它们已经被预定义了某个特定的值. 这些变量称作系统变量(有些书中也称为常量). 系统变量在 MATLAB 启动时就产生. MATLAB 常用的系统变量如表 1-2 所示.

表 1-2 MATLAB常用的系统变量

系统变量	变量功能	系统变量	变量功能
ans	运算结果的默认变量名	flintmax	最大的正整数
pi	圆周率	realmax	最大的正浮点数
eps	浮点数相对精度	realmin	最小的正浮点数
inf	无穷大	nargin	函数输入参数个数
NaN	不定数	nargout	函数输出参数个数

1.2.2 MATLAB数据类型

MATLAB数据类型有以下几种：数值类型，字符串，日期和时间，结构数组，细胞数组(元胞数组)，函数句柄，Java对象，逻辑类型等.

数值类型包括双精度浮点型，单精度浮点型，整型类型.

MATLAB中变量默认的类型为双精度浮点型(double). 在MATLAB中，通过表1-3所示的format函数来控制double型数据的显示格式.

表 1-3 double型数据的显示格式

format short	带有4位小数的显示格式，是MATLAB的默认显示格式
format long	带有15位小数的显示格式，如3.141592653589793
format shortE	显示带有4位小数的科学记数法，如3.1416e+00
format longE	显示带有15位小数的科学记数法，如3.141592653589793e+00
format shortG	更紧凑的短的固定小数或科学记数法，带有5位数字，如3.1416
format longG	更紧凑的长的固定的小数或科学记数法，带有15位数字，如3.14159265358979
format shortEng	短的工程表示，小数点后4位数字，指数是3位数字，如3.1416e+000
format longEng	长的工程表示，15位有效数字，指数是3位数字，如3.14159265358979e+000
format+	正数为+，负数为−，0为空格
format bank	显示为货币格式的数，带有2位小数，如3.14
format hex	十六进制表示的二进制双精度数，如400921fb54442d18
format rat	显示为有理数，如355/113
format compact	抑制多余的换行符在单一屏幕上显示输出
format loose	添加换行符，使输出更易读

在MATLAB语言中，最重要的功能就是进行各种矩阵的运算，所有的数值功能都是以矩阵为基本单位来实现的.

在MATLAB中，最基本的数据结构就是二维矩阵. 通过对二维矩阵的操作，可以方便地存储和访问大量的数据. 矩阵的元素可以是数值类型、逻辑类型、字符串类型或者其他任何MATLAB支持的数据类型.

下面先介绍一下简单的数值矩阵.

1.2.3 数值矩阵的建立

在MATLAB中，矩阵的建立方式多种多样. 比较常用的建立方式有直接输入、通过语句和函数建立矩阵和从外部数据文件中导入矩阵3种.

试读结束：需要全本请在线购买： www.ertongbook.com

1. 直接输入

直接输入是最简单的矩阵构建方式. 直接输入矩阵,应遵循如下几条规则:

(1) 矩阵元素应当在方括号内;

(2) 行内的元素,用逗号或者空格隔开;

(3) 行与行之间,用分号或者回车分割;

(4) 元素可以是数值或表达式.

2. 通过语句和函数建立矩阵

```
t = [0:0.1:5]            % 产生从 0 到 5 的行向量,元素之间间隔为 0.1.
t = linspace(n1,n2,n)    % 产生 n1 和 n2 之间线性均匀分布的 n 个数 (缺省 n 时,产生 100 个数).
t = logspace(n1,n2,n)    % 在 10^n1 和 10^n2 之间按照对数距离等间距产生 n 个数(缺省 n 时,产生 50
                           个数).
```

3. 从外部数据文件导入矩阵

MATLAB 可以从外部的纯文本文件、Excel 文件和各种数据库文件导入矩阵. 下面介绍从纯文本文件和 Excel 文件导入矩阵.

(1) 导入纯文本文件

可以把 Word 文档中整行整列的数据粘贴到纯文本文件,然后导入到 MATLAB 工作空间中.

例 1.1 纯文本文件 data11.txt 中存放如下格式的数据,把数据导入 MATLAB 中.

```
6  3  6  7  4  2  6
4  9  5  3  8  5  8
5  2  8  9  7  4  3
9  6  7  3  9  2  6
```

首先用记事本把上述数据保存到纯文本文件 data11.txt 中,文件存放在 MATLAB 的当前工作路径下. 以下所有操作的数据文件必须放在 MATLAB 的当前工作路径下,也就是说程序文件和数据文件要放在同一个目录下.

MATLAB 导入纯文本文件的调用格式如下:

```
a = load('data11.txt')
```

或者是

```
a = textread('data11.txt')
```

(2) 导入 Excel 文件

MATLAB 读入 Excel 文件的命令是 xlsread,使用格式为

```
num = xlsread(filename,sheet,Range)
[num,txt] = xlsread(filename,sheet,Range)
```

其中第 1 个返回值是数值矩阵,第 2 个返回值是字符串的细胞数组,sheet 是表单序号,Range 是数据域的范围.

例 1.2 把 Excel 文件 data12.xls 的表单 Sheet1 的域"A2:D5"中的数据赋给 a,表单 Sheet2 中的全部数据赋给 b.

```
a = xlsread('data12.xls',1,'A2:D5')
b = xlsread('data12.xls',2)
```

1.2.4 特殊矩阵

1. 单位矩阵

```
eye(m)                  % 生成 m 阶单位阵.
eye(m,n)                % 生成 m×n 矩阵,其中得到一个位于左上角的最大单位矩阵而其余处补 0.
```

2. 全部元素为 1 的矩阵

```
ones(n)                 % 生成全部元素为 1 的 n 阶方阵.
ones(m,n)               % 生成全部元素为 1 的 m×n 矩阵.
```

3. 全部元素为 0 的矩阵

```
zeros(n)                % 生成全部元素为 0 的 n 阶方阵.
zeros(m,n)              % 生成全部元素为 0 的 m×n 矩阵.
```

4. 空矩阵

空矩阵是一个特殊矩阵,这在线性代数中是不存在的. 例如

```
a = [ ]
```

矩阵 a 在工作空间之中,但它的大小为零. 通过空矩阵的办法可以删除矩阵的行与列. 例如

```
b(:,3) = []
```

表示删除矩阵 b 的第 3 列.

5. 随机数矩阵

```
rand(m,n)                % 产生 m×n 矩阵,其中的元素是服从[0,1]区间上均匀分布的随机数.
randi([imin,imax],m,n)   % 生成 m×n 矩阵,其中的元素为[imin,imax]区间上的随机整数.
normrnd(mu,sigma,m,n)    % 产生 m×n 矩阵,其中的元素是服从均值为 mu,标准差为 sigma 的正态
                           分布的随机数.
randn(m,n)               % 产生均值为 0,方差为 1 的正态分布的 m×n 随机数矩阵.
exprnd(mu,m,n)           % 产生 m×n 矩阵,其中的元素是服从均值为 mu 的指数分布的随机数.
poissrnd(mu,m,n)   % 产生 m×n 矩阵,其中的元素是服从均值为 mu 的泊松(Poisson)分布的随机数.
unifrnd(a,b,m,n)   % 产生 m×n 矩阵,其中的元素是服从区间[a,b]上均匀分布的随机数.
mvnrnd(mu,sigma,n) % 产生 n 对均值向量为 mu,协方差矩阵为 sigma 的多维正态分布的随机数.
```

1.2.5 运算符

MATLAB 提供了丰富的运算符,主要包括算术运算、关系运算和逻辑运算. 算术运算用于数值计算,关系运算和逻辑运算的返回值为逻辑型变量,其中 1 代表逻辑真,0 代表逻辑假.

1. 算术运算符

MATLAB 提供的基本算术运算有加(+)、减(-)、乘(*)、除(/或\)和乘方(^). 常用

的算术运算符如表 1-4 所示.

表 1-4 算术运算符

运 算 符	功 能
+	标量或矩阵加法
-	标量或矩阵减法
*	标量或矩阵乘法
.*	数组的逐个元素相乘
/	标量或矩阵的右除(矩阵相当于右边乘以广义逆阵)
./	数组的逐个元素右除
\	标量或矩阵的左除(矩阵相当于左边乘以广义逆阵)
.\	数组的逐个元素左除
^	标量或矩阵的乘方
.^	数组的逐个元素的乘方
'	矩阵的共轭转置
.'	矩阵的转置

2. 关系运算符

关系运算符用于比较两个操作数的大小,返回值为逻辑型变量. 在 MATLAB 中,关系运算符如表 1-5 所示.

表 1-5 关系运算符

关系运算符	说 明	函数	关系运算符	说 明	函数
<	小于	lt	>=	大于或等于	ge
<=	小于或等于	le	==	恒等于	eq
>	大于	gt	~=	不等于	ne

例 1.3 关系运算符操作实例.

```
clc, clear
a = 4 * rand(4), b = randi([0,4],4)          % 生成两个 4 阶方阵
s11 = a < b, s11 = lt(a,b)                   % 两种表示结果都是一样的
s21 = b <= 2, s22 = le(b,2)                  % 两种表示结果都是一样的
```

3. 逻辑运算符

在 MATLAB 中,逻辑运算主要分为两类,分别为基本逻辑运算和逐位逻辑运算. 基本逻辑运算有四种,分别为逻辑与(&)、逻辑或(|)、逻辑非(~)和逻辑异或,如表 1-6 所示. 逻辑与和逻辑或为双目运算符,逻辑非为单目运算符.

例 1.4 基本逻辑运算实例.

```
clc, clear
a = 3 * rand(3,4), b = unifrnd(0,3,3,4)
s1 = a > 1, s2 = gt(b,1.2)
s31 = s1 & s2, s32 = and(s1,s2)              % 逻辑与运算,两种表示结果都是一样的
s4 = xor(s1,s2)                              % 逻辑异或运算
```

表 1-6 基本的逻辑运算

运 算 符	函 数	说 明	运 算 符	函 数	说 明
&	and	逻辑与	~	not	逻辑非
|	or	逻辑或		xor	逻辑异或

在 MATLAB 中，可以对二进制数进行逐位逻辑运算，并将运算的结果转换为十进制数. 逐位逻辑运算函数如表 1-7 所示.

表 1-7 逐位逻辑运算函数

函 数	说 明	函 数	说 明
bitand(a,b)	逐位逻辑与	bitcmp(a)	逐位逻辑非
bitor(a,b)	逐位逻辑或	bitxor(a,b)	逐位逻辑异或

例 1.5 逐位逻辑运算实例.

```
clc, clear
a = imread('data6.bmp');        % 读取一副 bmp 图像
b = bitand(a,240);              % 原图像与 11110000(二进制) = 240(十进制)逐位与运算,提出原图像的高 4 位数据
c = bitand(a,15);               % 原图像与 00001111(二进制) = 15(十进制)逐位与运算,提出原图像的低 4 位数据
imshow(a)                       % 显示原图像
figure,imshow(b)                % 显示原图像的高 4 位数据的图像
figure, imshow(c)               % 显示原图像的低 4 位数据的图像
```

1.3 MATLAB 程序设计

1.3.1 文件类型与 M 文件

1. MATLAB 文件类型

MATLAB 能够创建的文件类型有很多种，主要包括：

(1) M 文件，它是 MATLAB 中常用的文本文件，后缀为 m.

(2) model 文件，它是 MATLAB 中通过 Simulink 组件建立的模型文件，后缀为 mdl.

(3) figure 文件，它是绘图后产生的图形窗口文件，后缀为 fig.

(4) data 文件，它是标准的 MATLAB 二进制数据文件，后缀为 mat.

(5) stateflow 文件，它是 MATLAB 中状态流文件，后缀为 cdr.

(6) report generator 文件，它是 MATLAB 中生成的报表文件，后缀为 rpt.

其中，mat 文件是 MATLAB 以标准二进制格式保存的数据文件，可将工作空间中有用的数据变量保存下来. mat 文件的生成和调用是由函数 save 和 load(matfile)完成的.

2. M 文件介绍

M 文件是用 MATLAB 语言编写的、可以完成某些操作或者实现某种算法的程序文本文件. 实际上，MATLAB 提供的内部函数以及各种工具箱都是利用 MATLAB 语言开发的 M 文件.

通常，M 文件根据调用方式的不同分为两类：脚本文件和函数文件，脚本文件是许多 MATLAB 代码按顺序组成的命令序列集合的 M 文件. 与在命令窗口逐行执行文件中的所有指令，其结果是一样的. 函数文件也称为子程序，它必须由 MATLAB 程序来调用. 函数文件往往具有一定的通用性，并且可以进行递归调用.

(1) 脚本文件

脚本文件的格式特征如下：

① 前面的若干行通常是程序的注释，注释以“%”开始，当然注释可以放在程序的任何部分. 注释可以是汉字，注释是对程序的说明，它增加了程序的可读性. 在执行程序时，MATLAB 将不理会“%”后直到行末的全部文字.

② 然后是程序的主体，整个程序应按 MATLAB 标识符的要求命名文件名.

(2) 函数文件

函数文件是用来定义子程序的. 它与脚本文件的主要区别有 3 点：

① 由 function 开头，紧随其后的函数名要与文件名相同.

② 一般有输入输出变量，可进行变量传递.

③ 除非用 global 声明，函数中的变量均为局部变量，不保存在工作空间中.

例 1.6 编写计算 $y=x^2+2x+3$ 的值的 MATLAB 函数.

```
function y = myfun1(x);
y = x.^2 + 2 * x + 3;
```

上述函数虽然只有两行，但也要放在一个文件中，调用函数 myfun1 的命令可以出现在命令窗口中或其他文件中，使用起来不太方便. 在这种情形下，我们可以定义匿名函数如下：

```
y = @(x) x.^2 + 2 * x + 3;
```

匿名函数的定义，和它的调用语句可以放在同一个文件中，使用起来很方便. 本书中，除非函数要有两个返回值，无法使用匿名函数外，其他情形下，我们都使用匿名函数.

1.3.2 MATLAB 流程控制结构

MATLAB 为用户提供了 4 种流程控制结构：条件结构、循环结构、开关结构和试探结构，用户可以根据某些判断结果来控制程序流的执行次序. 与其他程序语言相比，除了试探结构为 MATLAB 所特有外，其他结构与用法都十分相似. 因此，本小节只结合 MATLAB 特点对这几种流程控制结构做简要的介绍.

1. if 条件结构

if 条件结构是实现分支结构程序最常用的一种语句，其使用方式如下：

```
if 表达式
    语句组 1
elseif
    语句组 2
else
    语句组 3
end
```

2. switch-case 开关结构

switch-case 开关结构的使用格式为

```
switch 开关表达式
    case 表达式 1
        语句组 1
    case 表达式 2
        语句组 2
    …… …
    case 表达式 n
        语句组 n
    otherwise
        语句组 n+1
end
```

例 1.7 通过键盘输入百分制成绩，输出成绩的等级，其中 90～100 分等级为 A，80～89 分等级为 B，70～79 分等级为 C，60～69 分等级为 D，60 分以下等级为 E.

下面我们使用三种方式编写 MATLAB 程序.

(1) 使用 if 语句，编写文件名为 eq_ex1_7_1.m 的脚本文件：

```
clc, clear
n = input('请输入百分制成绩 n = ?\n')
if n<0 | n>100
    disp('输入有误,请重新输入百分制成绩\n')
elseif n>=90 & n<=100
    disp('A')
elseif n>=80 & n<=89
    disp('B')
elseif n>=70 & n<=79
    disp('C')
elseif n>=60 & n<=69
    disp('D')
else
    disp('E')
end
```

(2) 使用逻辑语句，编写文件名为 eq_ex1_7_2.m 的脚本文件：

```
clc, clear
n = input('请输入百分制成绩 n = ?\n')
m = 65*(n>=90 & n<=100) + 66*(n>=80 & n<=89) + 67*(n>=70 & n<=79) + …
 68*(n>=60 & n<=69) + 69*(n<=59);    % 把分数转换成 ABCDE 对应的 ascii 码
% 上面使用了续行符…
disp(char(m))                                % 显示 ascii 码对应的字符
```

(3) 使用 switch-case 开关结构，编写文件名为 eq_ex1_7_3.m 的脚本文件：

```
clc, clear
n = input('请输入百分制成绩 n = ?\n')
if n<0 | n>100
    disp('输入有误,请重新输入百分制成绩\n')
```

试读结束：需要全本请在线购买： www.ertongbook.com

```
else
    switch 11 - floor(n/10)
        case {1,2}
            disp('A')
        case 3
            disp('B')
        case 4
            disp('C')
        case 5
            disp('D')
        otherwise
            disp('E')
    end
end
```

3. for 循环结构

for 语句是一种基本的实现循环结构的语句，能够以确定的次数执行某一段程序. for 语句的格式如下：

```
for 指标 = 数组值
    语句组
end
```

for 语句的使用和其他程序语言是类似的，我们再举两个与其他程序语言不同的 for 语句例子.

例 1.8 编写依次显示数字 1,2,5,2,3 的小程序，要求显示一个数字后，停顿该数字所表示的秒数.

```
clc, clear
for i = [1 2 5 2 3]
    disp(i), pause(i)
end
```

例 1.9 依次显示 4 个 4 维单位坐标向量.

```
clc, clear
for a = eye(4)
    disp(a)          % 依次显示矩阵 a 的各列，第 1 次执行显示矩阵 a 的第 1 列
end
```

4. while 循环结构

while 循环结构的使用格式如下：

```
while 表达式
    语句组
end
```

例 1.10 读取例 1.7 的脚本文件 eq_ex1_7_1.m 的全部字符.

```
clc, clear
f = fopen('eq_ex1_7_1.m','r')                % 以只读的方式打开文件
```

```
s = [];                                  % 初始化
while ~feof(f)                           % 判断是否为文件尾
    row = fgetl(f);                      % 从文件中读入一行
    fprintf('row = %s\n',row)            % 显示读入的一行字符串
    if isempty(row)                      % 是否为空行
        break                            % 如果是空行,则退出 while 循环
    end
    s = strcat(s,row);                   % 如果不是空行,则连接字符串
end
disp(s)                                  % 显示读取的全部字符串
```

5. try 试探结构

try 试探结构的一般使用格式如下：

```
try
    语句组 1
catch
    语句组 2
end
```

说明：试探结构首先试探性地执行指令语句组 1，如果在此语句组执行过程中出现错误，则将错误信息赋给保留的 lasterr 变量，并放弃这组语句，转而执行语句组 2 中的语句. 若语句组 2 执行过程中又出现错误，MATLAB 将终止该结构.

例 1.11 由两个子矩阵 a 和 b 构成一个大矩阵 c，若 a 和 b 的维数不匹配时，生成细胞数组 c.

```
clc, clear
a = rand(3), b = randi([1,5],4)
try
    c = [a,b]
catch
    lasterr                              % 显示出错原因
    c = {a,b}                            % 构造细胞数组
end
```

1.3.3 MATLAB 程序的调试

MATLAB 程序出错主要为两类：

(1) 格式错误，如缺“(”或“)”等，在运行时可检测出大多数该类错误，并指出错在哪一行.

(2) 算法错误，逻辑上的错误，不易查找，遇到此类错误时需耐心. 一般可考虑如下方法：

- 删除句尾分号，显示中间结果；
- 在适当位置加上 keyboard 语句，使程序暂停；
- 在函数定义行之前加上%，注释掉，使之变成脚本语言；
- 使用 MATLAB 调试器，设置断点或单步执行，使用一些调试和分析工具.

熟练掌握并运用调试方法，能提高编程的效率.

1. 直接调试法

下面给出直接调试法的两个例子.

例 1.12 M 文件的 MATLAB 调试.

编写文件名为 eq_ex1_12 的脚本文件：

```
clc, clear
a = rand(4,5), b = randi([1,10],4,5)
c = a * b                                    % 矩阵的乘法运算
```

在命令窗口中输入文件名并执行：

```
>> eq_ex1_12
Error using *
Inner matrix dimensions must agree.
Error in eq_ex1_12 (line 3)
c = a * b                                    % 矩阵的乘法运算
```

下画线为 MATLAB 提供了超链接，用鼠标单击超链接即可进入该文件内容出错的第 3 行，用户要根据 MATLAB 提供的出错原因仔细查找. 本例给出的错误原因是乘法“*”的矩阵维数不匹配，这里的两个矩阵是同维数的，只能进行两个矩阵的点乘运算 c=a. * b，纠错时，在乘号前面加上一个点即可.

对于函数文件，程序运行的错误同样在命令窗口中显示，待用户将错误的程序修改后函数文件才能继续运行.

例 1.13 函数文件的调试.

编写文件名为 myarea. m 的函数文件：

```
function s = myarea(r);                      % 计算圆的面积的函数
s = pi * r^2;
```

编写文件名为 eq_ex1_13. m 的脚本文件：

```
clc, clear
r = [4 6 8];                                 % 三个圆周的半径
s = myarea(r)                                % 计算三个圆的面积
```

在命令窗口中运行脚本文件 eq_ex1_13. m，给出如下提示：

```
Error using ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
Error in myarea (line 2)
s = pi * r^2;
Error in eq_ex1_13 (line 3)
s = myarea(r)                                % 计算三个圆的面积
```

单击函数第 2 行带下画线的超链接，可实现函数的查询和修改. 本例将函数文件 myarea. m 的第二行修改为逐个元素的 2 次幂，即 s=pi * r. ^2 即可.

2. M 文件调试器

MATLAB 的 M 文件调试器向用户提供了简洁的调试工具栏，如图 1-1 所示.

调试工具栏的选项大体分为两种：断点设置和单步运行调试.

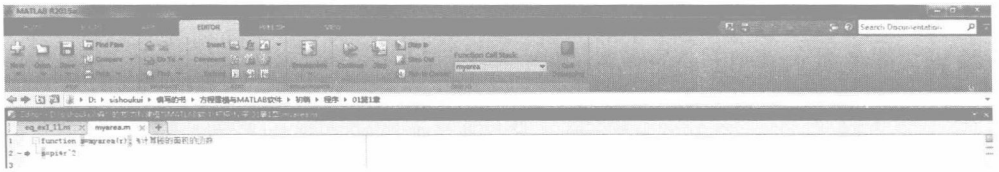

图 1-1 工具栏中调试图标

(1) 断点设置

有关断点操作的选项有 4 个，如图 1-2 所示.

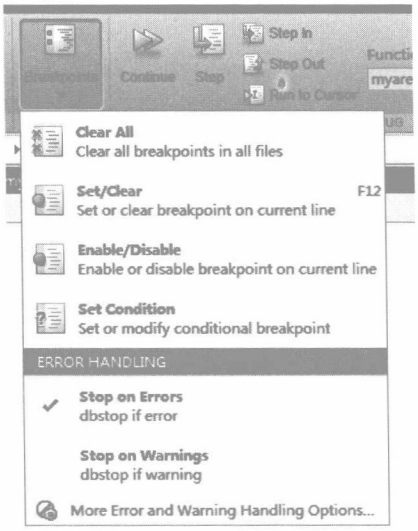

图 1-2 断点设置选项

Clear All：清除所有 M 文件中的断点.

Set/Clear：在光标所在行开始设置或清除断点.

Enable/Disable：将当前行的断点设置为有效或无效.

Set Condition：在光标所在行开头设置或修改条件断点，用于设置在满足什么条件时，此处断点有效. 选择此选项，会打开条件断点设置对话框.

(2) 单步运行调试

控制单步运行调试的选项有 4 个.

Step：在调试模式下，执行 M 文件的当前行.

Step In：在调试模式下，执行 M 文件的当前行，如果 M 文件当前行调用了另一个函数，那么进入该函数内部.

Step Out：当在调试模型下执行 Step In 进入某个函数内部之后，执行 Step Out 可以完成函数剩余部分的所有代码，并退出函数，暂停在进入函数内部前的 M 文件所在行末尾.

Run to Cursor：运行当前的 M 文件到光标所在处.

利用调试器调试步骤如下：

① 设置断点.

② 运行程序，检查中间结果. 当程序运行到断点处时，在断点和文本之间将会出现一个绿色的箭头，表示该程序运行至此停止. 在命令窗口的>>后输入变量名，检查变量的值，据此可以分析判断程序的正确性.

③ 继续运行. 选择工具栏中的 Continue 命令，程序继续运行，在断点处又暂停，这时又可输入变量名，检查变量的值. 如此重复，一直到发现问题为止.

④ 最后选择工具栏中的 Set/Clear 命令清除已设置的断点.

1.4 符号运算

符号运算又称计算机代数，通俗地讲就是用计算机推导数学公式，如对表达式进行因式分解、化简、微分、积分、解代数方程、求解常微分方程等. 与数值运算相比，符号计算存在以

下的特点：(1)运算以推理方式进行，因此不受截断误差和累积误差问题的影响；(2)符号计算的速度比较慢.

我们称 R2008a 及其之前的版本为低版本，R2008b 及其之后的版本为高版本. 下面我们主要介绍高版本的符号运算命令，本节的最后给出高低版本部分符号运算命令的对比.

1.4.1 符号对象的创建

在 MATLAB 中，提供了两个创建符号对象的函数，即 sym 和 syms 函数.

1. sym 函数

sym 函数用来建立单个符号量，函数的调用格式为

```
var = sym('var')          % 创建一个符号变量 var.
symexpr = sym(h)          % 通过匿名函数的函数句柄创建符号表达式.
A = sym('a', [m,n])       % 创建一个 m×n 矩阵的符号变量.
A = sym('a', n)           % 创建一个 n×n 矩阵的符号变量.
sym(Num)                  % 将一个数或数值矩阵转化为符号形式.
sym(Num,flag)             % 将一个数或数值矩阵转化为符号形式. 参数 flag 为转换的符号对象的格
                            式类型,flag 可以有如下选择:
```

- r：表示有理格式.
- d：表示十进制格式.
- e：表示带估计误差格式.
- f：表示浮点格式.

f(arg1,…,argN)＝sym('f(arg1,…,argN)')　%根据 f 指定的输入参数 arg1,…,argN 创建符号对象 f(arg1,…,argN).

assume(var,set)　%设置 var 属于 set，其中 var 为已存在的一个符号变量. set 的取值有如下几种方式：

- real：限定 var 为实型符号变量.
- integer：限定 var 表示整数的符号变量.
- rational：限定 var 表示有理数的符号变量.

```
assumptions               % 查看当前所有符号变量的属性.
assumptions(var)          % 查看已存在符号变量 var 的属性.
```

例 1.14　利用 sym 函数创建符号对象.

```
clc, clear
x = sym('x')                    % 定义符号变量 x
assume(x,'real')                % 设置符号变量 x 的属性
y = sym('y')                    % 定义符号变量 y
assume(y,'integer')             % 设置符号变量的属性
r1 = assumptions(x)             % 显示符号变量 x 的属性
r2 = assumptions                % 显示当前所有符号变量的属性
r = sym(1/3)
f = sym(1/3,'f')
d = sym(1/3,'d')
e = sym(1/3,'e')
```

```
A = sym('A',[3,4])                    % 生成 3×4 的符号矩阵
assume(A,'rational')
r3 = assumptions(A)                   % 显示矩阵 A 元素的属性
B = sym('t_%d_%d',4)                  % 生成 4×4 的符号矩阵
C = sym('x%d%d',4)                    % 生成 4×4 的符号矩阵
h_expr = @(x)(sin(x) + cos(x));       % 定义匿名函数
sym_expr = sym(h_expr)                % 生成匿名函数对应的符号函数
```

2. syms 函数

sym 函数一次只能定义一个符号变量，使用不方便. MATLAB 提供了另一个函数 syms，一次可以定义多个符号变量. syms 函数的调用格式为

```
syms var1 … varN                      % 创建符号变量 var1 … varN.
syms var1 … varN set                  % 创建符号变量 var1 … varN,并指定符号对象的格式.
syms var1 … varN clear                % 清除前面已创建的符号对象 var1 … varN.
syms f(var1,…,varN)                   % 创建符号函数 f,函数中包含符号变量 var1 … varN.
```

例 1.15 利用 syms 函数创建符号对象.

```
clc, clear
syms x y integer                      % 定义整型符号变量 x 和 y
assumptions                           % 显示当前工作空间中所有符号变量的属性
syms s(t) f(x,y)                      % 定义两个符号函数
s(t) = [t^2, t^3,t]                   % 如果 t 为矩阵,是.^运算.
f(x,y) = x^2 + y^3
t = [1 2 1;1 1 1]
s = s(t), s{:}                        % 显示细胞数组的所有元素
```

1.4.2 符号表达式的基本操作

1. 符号对象和数值对象的转换

借助于命令 sym(数值常数)可以将数值型数据转换为符号型数据. MATLAB 提供了相应的命令可以将符号型数据转换为数值型数据，便于用户进行相关的数值计算.

符号常量可以应用函数 double 转换为数值对象，其调用格式为

```
b = double(a)                         % 把符号常量 a 转换为数值对象 b.
```

例 1.16 把符号型数据转换为数值型数据.

```
clc, clear
a = sym('[sqrt(2),3;sqrt(3), log(pi)]')   % 生成符号矩阵
b = eig(a)                                % 求符号矩阵 A 的特征值
b = double(b)                             % 把符号值转换为数值型数据
```

2. 符号数据的精度控制

符号运算与数值计算比较，其最大的特点是符号运算的完全准确性，即在运算过程中不存在截断误差和累积误差. 但这种准确性是以牺牲计算速度和增加内存为代价的，并且一个符号数的表示很复杂，也不如用小数表示符号数直观. 为了兼顾计算速度和精度，节约内存，MATLAB 针对符号运算提供了“变精度”算法. 这种“变精度”算法由函数 digits 和 vpa

试读结束：需要全本请在线购买： www.ertongbook.com