

# Matlab

# 目录

数组与矩阵分析.....	15
数组及其函数.....	15
数组的建立和操作.....	15
数组的算术运算.....	15
数组的关系运算.....	15
数组的逻辑运算.....	15
数组信息的获取.....	16
矩阵的创建.....	16
矩阵的基本操作.....	17
矩阵的扩展.....	17
矩阵的块操作.....	17
矩阵中元素的删除.....	17
矩阵的转置.....	17
矩阵的旋转.....	18
矩阵的翻转.....	18
矩阵尺寸的改变.....	18
矩阵的基本数值运算.....	19
矩阵的加减运算.....	19
矩阵的乘法.....	19
矩阵的除法.....	19
矩阵元素的查找.....	20
矩阵元素的排序.....	21
矩阵元素的求和.....	21
矩阵元素的求积.....	22
矩阵元素的差分.....	23
特殊矩阵的生成.....	23
全零矩阵.....	23
全 1 矩阵.....	24
单位矩阵.....	24
0-1 间均匀分布的随机矩阵.....	25
标准正态分布随机矩阵.....	25
魔方矩阵.....	25
范德蒙矩阵.....	26
希尔伯特矩阵.....	26
托普利兹矩阵.....	27
伴随矩阵.....	27
帕斯卡矩阵.....	27
矩阵的特征和线性代数.....	28
方阵的行列式.....	28
特征值、特性向量和特性多项式.....	28
对角阵.....	29
上三角阵和下三角阵.....	29

矩阵的逆和伪逆	30
矩阵的秩	31
矩阵的迹	31
矩阵的范数	32
矩阵的条件数	32
矩阵的标准正交基	33
矩阵的超越函数	33
稀疏矩阵	34
矩阵存储方式	34
产生稀疏矩阵	35
特殊稀疏矩阵	36
矩阵的分解	37
Cholesky 分解	37
LU 分解	38
QR 分解	38
SVD 分解	39
Schur 分解	39
Hessenberg 分解	40
本章小结	40
字符串分析	41
字符串处理函数	41
字符串基本属性	41
字符串的构造	42
字符串的比较	43
字符串的查找和替换	43
字符串的转换	44
字符串的其他操作	45
字符的分类	46
字符串的执行	46
其他操作	47
本章小结	47
MATLAB 数据分析	48
多项式及其函数	48
多项式的建立	48
多项式的求值与求根	49
多项式的乘法和除法	50
多项式的导数和积分	50
多项式的展开	52
多项式的拟合	52
曲线拟合图形用户接口	53
插值	54
一维插值	54
二维插值	55
样条插值	57

高维插值 .....	57
函数的极限 .....	58
极限的概念 .....	58
求极限的函数 .....	58
本章小结 .....	59
积分和微分运算 .....	60
数值积分 .....	60
定积分概念 .....	60
利用梯形求面积 .....	60
利用矩形求面积 .....	61
单变量数值积分求解 .....	62
双重积分求解 .....	63
三重积分求解 .....	63
常微分方程 .....	64
常微分方程符号解 .....	64
常微分方程的数值解 .....	65
函数的极小值和零点 .....	65
一元函数的最小值 .....	66
多元函数的最小值 .....	66
一元函数的零点 .....	67
本章小结 .....	67
概率和数理统计 .....	68
随机数的产生 .....	68
二项分布随机数据的产生 .....	68
泊松分布 .....	69
指数分布随机数据的产生 .....	69
均匀分布随机数据的产生 .....	69
正态分布随机数据的产生 .....	70
其他常见分布随机数据的产生 .....	71
概率密度函数 .....	71
常见离散分布的概率密度函数 .....	71
常见连续分布的概率密度函数 .....	72
三大抽样分布的概率密度函数 .....	74
随机变量的数字特征 .....	75
平均值和中位数 .....	75
数据的排序 .....	77
期望和方差 .....	78
常见分布的期望和方差 .....	79
协方差和相关系数 .....	80
偏斜度和峰度 .....	81
参数估计 .....	82
点估计 .....	82
区间估计 .....	82
假设检验 .....	83

方差已知时正态总体均值的假设检验 .....	83
方差未知时正态总体均值的假设检验 .....	84
两个正态分布均值差的检验 .....	84
两个分布一致性检验 .....	84
方差分析 .....	85
单因素方差分析 .....	85
双因素方差分析 .....	86
统计图绘制 .....	86
正整数的频率表 .....	86
样本数据的盒图 .....	87
最小二乘拟合直线 .....	88
正态分布的概率图 .....	89
经验累积分布函数图 .....	89
威布尔概率图 .....	90
分位数-分位数图 .....	91
当前图形加一条参考线 .....	92
当前图形加多项式曲线 .....	92
样本的概率图形 .....	93
带有正态密度曲线的直方图 .....	94
在指定区间绘制正态密度曲线 .....	94
本章小结 .....	95
MATLAB 符号计算 .....	96
符号运算入门 .....	96
符号变量的创建 .....	96
符号变量的创建 .....	96
符号函数和符号方程 .....	98
简单实例分析 .....	98
求解一元二次方程的根 .....	98
求导数 .....	99
计算不定积分 .....	99
计算定积分 .....	99
求解一阶微分方程 .....	99
符号运算精度 .....	100
符号表达式的操作 .....	100
符号表达式的基本运算 .....	100
符号表达式的常用操作 .....	101
符号表达式的化简 .....	102
符号表达式的替换 .....	103
反函数运算 .....	104
复合函数运算 .....	104
符号矩阵的计算 .....	105
符号矩阵的生成 .....	105
符号矩阵的四则运算 .....	106
符号矩阵的线性代数运算 .....	107

符号微积分	109
符号表达式的微分运算	109
符号表达式的极限	109
符号表达式的积分	110
级数的求和	110
泰勒级数	111
符号表达式积分变换	111
Fourier 变换及其反变换	112
Laplace 变换及其反变换	112
Z 变换及其反变换	113
符号方程的求解	114
符号代数方程组的求解	114
微分方程的求解	115
符号函数的图形绘制	116
符号函数曲线的绘制	116
符号函数的三维网格图	118
符号函数的等值线图	121
符号函数的三维彩色曲面图	122
图形化符号函数计算器	124
单变量符号函数计算器	124
泰勒级数逼近计算器	125
Maple 接口	126
利用 sym 函数调用 Maple 函数	126
利用 maple 函数调用 Maple 函数	126
本章小结	127
二维数据可视化	128
MATLAB 绘图	128
基本绘图函数	128
子图绘制	130
叠加图绘制	131
设置坐标轴	132
网格线和边框	132
坐标轴的缩放	133
图形的拖拽	134
数据光标	134
绘制直线	135
极坐标绘图	135
对数和半对数坐标系绘图	136
双纵轴绘图	137
图形的窗口和标注	137
图形窗口	138
图形标注概述	138
图形标题	139
坐标轴标题	140

图例 .....	141
颜色条 .....	142
文本框标注 .....	143
获取和标记数据点 .....	144
特殊图形的绘制 .....	145
函数绘图 .....	145
柱状图 .....	146
饼状图 .....	147
直方图 .....	148
面积图 .....	148
散点图 .....	149
等高线绘图 .....	150
误差图 .....	150
填充图 .....	151
火柴杆图 .....	151
阶梯图 .....	151
罗盘图 .....	152
羽毛图 .....	152
向量场图 .....	153
彗星图 .....	154
伪彩色绘图 .....	154
图形句柄 .....	154
图形句柄 .....	155
坐标轴句柄 .....	156
本章小结 .....	158
三维数据可视化 .....	159
创建三维图形 .....	159
三维图形概述 .....	159
三维曲线图 .....	160
三维曲面图 .....	160
特殊三维绘图 .....	163
非网格数据绘图 .....	168
创建三维片块模型 .....	169
三维隐函数绘图 .....	169
三维图形的显示 .....	170
设置视角 .....	170
色彩控制 .....	172
光照效果 .....	175
Camera 控制 .....	177
图形绘制实例 .....	178
图形的输出 .....	179
本章小结 .....	180
MATLAB 基本编程 .....	181
MATLAB 编程概述 .....	181

M 文件的创建 .....	181
脚本 M 文件 .....	181
函数 M 文件 .....	182
函数的参数传递 .....	183
流程控制 .....	184
变量 .....	184
顺序结构 .....	185
分支结构 .....	185
循环结构 .....	186
try...catch 语句 .....	188
人机交互函数 .....	189
函数类型 .....	190
主函数 .....	190
子函数 .....	191
嵌套函数 .....	191
私有函数 .....	192
重载函数 .....	193
匿名函数 .....	194
函数句柄 .....	196
P 码文件和 ASV 文件 .....	197
P 码文件 .....	197
ASV 文件 .....	198
本章小结 .....	198
程序调试和编程技巧 .....	199
M 文件调试 .....	199
出错信息 .....	199
直接调试法 .....	200
工具调试法 .....	200
错误处理 .....	202
M 文件性能分析 .....	203
Code Analyzer 工具 .....	203
Profiler 分析工具 .....	204
编程技巧 .....	205
程序执行时间 .....	205
编程技巧 .....	205
小技巧 .....	207
本章小结 .....	207
Simulink 基本知识 .....	208
Simulink 概述 .....	208
Simulink 的概念 .....	208
Simulink 的应用和特点 .....	209
Simulink 的基本操作 .....	209
启动 Simulink .....	209
选择模块 .....	210

模块的连接	210
模块参数的设置	212
仿真器设置	212
运行仿真	213
常用的模块库	213
Simulink 常用模块子集	214
连续时间模块子集	214
非连续时间模块子集	215
离散时间模块子集	216
逻辑和位操作子集	216
查表模块子集	217
数学运算模块子集	218
端口和子系统模块子集	218
信号特征模块子集	219
信号路径模块子集	219
Sinks 模块子集	220
信号源模块子集	220
用户定义模块子集	221
子系统及其封装	221
子系统	221
子系统的封装	222
Simulink 模型工作空间	222
本章小结	223
Simulink 建模和 S-函数	224
回调函数	224
模型回调函数	224
模块回调函数	225
运行仿真	225
仿真参数的设置	225
仿真的出错信息	226
使用 MATLAB 命令行运行仿真	226
模型的调试	227
Simulink 调试器	227
命令行调试	228
S-函数建模	228
S-函数介绍	228
S-函数工作原理	228
M 文件的 S-函数	229
S-函数实例分析	229
本章小结	230
GUI 编程开发	231
图形句柄	231
MATLAB 图形系统	231
图形句柄	231

图形对象的属性	232
图形对象	232
创建图形对象	232
获取对象的属性	233
设置对象的属性	233
对象的基本操作	233
root 根对象	234
figure 对象	235
axes 坐标轴对象	235
核心图形对象	236
用户接口对象	236
uicontrol 对象	236
uimenu 对象	237
uicontextmenu 对象	238
uitoolbar 对象	239
uibuttongroup 对象	239
uipanel 对象	239
uitable 对象	240
常用的对话框	240
消息对话框	241
错误对话框	241
警告对话框	242
帮助对话框	242
输入对话框	242
列表对话框	243
问题对话框	243
进度条设置对话框	244
路径选择对话框	245
文件打开对话框	245
文件保存对话框	246
页面设置对话框	246
打印预览对话框	246
打印对话框	247
颜色设置对话框	247
字体设置对话框	247
本章小结	248
GUIDE 工具建立 GUI 界面	249
采用 GUIDE 建立 GUI	249
图形用户界面概述	249
GUIDE 启动	249
GUI 对象的常用属性	251
GUI 的 M 文件	252
数据传递	252
简单实例分析	253

简单的按钮 .....	253
简单的滑动条 .....	253
弹出式菜单 .....	254
列表框 .....	254
加法运算 .....	254
坐标轴 .....	255
菜单 .....	255
快捷菜单 .....	255
综合实例分析 .....	255
声音的显示和播放 .....	256
语音实时采集和显示 .....	256
本章小结 .....	256
文件夹管理和文件 I/O 操作 .....	257
文件夹管理 .....	257
当前文件夹 .....	257
文件夹管理 .....	257
低级文件 I/O .....	258
打开和关闭文件 .....	259
读取和写入二进制文件 .....	259
逐行读取文本文件 .....	260
格式化读取和写入文件 .....	260
文件内部的位置控制 .....	262
高级文件 I/O .....	263
使用向导来导入数据 .....	263
带有分隔符的数据文件 .....	263
带有文本内容的数据文件 .....	263
利用函数 importdata() 带入数据 .....	264
利用函数 diary() 导出数据 .....	264
其他函数 .....	264
音频文件 .....	265
读取音频文件 .....	265
播放和录音 .....	266
写入音频文件 .....	267
图像文件 .....	267
读取和显示图像 .....	268
获取文件信息 .....	268
写入图片文件 .....	269
通过函数 image 显示图像 .....	269
视频文件 .....	270
读取视频文件 .....	270
获取文件信息 .....	271
写入视频文件 .....	271
表单文件 .....	272
读取表单文件 .....	273

获取表单文件信息 .....	273
写入表单文件 .....	274
Lotus 123 表单文件 .....	274
本章小结 .....	274
MATLAB 编译器 .....	275
编译器概述 .....	275
编译器的安装和配置 .....	275
查看系统的信息 .....	275
对编译器进行配置 .....	276
编译器的使用 .....	276
编译过程 .....	276
deploytool 工具 .....	277
MCR 的安装 .....	278
本章小结 .....	278
信号处理工具箱 .....	279
数字信号处理 .....	279
信号处理工具箱 .....	279
离散信号 .....	279
波形发生器 .....	281
改变信号的采样率 .....	283
改变信号的采样率 .....	283
降低信号的采样率 .....	284
提高信号的采样率 .....	285
FIR 数字滤波器 .....	286
FIR 滤波器介绍 .....	286
常用的窗函数 .....	286
基于窗函数的 FIR 滤波器 .....	288
基于频率采样的 FIR 滤波器 .....	288
FIR 滤波器实例分析 .....	289
IIR 数字滤波器 .....	289
IIR 数字滤波器介绍 .....	290
IIR 数字滤波器 .....	290
IIR 数字滤波器阶数的选择 .....	291
IIR 滤波器实例分析 .....	292
非参数法功率谱估计 .....	293
周期图法功率谱估计 .....	293
Welch 法功率谱估计 .....	293
参数模型法功率谱估计 .....	294
AR 模型功率谱估计 .....	294
Yule-Walker 法功率谱估计 .....	295
Burg 法功率谱估计 .....	295
协方差法功率谱估计 .....	296
子空间法功率谱估计 .....	296
特征向量法功率谱估计 .....	296

MUSIC 算法功率谱估计 .....	297
本章小结 .....	297
小波分析工具箱 .....	298
小波分析 .....	298
小波分析简介 .....	298
连续小波变换 .....	298
离散小波变换 .....	298
多分辨分析 .....	299
小波包分解 .....	299
常用的小波 .....	299
一维小波分解和重构 .....	300
一维连续小波分解 .....	300
一维离散小波分解和重构 .....	300
二维小波分解和重构 .....	302
二维小波的单层分解和重构 .....	303
二维小波的多层分解和重构 .....	303
小波包分析 .....	304
一维小波包的分解和重构 .....	304
二维小波包的分解和重构 .....	306
小波能量和小波熵 .....	307
小波工具箱的 GUI 工具 .....	307
小波工具箱介绍 .....	307
小波和小波包的显示工具 .....	308
一维小波分析工具 .....	308
二维小波分析工具 .....	309
本章小结 .....	309
图像处理工具箱 .....	310
图像处理基础 .....	310
数字图像处理 .....	310
图像处理工具箱介绍 .....	310
MATLAB 中图像类型 .....	310
MATLAB 图像类型转换 .....	311
图像的读写和显示 .....	312
图像浏览器 .....	313
图像的常用操作 .....	313
图像基本运算 .....	314
图像的直方图和直方图均衡化 .....	315
图像的对比度增强 .....	315
图像的插值 .....	316
图像中添加噪声 .....	316
图像剪切 .....	317
图像的变换技术 .....	317
图像的傅里叶变换 .....	317
图像的离散余弦变换 .....	318

图像增强技术 .....	318
像数值及其统计 .....	318
图像的滤波 .....	319
图像的边缘检测 .....	320
数字图像的采集 .....	320
USB 摄像头设置 .....	320
图像和视频的采集 .....	321
imaqtool 工具 .....	322
本章小结 .....	323
神经网络工具箱 .....	324
人工神经网络概述 .....	324
人工神经网络介绍 .....	324
人工神经元 .....	324
MATLAB 神经网络工具箱 .....	325
感知器神经网络 .....	325
感知器神经网络 .....	325
设计实例分析 .....	325
线性神经网络 .....	326
线性神经网络 .....	326
设计实例分析 .....	327
BP 网络 .....	328
BP 网络介绍 .....	328
BP 网络的创建 .....	328
BP 网络设计实例 .....	328
径向基神经网络 .....	330
径向基神经网络 .....	330
广义回归神经网络 .....	330
概率神经网络 .....	331
神经网络 GUI 工具 .....	331
nntool 工具 .....	331
nprtool 工具 .....	332
本章小结 .....	332
MATLAB 与 Word 和 Excel 的接口 .....	333
MATLAB 与 Word 混合使用 .....	333
Notebook 安装 .....	333
Notebook 的使用环境 .....	333
Notebook 实例分析 .....	334
MATLAB 与 Excel 混合使用 .....	334
Excel link 的安装 .....	334
设置启动方式 .....	335
启动和终止 Excel link .....	335
Excel link 工具栏 .....	335
实例分析 .....	336
本章小结 .....	336

# 数组与矩阵分析

Matlab 语言最基本和最重要的功能就是进行矩阵运算，所有的数值功能都以矩阵为基本单元来实现。本章将对 Matlab 中的数组和矩阵及其运算进行详细介绍。

## 数组及其函数

建立数组，以及数组的常用操作等，包括数组的算术运算、关系运算和逻辑运算，以及数组信息的获取等。

### 数组的建立和操作

使用`[]`、`,`、空格及`;`来创建数组。数组中同一行的元素之间用逗号或空格进行分割，不同行之间用分号进行分割。

空数组，不含有任何元素，可以用于数组的声明或者清空等。创建空数组只要把变量赋值为一对方括号即可。

数组是有方向的，一维数组包括行向量和列向量。

### 数组的算术运算

数组运算是从数组的单个元素出发，针对每个元素进行的运算。一维数组的基本算术运算有：`+`、`-`、`.*`、`./`、`.\`和`^`。

数组加减运算规则，要求数组 A 和 B 的维数相同。对于元素进行相加减。

数组乘除运算规则，要求数组 A 和 B 的维数相同。对于元素进行乘除。(2.3  
是正常的除法哦)

### 数组的关系运算

`<`、`<=`、`>`、`>=`、`==`、`~=`

比较的量是两个维数相同的数组时，比较两数组相同位置的元素，并给出比较结果。比较的量是标量和数组时，把标量与数组的每一个元素逐个比较。

A (A>10) =10

B (B==5) =100(新的索引方式)

### 数组的逻辑运算

`&`（与）、`|`（或）、`~`（非）。数组对应元素。

## 数组信息的获取

isempty(A) 是否为空数组  
isscalar (A) 检测数组是否为单个元素的标量  
isvector(A) 检测数组是否为行向量或者列向量  
isrow () 检测数组是否为列向量  
iscolumn () 检测数组是否为行向量  
issparse() 检测数组是否为稀疏矩阵  
size() 获取数组的行数和列数  
length() 获取一维数组的长度，如果是二维数组，则返回行数和列数中的较大者  
ndims() 计算数组的维度  
a = 5  
ndims(a) 为 2 (二维：一行一列)  
whos 来获取数组的大小，以及占用内存的多少  
find() 返回关系表达式为真的元素的下标//下标！  
sort() 默认按照升序排列，返回值为排序后的数组  
//sort(A, "ascend") sort(A, "descend")

## 矩阵的创建

直接输入矩阵：把矩阵的元素放到[]中，每行中的元素用空格或者逗号隔开，每列用分号隔开；

函数生成矩阵：eye() 生成单位矩阵；

M 文件建立矩阵：

从外部数据文件中导入矩阵：imread() 读取图片，得到图像数据的二维矩阵。

//使用矩阵下标可以非常灵活的获取矩阵（行：行，列：列）

矩阵的元素按照列进行保存，先第一列，再第二列，直到结束。

//使用 sub2ind(size(A), n, m) 和 ind2sub(size(A), n) 进行单下标与双下标转换

A=[1:4;5:8;9:12;13:16]

B=A>5

A(A>5)=5

```
f1=find(A>4)  
A(find(A>4))=6
```

在程序中，对矩阵中的元素进行赋值。如果行或者列超出矩阵的大小，则 MATLAB 自动扩充矩阵的大小，然后再进行赋值，扩充部分用零填充。

## 矩阵的基本操作

在 MATLAB 中，矩阵是计算的基本单元，有很多关于矩阵操作的函数。下面介绍常用的矩阵的扩展、块操作、转置、旋转和翻转，以及改变矩阵的大小等。

### 矩阵的扩展

可以通过数组的扩展，将多个小矩阵转换为大的矩阵。

C=cat(DIM, A, B): 行 1 列 2 // 列 1 行 2

C=vertcat(A, B): 在垂直方向上连接数组 A 与 B; DIM=1

C=horzcat(A, B): 在水平方向上连接数组 A 与 B; DIM=2

### 矩阵的块操作

repmat()、blkdiag()、kron()

repmat(): 进行数据块的复制

B=repmat(A, m, n) // 以 A 作为基本元素

B=repmat(A, m)

blkdiag(): 将多个矩阵作为对角块产生新的矩阵

Y=blkdiag(A, B)

Y=blkdiag(A, B, ...)

### 矩阵中元素的删除

利用空矩阵删除矩阵的元素：X=[] // clear X 是从工作空间中删除  
// 是直接删除不是赋值为 0!

### 矩阵的转置

采用转置操作符（'），从而实现矩阵 A 的 Hermitian 转置为 A'  
// 如果矩阵中含有复数，则转化为共轭复数

矩阵真正的转置为 A.'；也可以使用 transpose(A) 实现

```
% chap3_35.m

clear all;
A=rand(2, 4)
A1=A'
A2=transpose(A)
B=[2+3i, 4+5i, 3; 2 4+i 5+3*i]
B1=B'
B2=B.'|
B3=ctranspose(B)
```

## 矩阵的旋转

矩阵的旋转可以采用转置的方法，也可以采用函数 rot90()

B=rot90(A)：将矩阵逆时针旋转 90°

B=rot90(A, k)：将矩阵逆时针旋转 90° 的 k 倍

```
% chap3_36.m
```

```
clear all;
A=rand(3, 4)
B=rot90(A)| I
C=rot90(A, 2)
D=rot90(A, 4)
```

## 矩阵的翻转

函数 fliplr(A)：实现矩阵左右翻转

函数 flipud(A)：实现矩阵上下翻转

函数 flipdim(A, k)：

```
% chap3_37.m
```

```
clear all;
A=rand(3, 4)
B=fliplr(A)| I
C=flipud(A)| I
D=flipdim(A, 1)
E=flipdim(A, 2)
```

## 矩阵尺寸的改变

在矩阵总元素保持不变的前提下，在 MATLAB 中使用函数 reshape() 改变矩阵的尺寸。

`Y=reshape(X, m, n)`: 将矩阵转换为 m 行 n 列的二维矩阵

```
% chap3_38.m  
clear all;  
X=[1:4;5:8]  
Y1=reshape(X, 1, 8)  
Y2=reshape(Y1, [4, 2])  
Y3=reshape(X, size(Y2))
```

## 矩阵的基本数值运算

矩阵的基本数值计算包括矩阵的加减乘除以及矩阵元素的查找、排序、求和和求积等

### 矩阵的加减运算

要求矩阵 A、B 的维数必须相同。矩阵的加减是矩阵中对应元素加减。如果有一个是标量，则将矩阵中的每一个元素和改标量进行加减运算。

```
% chap3_39.m
```

```
clear all;  
A=[1:4;5:8]  
B=[2 3 5 4;6 7 4 5]  
C=A-B  
D=A+100  
|
```

### 矩阵的乘法

矩阵 A 与 B 的乘法为 `A*B`，要求矩阵 A 的列数与矩阵 B 的行数相等

```
% chap3_40.m
```

```
clear all;  
A=[1:4;5:8]  
B=[2 3;1 3;6 7;4 5]  
C=A*B  
D=A.*B'  
E=A*5
```

### 矩阵的除法

方程组 `A*X=B` 的解： `X=A\B` (`inv(A)*B`)

方程组 `X*A=B` 的解： `X=B/A` (`B*inv(A)`)

```
//如果 X 不存在或不唯一，则系统显示警告信息
```

```
% chap3_41.m

clear all;
A=[2 1 2 3;7 3 1;9 4 2]
B=[3 5 7;2 12 4;2 7 4]
C1=A\B
C2=inv(A)*B
D1=B/A
D2=B*inv(A)           I
E1=A^3
E2=A*A*A
```

```
% chap3_42.m
```

```
clear all;
A=[1 2 3;7 3 1;9 4 2]
B=[3 5 7;2 12 4;2 7 4]
C1=A./B I
C2=A.\B
C3=A./2
D=A.^2
```

## 矩阵元素的查找

采用函数 find() 进行矩阵元素的查找。

i=find(A)：查找矩阵中的非零元素，返回这些元素的单下标

[i, j]=find(A)：查找矩阵中的非零元素，返回这些元素的双下标

```
% chap3_43.m
```

```
clear all;
A=[1 0 3;0 3 1;9 4 2]
B=find(A)           I
C=find(A>=3)
D=A(find(A>=3))
A(find(A==9))=100
```

## 矩阵元素的排序

在MATLAB中，矩阵元素的排序使用函数**sort( )**，该函数默认按照升序排列，返回排序后的矩阵。

**Y=sort(X)**: 该函数对矩阵按照升序进行排列。当X为向量时，返回由小到大排序后的向量；当X为矩阵时，返回X中各列按照由小到大排序后的矩阵。

**Y=sort(X, DIM)**: 该函数返回在给定的维数上的按照由小到大的顺序排序后的结果，当**DIM=1**时，按照列进行排序，当**DIM=2**时，按照行进行排序。

**Y=sort(X, DIM, 'MODE')**: 该函数可以指定排序的方式。参数**MODE**默认值为'**ascend**'，即按照升序进行排列；当**MODE**为'**descend**'时，对矩阵进行降序排列。

函数 **sort()**

```
% chap3_44.m  
  
clear all;  
A=[1 0 3;0 3 1;9 4 2]  
B=sort(A)  
C=sort(A, 2)  
D=sort(A, 'descend')  
E=sort(A, 2, 'descend')
```

## 矩阵元素的求和

在MATLAB中，进行矩阵中元素求和时采用函数**sum( )**和**cumsum( )**。其调用格式为：

**Y=sum(X)**: 该函数对矩阵X的元素求和，返回矩阵中各列元素的和组成的向量。

**Y=sum(X, DIM)**: 该函数返回在给定的维数**DIM**上的元素的和，当**DIM=1**时，计算矩阵各列元素的和，当**DIM=2**时，得到矩阵各行元素的和。

函数**cumsum( )**的调用格式和函数**sum( )**类似，不同之处是函数**cumsum( )**的返回值为矩阵。

函数 **sun()**, **cumsum()**//逐步求和

```
% chap3_45.m

clear all;
A=[1 2 3 4;5 6 7 8;1 1 1 1]
B1=sum(A)
B2=sum(A, 2)
C1=cumsum(A)
C2=cumsum(A, 2)
D=sum(sum(A))
```

## 矩阵元素的求积

在MATLAB中，进行矩阵中元素求积时采用函数**prod( )**和**cumprod( )**。其调用格式为：

**Y=prod(X)**: 该函数对矩阵的元素求积，返回矩阵X中各列元素的积组成的向量。

**Y=prod(X, DIM)**: 该函数返回在给定的维数上的元素的积，当**DIM=1**时，计算矩阵各列元素的积，当**DIM=2**时，得到矩阵各行元素的积。

函数**cumprod( )**的调用格式和函数**prod( )**类似，不同之处是函数**cumprod( )**的返回值为矩阵。

## 函数 prod(), cumprod()

```
% chap3_46.m

clear all;
A=[1 2 3 0;2 6 7 8;1 2 3 5]
B1=prod(A)
B2=prod(A, 2)
C1=cumprod(A)
C2=cumprod(A, 2)
```

## 矩阵元素的差分

在MATLAB中，利用函数**diff( )**计算矩阵的差分，该函数的调用格式为：

**Y=diff(X)**: 该函数计算矩阵各列的差分。

**Y=diff(X, N)**: 该函数计算矩阵各列的N阶差分。

**Y=diff(X, N, DIM)**: 该函数计算矩阵在方向**DIM**上的N阶差分。当**DIM=1**时，计算矩阵各列元素的差分，当**DIM=2**时，得到矩阵各行元素的差分。

函数 diff()计算矩阵的差分//字面意思！差！

```
% chap3_47.m  
  
clear all;  
A=[1 2 3 0;2 6 7 8;1 2 3 5]  
B1=diff(A)  
B2=diff(A, 2)  
B3=diff(A, 1, 1) I  
B4=diff(A, 1, 2)
```

## 特殊矩阵的生成

在MATLAB中，有许多用于创建矩阵的函数。通过这些函数可以创建二维矩阵，甚至更高维的矩阵。下面对这些产生通用特殊矩阵的函数进行介绍。

### 全零矩阵

函数 zeros()

在MATLAB中，采用函数**zeros( )**产生全零矩阵，该函数的调用格式为：

**A=zeros(N)**: 该函数产生N行N列的全零矩阵。

**A=zeros(M, N)**: 该函数产生M行N列的全零矩阵。

**A=zeros(M, N, P,...)**: 该函数产生M\*N\*P\*...的全零矩阵。

**A=zeros(size(B))**: 该函数产生和矩阵B维数相同的全零矩阵。

```
% chap3_48.m  
  
clear all;  
A=zeros(4)  
B=zeros(2, 3) I  
C=zeros(size(B))
```

## 全1矩阵

函数 ones()

在MATLAB中，采用函数ones( )产生全1矩阵，该函数的调用格式和函数zeros( )基本一致。

```
% chap3_49.m
```

```
clear all;  
A=ones(3)  
B=ones([3, 2])  
C=ones(size(A))
```

## 单位矩阵

在MATLAB中，采用函数eye( )产生单位矩阵，该函数的调用格式为：

A=eye(N): 该函数产生N行N列的单位矩阵。

A=zeros(M, N): 该函数产生M行N列的矩阵，对角线元素为1，其余元素均为0。

A=zeros(size(B)): 该函数产生和矩阵B维数相同的单位矩阵。

```
% chap3_50.m
```

```
clear all;  
A=eye(3)  
B=eye([4, 3])  
C=eye(size(B))
```

## 0-1 间均匀分布的随机矩阵

在MATLAB中，采用函数rand( )产生0~1之间均匀分布的随机矩阵，该函数的调用格式为：

**A=rand(N):** 该函数产生N行N列的0~1之间均匀分布的随机矩阵。

**A=rand(M, N):** 该函数产生M行N列的0~1之间均匀分布的随机矩阵。

**A=rand(M, N, P,...):** 该函数产生M\*N\*P\*...的0~1之间均匀分布的随机矩阵。

**A=rand(size(B)):** 该函数产生和矩阵B维数相同的0~1之间均匀分布的随机矩阵。

```
% chap3_51.m
```

```
clear all;
A=rand(3)
B=rand([2, 3])
C=rand(size(B))
```

0-N 之间就乘以 N!

## 标准正态分布随机矩阵

在MATLAB中，采用函数randn( )产生均值为0，方差为1的标准正态分布随机矩阵。该函数的调用格式和函数rand( )基本一致。

```
% chap3_52.m
```

```
clear all;
A=randn(3)
B=randn([2, 3])
C=randn(size(B))
```

## 魔方矩阵

魔方矩阵中每行、每列及两条对角线上的元素和都相等。对于n阶魔方阵，其元素由1,2,3,...,n<sup>2</sup>组成，共n<sup>2</sup>个整数。

在MATLAB中，通过函数magic(n)，求n阶魔方矩阵。

```
% chap3_53.m
```

```
clear all;
A=magic(2)
B=magic(3)
C=magic(4)|
```

## 范得蒙矩阵

范得蒙（Vandermonde）矩阵最后一列全为1，倒数第二列为一个指定的向量，其他各列是其后列与倒数第二列的点乘积。可以用一个指定向量生成一个范得蒙矩阵。

在MATLAB中，通过函数vander(V)生成以向量V为基础向量的范得蒙矩阵。

```
% chap3_54.m
```

```
clear all;
A=vander([1;2;4;6])
B=vander([1 2 3 4])
```

## 希尔伯特矩阵

在MATLAB中，通过函数hilb( )生成希尔伯特（Hilbert）矩阵。该函数的调用格式为： hilb(n)，产生n阶的希尔伯特矩阵。希尔伯特矩阵是一种病态矩阵，矩阵中任何一个元素发生微小的变化，整个矩阵的值和逆矩阵都发生巨大的变化。

MATLAB中，通过函数函数invhilb( )求希尔伯特矩阵的逆矩阵，该函数的调用格式为： invhilb(n)，该函数产生n阶希尔伯特矩阵的逆矩阵。

```
% chap3_55.m
```

```
clear all;
A=hilb(3)
B=hilb(4)
C=invhilb(3)
D=invhilb(4)
A*C           I
```

## 托普利兹矩阵

托普利兹（Toeplitz）矩阵除第一行和第一列外，其它每个元素都与左上角的元素相同。在MATLAB中，通过函数 **toeplitz( )**生成托普利兹矩阵。该函数的调用格式为：

**toeplitz(x)**: 该函数用向量x生成一个对称托普利兹矩阵。

**toeplitz(x, y)**: 该函数产生一个以x为第一列，y为第一行的托普利兹矩阵。x和y均为向量，两者不必等长。需要注意的是，向量x和y的第一个元素必须相等。

```
% chap3_56.m
```

```
clear all;
A=toeplitz(3:6)
x=3:8
y=3:7
B=toeplitz(x, y)
```

## 伴随矩阵

在MATLAB中，通过函数**compan( )**产生伴随矩阵，该函数的调用格式为：

**compan(p)**，其中p为多项式的系数向量，高次幂系数排在前，低次幂排在后。

```
% chap3_57.m
```

```
clear all;
p1=[2 3 4]
A=compan(p1)
p2=[1 2 3 4]
B=compan(p2)

//嘶，感觉很奇怪 (qaq) !
```

## 帕斯卡矩阵

二次项展开后的系数随n的增大组成一个三角形表，称为杨辉三角形。由杨辉三角形表组成的矩阵称为帕斯卡（Pascal）矩阵。该函数的调用格式为：**pascal(n)**，产生一个n阶的帕斯卡矩阵。

```
% chap3_58.m
```

```
clear all;
A=pascal(4)
B=pascal(5)
```

T

## 矩阵的特征和线性代数

下面介绍矩阵的一些基本操作，包括矩阵的特征值、三角阵、对角阵、矩阵的翻转等，以及矩阵的一些特征，例如矩阵的秩、矩阵的迹和矩阵的范数等。最后介绍了矩阵的超越函数。

### 方阵的行列式

把一个方阵看作一个行列式，并对其按行列式的规则求值，这个值就称为矩阵所对应的行列式的值。

在MATLAB中，采用函数`det()`求方阵的行列式，该函数的调用格式为：`det(X)`，得到方阵X的行列式。

```
% chap3_59.m
```

```
clear all;
A=magic(3)
B=[1:3;2 5 7;3 8 7]
y1=det(A)
y2=det(B)
```

### 特征值、特性向量和特性多项式

在MATLAB中，计算矩阵A的特征值和特征向量的函数是`eig()`，该函数的调用格式为：

`E=eig(A)`：求矩阵A的全部特征值，组成向量E。

`[V, D]=eig(A)`：该函数计算矩阵的特征值和特征向量，返回值V和D为两个方阵。方阵V的每一列为一个特征向量，方阵D为对角矩阵，对角线上的元素为特征值。

```
% chap3_60.m
```

```
clear all;
A=magic(3)
E=eig(A)
[V, D]=eig(A)
```

```
% chap3_61.m
```

```
clear all;
p=[3 5 2 1]
A=compan(p)
x1=eig(A)
x2=roots(p)
```

//伴随矩阵特征值就是多项式方程的根! (神奇)

## 对角阵

只有对角线上有非0元素的矩阵称为对角矩阵，对角线上的元素相等的对角矩阵称为数量矩阵，对角线上的元素都为1的对角矩阵称为单位矩阵。在MATLAB中，通过函数**diag( )**获取矩阵的对角线元素，该函数的调用格式为：

**diag(A)**：该函数用于提取矩阵A的主对角线元素，产生一个列向量。

**diag(A, k)**：该函数提取第k条对角线的元素，组成一个列向量。

```
% chap3_62.m
```

```
clear all;
A=rand(3, 4)
b1=diag(A)
b2=diag(A, 1)
b3=diag(A, 2)
```

## 上三角阵和下三角阵

三角阵可以分为上三角阵和下三角阵，所谓上三角阵，即矩阵的对角线以下的元素全为0的矩阵，而下三角阵则是对角线以上的元素全为0的一种矩阵。在MATLAB中，通过函数**triu( )**获取矩阵的上三角矩阵，该函数的调用格式为：

**triu(A)**，该函数返回矩阵A的上三角矩阵。

**triu(A, k)**：该函数返回矩阵A的第k条对角线以上的元素。

在MATLAB中，采用函数**tril( )**求矩阵的下三角矩阵，该函数的调用格式和函数**triu( )**完全相同，这里不再赘述。

```
% chap3_63.m
```

```
clear all;
A=rand(3,3)
B1=triu(A)
B2=tril(A)
B3=tril(A,1)
```

## 矩阵的逆和伪逆

对于方阵A，如果存在一个与其同阶的方阵B，使得

$A^*B=B^*A=E$ ，则称A和B互为逆矩阵。采用函数`inv()`求方阵的逆矩阵。

如果矩阵A不是一个方阵，或者是一个非满秩方阵时，没有逆矩阵，但可以找到一个与A的转置矩阵同型的矩阵B，

使得：

$$\begin{cases} A \times B \times A = A \\ B \times A \times B = B \end{cases}$$

此时，称矩阵B为矩阵A的伪逆，也称为广义逆矩阵。在

MATLAB中，求矩阵的广义逆矩阵的函数是`pinv()`。该函数的调用格式为：`pinv(A)`，该函数计算矩阵A的广义逆矩阵。

```
% chap3_64.m
```

```
clear all;
A=magic(3)
B=[1 3;2 6]
C=inv(A)
C*A
inv(B)
D=pinv(B)
B*D*B
```

## 矩阵的秩

矩阵的秩包括行秩和列秩，行秩和列秩相等。行秩为矩阵的行向量组成的极大无关组中行向量的个数，列秩为矩阵的列向量组成的极大无关组中列向量的个数。矩阵的秩反映了矩阵中各行向量之间和各列向量之间的线性关系。对于满秩矩阵，秩等于行数或列数，其各行向量或列向量都线性无关。

在MATLAB中，通过函数**rank( )**求矩阵的秩。该函数的调用格式为：**rank(A)**，该函数求矩阵的秩。

```
% chap3_65.m  
clear all;  
A=magic(3)  
B=[1 2 4;2 3 5;2 4 8]  
r1=rank(A)  
r2=rank(B)
```

## 矩阵的迹

矩阵的迹等于矩阵的对角线元素之和，也等于矩阵的特征值之和。

在MATLAB中，通过函数**trace( )**求矩阵的迹。该函数的调用格式为：**trace(A)**，求矩阵的迹。

函数 **trace()**

```
% chap3_66.m  
clear all;  
A=magic(3)  
t1=trace(A)  
eig(A)  
t2=sum(eig(A))|
```

## 矩阵的范数

矩阵的范数常用的有3种。在MATLAB中，求矩阵范数的函数为**norm( )**，该函数的调用格式为：

**norm(X)**或**norm(X, 2)**：计算矩阵的2-范数，返回矩阵的最大奇异值。

**norm(X, 1)**：计算矩阵的1-范数，返回矩阵的列向元素和的最大值。

**norm(X, inf)**：计算矩阵的 $\infty$ -范数，返回矩阵的行向元素和的最大值。

**norm(X, 'fro')**：该函数计算矩阵的Frobenius范数。

```
% chap3_67.m  
clear all;  
A=[1 2 3;3 5 7;2 5 8]  
n1=norm(A, 1)  
n2=norm(A)  
n3=norm(A, inf)  
n4=norm(A, 'fro')  
n5=normest(A)
```

函数 normest()//估计值

## 矩阵的条件数

矩阵的条件数是用来判断矩阵病态的一个量，矩阵的条件数越大，表明该矩阵越病态，否则该矩阵越良态。**Hilbert**矩阵就是有名的病态矩阵。

在MATLAB中，通过函数**cond( )**求矩阵的条件数，该函数的调用格式为：

**cond(X, 1)**：该函数计算矩阵X的1-范数下的条件数。

**cond(X)**或**cond(X, 2)**：该函数计算矩阵X的2-范数下的条件数。

**cond(X, inf)**：该函数计算矩阵X的 $\infty$ -范数下的条件数。

```
% chap3_68.m  
clear all;  
A=[1 2 3;3 5 7;9 5 8]  
c1=cond(A, 1)  
c2=cond(A, 2)  
c3=cond(A, inf)
```

## 矩阵的标准正交基

在MATLAB中，通过函数**orth( )**得到矩阵的标准正交基，该函数的调用格式为：**B=orth(A)**，矩阵B的列向量组成了矩阵A的一组标准正交基。

```
% chap3_69.m
```

```
clear all;
A=[1 2 3;3 5 7;9 5 8]
B=magic(3)
C=orth(A)
D=orth(B)
C'*C
eye(rank(A))
```

## 矩阵的超越函数

下面介绍矩阵的超越函数，主要包括：函数**sqrtm( )**、函数**logm( )**、函数**expm( )**和函数**funm( )**。

函数**sqrtm( )**用于计算矩阵的平方根。

函数**logm()**用于计算矩阵的自然对数。

函数**expm()**用于求矩阵的指数。

函数**funm( )**用于计算矩阵的超越函数值。

```
% chap3_70.m
```

```
clear all;
A=[3 2 3;6 5 7;9 5 8]
X=sqrtm(A)
B=X*X
```

~

```
% chap3_71.m
```

```
clear all;
A=[3 2 3;6 5 7;9 5 8]
X1=expm(A)
Y1=logm(X1)
X2=logm(A)
Y2=expm(X2)
[V, D]=eig(A)
V*diag(exp(diag(D)))/V
```

```
% chap3_72.m

clear all;
A=[3 2 3;6 5 7;9 5 8]
X1=funm(A, @sin)
X2=funm(A, @cos)
X3=funm(A, @exp)
Y=expm(A)
```

## 稀疏矩阵

在MATLAB中，对于矩阵的存储有两种方式：完全存储方式和稀疏存储方式。完全存储方式是将矩阵的全部元素按照矩阵的列存储。以前讲到的矩阵的存储方式都是按这种方式存储的。如果矩阵中的元素只有少数不是零，会浪费大量的存储空间。下面介绍矩阵的稀疏存储。

### 矩阵存储方式

稀疏存储矩阵只是矩阵的存储方式不同，它的运算规则与普通矩阵是一样的。用户可以创建整型、双精度、复数类型和逻辑类型的稀疏矩阵。稀疏矩阵不能自动生成。定义在完全存储方式下的运算只能产生完全存储的矩阵，不论多少个元素为0。在运算过程中，稀疏存储矩阵可以直接参与运算，产生的结果也是稀疏矩阵。

表 3.1 稀疏矩阵常用函数。

函数名。	函数的功能。
sparse <sup>(1)</sup>	将普通矩阵转换为稀疏矩阵。
full <sup>(2)</sup>	将稀疏矩阵转换为普通矩阵。
nnz <sup>(3)</sup>	矩阵中非零元素个数。
nonzeros <sup>(4)</sup>	矩阵中非零元素值。
issparse <sup>(5)</sup>	测试矩阵是否为系数矩阵。
spy <sup>(6)</sup>	图形化显示稀疏矩阵中非零元素。
spalloc <sup>(7)</sup>	为稀疏矩阵分配内存空间。

产生稀疏矩阵

在MATLAB中，通过函数**sparse( )**把普通矩阵转换为稀疏矩阵，该函数的调用格式为：

**S=sparse(A)**，该函数将矩阵A转换为稀疏矩阵S。当矩阵A是稀疏存储方式时，则函数调用相当于**S=A**。

**S=sparse(m, n)**：该函数产生大小为m行n列，所有元素都是0的稀疏矩阵。

在MATLAB中，通过函数**full( )**把稀疏矩阵转换为普通矩阵，该函数的调用格式为：**B=full(A)**，该函数将稀疏矩阵A转换为普通矩阵B。

```
% chap3_73.m
```

```
clear all;
A=rand(15, 10)>0.98
S=sparse(A)
whos
```

```
% chap3_74.m
```

```
clear all;
A=[0 0 0 2;0 0 3 0;0 0 0 0;4 0 0 0]
S1=sparse(A)
S2=sparse([4, 2, 1], [1, 3, 4], [4 3 2], 4, 4)
B=full(S1)|
```

在MATLAB中采用函数**nnz(S)**计算稀疏矩阵S中非零值的个数。

通过函数**spy( )**对稀疏矩阵中非零元素的分布进行图形化显示。

获取稀疏矩阵中非零元素的密度。



在MATLAB中，采用函数**spalloc( )**为稀疏矩阵分配空间。

```
% chap3_75.m

clear all;
A=[0 0 0 2;0 0 3 0;0 0 0 0;4 0 0 0]
S=sparse(A)
n1=nnz(S)
n2=nonzeros(S)
n3=nzmax(S)
spy(S)|
```

% chap3\_76.m

```
clear all;
v=[1 2 3 4 5];
X=diag(v, 1)
S=sparse(X)|      I
nnz(S)
d=nnz(S)/prod(size(S))
```

% chap3\_77.m

```
clear all;
n=5;
S=spalloc(n, n, 3*n);
for j = 1:n
    S(:, j)=[zeros(n-3, 1)' round(rand(3, 1))'|
```

## 特殊稀疏矩阵

单位矩阵只有对角线元素为1，其它元素都为0，是一种具有稀疏特征的矩阵。在MATLAB中，函数eye( )产生一个完全存储方式的单位矩阵。在MATLAB中，还有一个产生稀疏存储方式的单位矩阵的函数speye( )。该函数的调用格式为：

**S=speye(n)** 产生一个n行n列的单位稀疏存储矩阵。

**S=speye(m, n)** 产生一个m行n列的单位稀疏存储矩阵。

% chap3\_78.m

```
clear all;
A=speye(5)
B=speye(4, 6)
C=full(A)|
```

函数spones( )将稀疏矩阵中的非零元素替换为1。



通过函数spconvert( )将普通矩阵转换为稀疏矩阵。

```
% chap3_79.m  
clear all;  
S=sprandsym(8, 0.1)  
R=spones(S)  
  
clear all;  
A=[2 3 4 5;5 6 7 2;1 2 4 0;8 7 9 2]  
S=spconvert(A)
```

## 矩阵的分解

矩阵分解是指根据一定的原理用某种算法将一个矩阵分解成若干个矩阵的乘积。常见的矩阵分解有**Cholesky**分解、**LU**分解、**QR**分解，以及**Schur**分解和**Hessenberg**分解等。下面分别进行介绍。

### Cholesky 分解

对于正定矩阵，可以分解为上三角矩阵和下三角矩阵的乘积，这种分解称为**Cholesky**分解。并不是所有的矩阵都可以进行**Cholesky**分解。能够进行**Cholesky**分解的矩阵必须是正定的，矩阵的所有对角元素必须是正的，同时矩阵的非对角元素不能太大。

在MATLAB中，通过函数**chol( )**进行矩阵的**Cholesky**分解。在采用函数**chol( )**进行**Cholesky**分解时，最好先通过函数**eig( )**得到矩阵的所有特征值，检查特征值是否为正

```
% chap3_82.m  
clear all;I  
A=pascal(4)  
eig(A)  
R=chol(A)  
R'*R
```

## LU 分解

高斯消去法又称为LU分解，将方阵A分解为下三角矩阵的置换矩阵L和上三角矩阵U的乘积。在MATLAB中，通过函数lu( )进行矩阵的LU分解。该函数的调用格式为：

[L1, U1]=lu(A): 该函数将矩阵分解为下三角矩阵的置换矩阵L1和上三角矩阵U1。

[L2, U2, P]=lu(A): 该函数将矩阵分解为下三角矩阵L2和上三角矩阵U2，以及置换矩阵P。

Y=lu(A): 该函数将下三角矩阵和上三角矩阵合并在矩阵Y中，矩阵Y的对角元素为上三角矩阵的对角元素。

```
% chap3_83.m

clear all;
A=[ 2 3 4;8 4 9;5 3 1]
[L1, U1]=lu(A)
[L2, U2, P]=lu(A)
Y1=lu(A)
L1*U1
Y2=L2+U2-eye(size(A))
```

## QR 分解

矩阵的正交分解，又称为QR分解。QR分解将一个m行n列的矩阵A分解为一个正交矩阵Q（m行m列）和一个上三角矩阵R（大小为m行n列）的乘积。

在MATLAB中，通过函数qr( )进行矩阵的QR分解。该函数的调用格式为：[Q, R]=qr(A)，该函数将矩阵A进行QR分解，返回正交矩阵Q和上三角矩阵R。

```
% chap3_84.m

clear all;
A=[ 2 3 4;8 4 9;5 3 1]
[Q1, R1]=qr(A)
B=[1 2 3 4;3 5 6 2;3 6 9 12]
[Q2, R2]=qr(B)
Q2*R2
```

## SVD 分解

奇异值分解在矩阵分析中非常重要，也是常用的矩阵分解之一。在MATLAB中，通过函数`svd( )`进行矩阵的SVD分解（或奇异值分解）。

**s=svd(A):** 该函数对矩阵A进行奇异值分解，返回由奇异值组成的列向量，奇异值按照从大到小的顺序进行排列。

**[U, S, V]=svd(A):** 该函数对矩阵进行奇异值分解，其中U和V为酉矩阵，S为一个对角矩阵，对角线的元素为矩阵的奇异值的降序排列。

```
% chap8_85.m

clear all;
A=[ 2 3 4;8 4 9;5 3 1]
s=svd(A)
[U, S, V]=svd(A)
U*S*V'
norm(A)      I
```

## Schur 分解

对矩阵A的Schur分解公式为： $A=U*S*U'$ ，矩阵A必须是方阵，U为酉矩阵，S为块对角矩阵。在MATLAB中，通过函数`schur( )`进行矩阵的Schur分解。该函数的调用格式为：

**[U, S]=schur(A):** 该函数将矩阵A进行Schur分解，返回酉矩阵U和块对角矩阵S。

**S=schur(A):** 该函数仅返回块对角矩阵S。

```
% chap8_86.m

clear all;
A=pascal(4)
[U, S]=schur(A)
U*S*U'
```

## Hessenberg 分解

对于任意一个n阶方阵可以进行Hessenberg分解，分解公式为： $A=PHP'$ ，其中P是酉矩阵，H的第一子对角线下的元素均为0，即H为Hessenberg矩阵。在MATLAB中，采用函数hess( )进行方阵的Hessenberg分解，该函数的调用格式为：

**H=hess(A)**: 该函数对方阵A进行Hessenberg分解，返回Hessenberg矩阵。

**[P, H]=hess(A)**: 该函数对方阵A进行Hessenberg分解，返回值为P和H，满足 $A=PHP'$ 。

```
% chap8_87.m

clear all;
A=[1 3 5 3 ;2 6 9 1;7 4 1 10;2 5 9 3];
H1=hess(A)
[P2,H2]=hess(A)
B=P2*H2*P2'
C=P2'*P2
```

## 本章小结

本章对数组和矩阵进行了详细的介绍，重点讲解了矩阵的建立、矩阵的特征和分解、以及稀疏矩阵等内容。读者需要多加练习，对这些函数进行更为深入的理解。

# 字符串分析

在使用MATLAB时经常会遇到对字符或字符串的操作。本章将对MATLAB的字符串函数进行详细的介绍。字符串是指 $1 \times n$ 的字符数组。

在MATLAB软件中提供了很多的字符或字符串操作方法和函数，包括字符串的创建、字符串的属性、比较、查找以及字符串的转换和执行等。

字符串是指 $1 \times n$ 的字符数组

## 字符串处理函数

在MATLAB中，单个字符是按照Unicode编码存储的，每个字符占两个字节。MATLAB内部按照字符的编码数值对字符串进行运算。字符和字符串是MATLAB语言的重要组成部分，MATLAB提供了强大的字符串处理功能，下面对常用的字符串处理函数进行详细的介绍。

表 4.1 字符串的常见操作<sup>①</sup>

函数名 <sup>②</sup>	说明 <sup>③</sup>	函数名 <sup>②</sup>	说明 <sup>③</sup>
strcat <sup>④</sup>	字符串连接 <sup>⑤</sup>	strvcat <sup>④</sup>	字符串的垂直连接 <sup>⑤</sup>
strcmp <sup>④</sup>	字符串比较 <sup>⑤</sup>	stncmp <sup>④</sup>	比较前n个字符 <sup>⑤</sup>
findstr <sup>④</sup>	字符串查询 <sup>⑤</sup>	strjust <sup>④</sup>	证明字符数组 <sup>⑤</sup>
strmatch <sup>④</sup>	查找匹配的字符串 <sup>⑤</sup>	strrep <sup>④</sup>	字符串替代 <sup>⑤</sup>
strtok <sup>④</sup>	查找字符串中的记号 <sup>⑤</sup>	blanks <sup>④</sup>	生成空字符串 <sup>⑤</sup>
deblank <sup>④</sup>	删除空格 <sup>⑤</sup>	ischar <sup>④</sup>	字符串检验 <sup>⑤</sup>
iscellstr <sup>④</sup>	字符串的单元检测 <sup>⑤</sup>	isletter <sup>④</sup>	检验是否为字母 <sup>⑤</sup>
isspace <sup>④</sup>	检验是否为空格 <sup>⑤</sup>	strings <sup>④</sup>	strings函数的帮助 <sup>⑤</sup>

## 字符串基本属性

在MATLAB中，对字符串的设定非常的简单，只需要用单引号（'）将需要设定的字符串括起来。

通过函数disp( ): 对字符串进行显示。

通过函数size( ): 取得该字符串的长度。

通过函数double( ): 将字符串以ASCII码显示。

通过函数char( ): 将ASCII码以字符串显示。

```
% chap4_1.m

clc;
clear all;
str='My name is Robin.';
disp(str); %字符串的输出
str_size=size(str);
str(1:7) %字符串的长度
str2=str(end:-1:1) %截取字符串的一部分
str_ascii=double(str) %字符串反序输出
char(str_ascii) %字符串的ASCII码显示
str3='大家好!';
str3_ascii=double(str3) %以字符串显示
char(str3_ascii) %显示汉字
str3_ascii %显示汉字的编码
char(str3_ascii) %显示汉字
```

## 字符串的构造

在MATLAB中，创建字符串非常的简单，将字符串中的字符放到一对单引号之间即可。该单引号必须是在英文状态下输入的。多个字符串可以构成字符矩阵，但是矩阵的每行字符数必须相等。

函数`strcat( )`和函数`strvcat( )`对字符串进行连接。函数`strcat( )`将多个字符串连接成行向量，字符串首尾连接在一起，形成一个新的字符串。函数`strvcat( )`将多个字符串连接成列向量。

```
% chap4_3.m

clc;
clear all;
str1='My name is ';
str2='Robin';
s1=[str1 str2] %连接两个字符串
s2=strcat(str1,str2) %采用函数strcat( )连接字
s3=strvcat(str1,str2) %把多个字符串变为矩阵
```

## 字符串的比较

在MATLAB中，对两个字符串进行比较采用函数strcmp( )和函数strncmp( )，下面分别进行介绍：

函数strcmp( )的调用格式为n=strcmp(str1, str2)，该函数比较字符串str1和字符串str2是否相等，如果相等，函数返回值为1；当不相等时，返回值为0。

函数strncmp( )的调用格式为n=strncmp(str1, str2, k)，该函数比较字符串str1和字符串str2的前k个字符是否相等，如果相等，返回值为1；当不相等时，返回值为0。该函数区分字符的大小写。

函数strcmpi( )的调用格式为n=strcmpi(str1, str2, k)，和函数strncmp( )基本一样。该函数和函数strncmp( )的不同之处是，该函数不区分字符的大小写。

```
% chap4_4.m  
clc;  
clear all;  
str1='windows';  
str2='word';  
n=strcmp(str1,str2)          %判断两个字符串是否相等  
n1=strncmp(str1,str2,1)      %判断前k个字符是否相等  
n2=strncmp(str1,str2,2)
```

## 字符串的查找和替换

字符串的查找和替换时字符串操作的一项重要内容，在MATLAB中提供了函数findstr( )、strfind( )、和strrep( )等函数来实现对字符串的查找和替换操作。

函数strfind( )的调用格式为k=strfind(text, pattern)，在text字符串中查找pattern字符或字符串，当查找成功后返回第一个相同字符的具体位置。

函数strrep( )的调用格式为S=strrep(S1, S2, S3)，该函数会将字符串S1中的子串S2都替换为S3，然后返回到字符串S中。

```
% chap4_7.m

clc;
clear all;
str='My English name is Robin.';
n1=strfind(str,'n')
n2=strfind('n',str)
n3=strfind(str,'name')      %查找字符串
n4=strfind(str,'Name')      %区分大小写
n5=strfind(str,' ')          %查找空格
```

```
% chap4_8.m
```

```
clc;
clear all;
str='My English name is Robin.';
s1=strrep(str,'Robin','Tom')    %字符串替换
s2=strrep(str,'robin','Tom')    %如果没有找到,
str2={'c:\Robin';'c:\Tom'};
str3=strrep(str2,'c:','d:')
```

## 字符串的转换

在MATLAB中，可以使用函数**num2str( )**、**int2str( )**、**str2num( )**和**str2double( )**等实现字符串和数值之间的相互转换。下面分别进行介绍：

**函数t=num2str(X):** 函数将数字X转换为字符串t。如果输入参数X为矩阵，则转换为一个字符串矩阵。该函数也可以指定数字的精度，调用格式为**t=num2str(X, n)**，其中t的精度为n位。

**函数t=int2str(X),** 该函数将整数X转换为字符串。如果X不是整数，先将X取整，然后再转换为字符串。

**函数x=str2num(S),** 该函数将字符型矩阵S转换为一个数字矩阵。

**函数x=str2double(S),** 该函数将字符串转换为双精度的数值。

**函数str=mat2str(mat),** 该函数将数组或矩阵转换为对应的字符串。

```
% chap4_9.m  
I  
clc;  
clear all;  
t1=num2str(randn(3,3)*100)      %将数字转换为字符串  
pi=num2str(pi,6)                 %只显示6位  
t2=int2str(randn(3,3))          %先取整，然后转换  
x1=str2num(t2)                  %将字符串矩阵转换为双精度数  
x2=str2double('123+45i')        %将字符串转换为复数  
str=mat2str(magic(3))           %将矩阵转换为字符串
```

```
>> t1=num2str(randn(3,3)*100)  
  
t1 =  
  
3×37 char 数组  
  
' 53.76671     86.21733    -43.3592'  
' 183.3885     31.87652     34.26245'  
' -225.8847    -130.7688    357.8397'
```

```
>> whos  
Name      Size            Bytes  Class      Attributes  
t1         3x37             222   char
```

```
>> pi=num2str(pi,6)  
  
pi =  
  
' 3.14159'
```

## 字符串的其他操作

除了字符串的构造、查找、替换，以及字符串的转换等，在**MATLAB**中还有一些字符串处理函数，能够进行字符串的一些操作，例如字符的分类、字符串的执行等，还可以进行字符串大小写的转换等。

## 字符的分类

在MATLAB中，字符串中的字符通常可以分为空白字符、字母字符和其他类型的字符。可以通过函数`isspace( )`和函数`isletter( )`对字符串中的字符进行分类。下面分别进行介绍：

**函数`isspace(S)`：**字符串S进行分类，如果为空白字符，返回值为1；否则返回值为0；

**函数`isletter(S)`，**对字符串S进行分类，如果为字母字符，返回值为1；否则返回值为0。

```
% chap4_11.m  
clc;  
clear all;  
str='My name is Robin!';  
isspace(str)           %是否为空格  
isletter(str)          %是否为字母
```

## 字符串的执行

MATLAB表达式可能包含在字符串中，下面详细介绍如何执行这些包含在字符串中的表达式。在MATLAB中提供了三个函数`eval( )`、`evalc( )`和`evalin( )`，可以执行字符串中包含的MATLAB程序。

**函数`eval( )`的调用格式为：`eval('expression')`：**在该函数中`expression`为包含MATLAB表达式的字符串。

```
% chap4_12.m  
clc;  
clear all;  
x=0:pi/50:2*pi;  
str='figure;y=sin(x);plot(x,y);legend("sinx")'  
eval(str);  
  
//这个有啥用（笑死 hh）
```

## 其他操作

在MATLAB中通过函数upper( )可以将字符串转换为大写字母。

函数lower( )将字符串转换为小写字母。

利用函数ischar( )判断是否为字符，如果为字符则返回值为1，否则返回值为0。

在MATLAB的命令窗口，输入命令“help strfun”，可以显示所有的字符串操作函数。

命令窗口输入 help strfun

```
% chap4_16.m  
clc;  
clear all;  
str='My name is Robinson.'  
s1=upper(str)  
s2=lower(str)  
x=[1 2 3];  
y1=ischar(str)  
y2=ischar(x)
```

## 本章小结

本章介绍了MATLAB中一种常用的数据类型：字符串，重点讲解了字符串的构造、属性、比较，以及字符串的查找和替换等操作。此外，还介绍了字符串的分类和执行。这些是本章的重点内容，读者要熟练掌握。

# MATLAB 数据分析

针对数据分析和处理，MATLAB提供了大量的函数，非常方便和灵活。本章将详细的介绍利用MATLAB进行一些基本的数据分析，主要包括多项式及其函数，插值，以及函数的极限。MATLAB能够很好的解决多项式运算问题，这些函数用于多项式的建立、多项式求值、多项式乘法和除法、多项式求导、多项式展开和拟合等等。插值函数，主要包括一维插值、二维插值、样条插值和高维插值等内容。MATLAB提供了非常强大的函数，非常方便和灵活的求得函数的极限。

## 多项式及其函数

MATLAB提供了一些处理多项式的专用函数，用户可以很方便地进行多项式的建立、多项式求值、乘法和除法运算，以及求多项式的导数和微分、多项式的根、多项式的展开和拟合等。

### 多项式的建立

MATLAB语言中，对于多项式，用多项式的系数按照降幂次序存放在向量中。顺序必须是从高到低进行排列。例如，多项式可以用系数向量来表示。多项式就转换为多项式系数向量问题，次多项式用一个维的行向量表示，在多项式中缺少的幂次要用“0”来补齐。

降幂！

```
%chap5_1.m  
  
clc;  
clear all;  
p1=[4 3 2 1];  
y=poly2sym(p1) %由向量创建多项式  
disp(y) | %显示多项式  
  
>> clear  
>> p1=[4 3 2 1];  
>> y=poly2sym(p1);  
>> disp(y)  
4*x^3 + 3*x^2 + 2*x + 1
```

```
%chap5_2.m

clc;
clear all;
r=[2 3 4];
p=poly(r);      %由多项式的根创建多项式
y=poly2sym(p)   %显示多项式
```

## 多项式的求值与求根

### 1. 多项式求值

在MATLAB中，提供了两个函数对多项式进行求值，函数**polyval( )**和**polyvalm( )**。函数**polyval( )**以数组或矩阵中的元素为计算单位，函数**polyvalm( )**以矩阵为计算单位。

### 2. 多项式求根

在MATLAB中，利用**roots( )**函数用来求多项式的根，其调用格式为：**x=roots(p)**，其中参数**p**为多项式系数，输出参数**x**为多项式的根。对于**n**次多项式具有**n**个根，这些根可能是实根，也可能是共轭复根。在MATLAB中，如果已经知道多项式的根，可以利用函数**poly( )**求多项式的系数，其调用格式为：**y=poly(x)**，输入参数**x**为根，输出参数**y**为得到的多项式系数向量。

```
% chap5_3.m

clc;
clear all;
p=[1 -1 -5];      %多项式的系数向量
x=2:5;
y=polyval(p, x)    %多项式在x处的值

%chap5_4.m

clc;
clear all;
x=[1 2 ;3 4];
p=[2 3 4];
y1=polyvalm(p, x) %采用polyvalm( )函数，以矩阵为计算单位
x=[1 2 ;3 4];
p=[2 3 4];
y2=polyval(p, x)  %采用polyval( )函数，以矩阵的元素为计算单位

//结果为什么不一样？ //确实不一样（笑死）
```

```
% chap5_5.m

clc;
clear all;
p=[1 0 0 -1 -6];
x1=roots(p)          %对多项式p求根
x2=[4 5];
y=poly(x2);          %求以x2为根的多项式
y=poly2sym(y)
```

## 多项式的乘法和除法

在MATLAB中，使用函数conv( )对多项式进行乘法运算。其调用格式为c=conv(a, b)，a和b为多项式的系数向量，该函数实现向量a和b的卷积，在代数上相当于多项式a乘以多项式b，其中c为相乘所产生的多项式的系数向量。

!!!该函数实现向量 a 和 b 的卷积，在代数上相当于多项式 a 乘以多项式 b  
C 为相乘所产生的多项式的系数向量!!!

```
%chap5_6.m

clc;
clear all;
p1=[4 2 0 5];           %缺少的幂次用0补齐
p2=[5 8 1];
y1=poly2sym(p1);
y2=poly2sym(p2);
p3=conv(p1, p2);        %多项式相乘
y=poly2sym(p3);
syms x
p1=sym2poly(4*x^3+2*x^2+5)
p2=sym2poly(5*x^2+8*x+1)
p3=conv(p1, p2);        %多项式相乘
y=poly2sym(p3)
```

## 多项式的导数和积分

### 1. 多项式的导数

在MATLAB中，采用函数polyder( )进行多项式的求导，调用方式为：

**y=polyder(p):** 对以向量p为系数的多项式求导；

**y=polyder(a, b):** 对以a和b为系数的多项式的乘积进行求导；

**[q,d]=polyder(b, a):** 返回以b为系数的多项式除以以a为系数的多项式的商的导数，并以q/d格式表示。

## 2. 多项式的积分

在MATLAB中，使用函数**polyint( )**对多项式进行积分运算，其调用方式为：

**polyint(p, k)**: 返回以向量p为系数的多项式的积分，积分的常数项为k；

**polyint(p)**: 返回以向量p为系数的多项式的积分，积分的常数项为默认值0。

```
% chap5_8.m
```

```
clc;
clear all;
p1=[4 3 2];
p2=[2 2 1];
y1=polyder(p1);%对多项式p1求导
y1=poly2sym(y1)
y2=polyder(p1, p2); %对多项式p1和p2的乘积求导
y2=poly2sym(y2)
[q, d]=polyder(p1, p2);%对多项式p1除以p2的商求导
q=poly2sym(q)
d=poly2sym(d)
```

```
% chap5_9.m
```

```
clc;
clear all;
p1=[3 2 2];
y1=polyint(p1, 3); %对多项式p1进行积分，常数项为3
y1=poly2sym(y1)
y2=polyint(p1); %对多项式p1进行积分，常数项为0
y2=poly2sym(y2)    I
```

//注意是多项式的积分！

## 多项式的展开

在MATLAB中，有理多项式用他们的分子多项式和分母多项式进行表示，函数**residue( )**可以将多项式之比用部分分式展开，也可以将一个部分分式用多项式之比进行表示。

**[r, p, k]=residue(b, a):** 求多项式之比b/a的分式展开，函数的返回值r是余数，p是部分分式的极点，k是常数项。如果多项式a没有重根，展开的形式如下：

$$\frac{b(x)}{a(x)} = \frac{r_1}{x - p_1} + \frac{r_2}{x - p_2} + \cdots + \frac{r_n}{x - p_n} + k_s$$

```
% chap5_10.m
```

```
clc;
clear all;
b=[1 -1 -7 -1];%分子多项式
a=poly([1;5;6]);%分母多项式
[r, p, k]=residue(b, a) %进行多项式b/a展开
[b1, a1]=residue(r, p, k);%通过余数、极点和常数项来求多项式b1
b1=poly2sym(b1)
a1=poly2sym(a1)| I
b=[1 -1 -7 -1];%多项式a有三重根
a=poly([1;1;1]);%分母多项式
[r, p, k]=residue(b, a) %进行多项式b/a展开
```

## 多项式的拟合

在MATLAB中，函数**polyfit( )**采用最小二乘法对给定的数据进行多项式拟合，得到该多项式的系数。该函数的调用方式为：**p = polyfit(x, y, n)**，采用n次多项式来拟合数据x和y，得到以p为系数的多项式。该函数使得p(x)与y最小均方误差最小。

```
% chap5_11.m

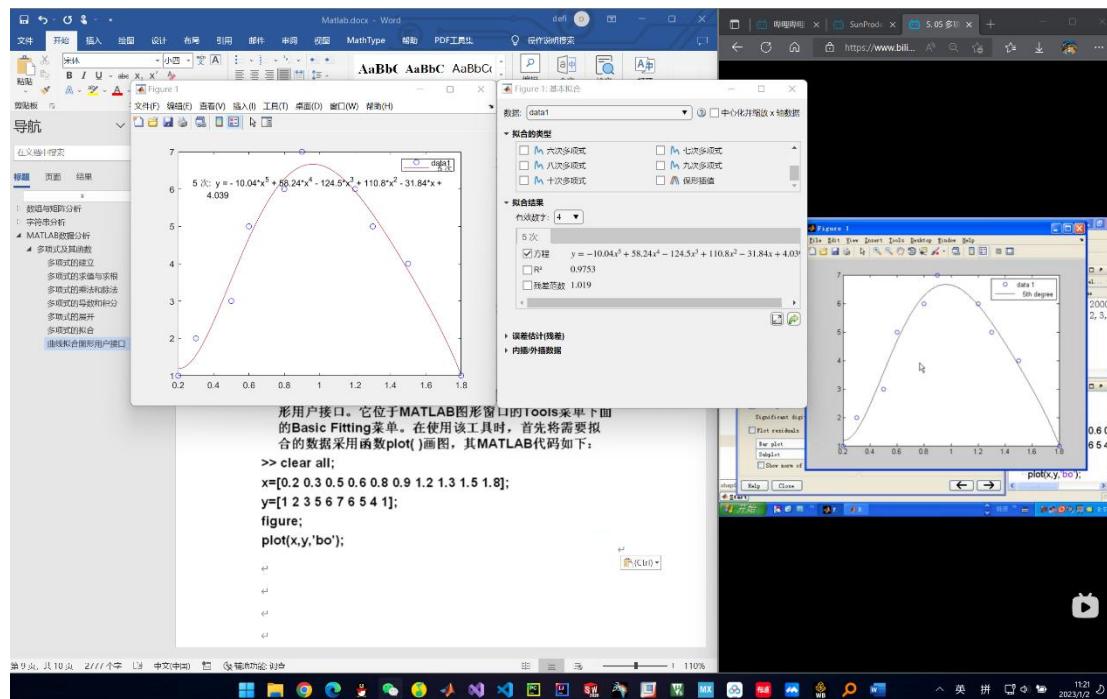
clc;
clear all;
x=[0.2 0.3 0.5 0.6 0.8 0.9 1.2 1.3 1.5 1.8];
y=[1 2 3 5 6 7 6 5 4 1];
p5=polyfit(x, y, 5); %5阶多项式拟合
y5=polyval(p5, x);
p5=vpa(poly2sym(p5), 5)%显示5阶多项式
p9=polyfit(x, y, 9); %9阶多项式拟合
y9=polyval(p9, x);
figure;%画图显示
plot(x, y, 'bo');
hold on;
plot(x, y5, 'r:');
plot(x, y9, 'g--');
legend('原始数据', '5阶多项式拟合', '9阶多项式拟合');

xlabel('x');
ylabel('y');
```

曲线拟合图形用户接口

为了方便用户的使用，在MATLAB中提供了曲线拟合的图形用户接口。它位于MATLAB图形窗口的Tools菜单下面的Basic Fitting菜单。在使用该工具时，首先将需要拟合的数据采用函数plot( )画图，其MATLAB代码如下：

```
>> clear all;
x=[0.2 0.3 0.5 0.6 0.8 0.9 1.2 1.3 1.5 1.8];
y=[1 2 3 5 6 7 6 5 4 1];
figure;
plot(x,y,'bo');
```



## 插值

在数字信号处理和图像处理中，插值是极其常用的方法。

MATLAB提供了大量的插值函数。在MATLAB中，插值函数保存在MATLAB工具箱的polyfun子目录下。下面对一维插值、二维插值、样条插值和高维插值分别进行介绍。

### 一维插值

一维插值是进行数据分析的重要方法，在MATLAB中，一维插值有基于多项式的插值和基于快速傅里叶的插值两种类型。一维插值就是对一维函数进行插值。

#### 1. 一维多项式插值

在MATLAB中，一维多项式插值采用函数interp1( )进行实现。

#### 2. 一维快速傅里叶插值

在MATLAB中，一维快速傅里叶插值通过函数interpft( )来实现，该函数利用傅里叶变换将输入数据变换到频域，然后用更多点的傅里叶逆变换，变換回时域，其结果是对数据进行增采样。

```
% chap5_12.m  
clc;  
clear all;  
x=0:0.2:2;  
y=(x.^2-3*x+5).*exp(-3*x).*sin(x);  
xi=0:0.03:2; %要插值的数据  
yi_nearest=interp1(x,y,xi,'nearest'); %临近点插值  
yi_linear=interp1(x,y,xi); %默认为线性插值  
yi_spline=interp1(x,y,xi,'spline'); %三次样条插值  
yi_pchip=interp1(x,y,xi,'pchip'); %分段三次Hermite  
yi_v5cubic=interp1(x,y,xi,'v5cubic'); %MATLAB5中三次多  
figure; %画图显示  
hold on;  
subplot(231);  
plot(x,y,'ro');
```

```

% chap5_13.m
subplot(231);
plot(x,y,'ro',xi,yi_nearest,'b-');
title('临近点插值');
subplot(232);
plot(x,y,'ro',xi,yi_linear,'b-');
title('线性插值');
subplot(233);
plot(x,y,'ro',xi,yi_spline,'b-');
title('三次样条插值');
subplot(234);
plot(x,y,'ro',xi,yi_spine,'b-');
title('分段三次Hermite插值');
subplot(235);
plot(x,y,'ro',xi,yi_pchip,'b-');
title('MATLAB5中三次多项式插值');

% chap5_13.m

clc;
clear all;
x=0:1.1:8;
y=sin(x);
n=2*length(x); %增采样1倍
yi=interpft(y,n);%采用一维快速傅立叶插值
xi=0:0.55:8.3;%要插值的数据
figure;%画图显示
hold on;           I
plot(x,y,'ro',xi,yi,'b-');
legend('原始数据','插值后结果');

```

## 二维插值

二维插值主要用于图像处理和数据的可视化，其基本思想与一维插值相同，对函数进行插值。

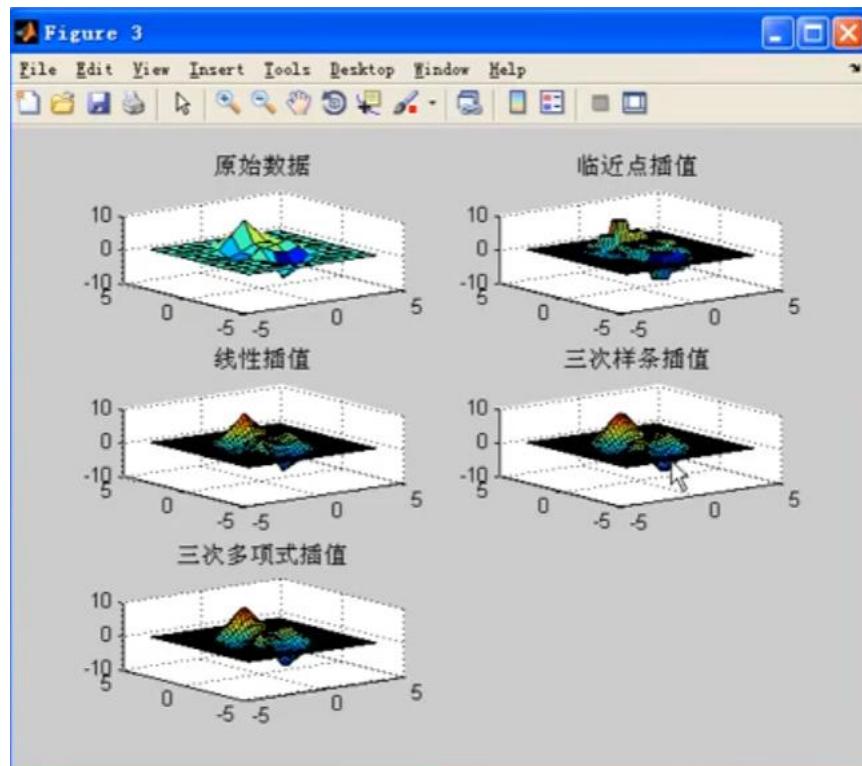
**zi=interp2(x, y, z, xi, yi):** 通过初始数据x、y和z产生插值函数 $y = f(x, y)$ ，返回值zi是 $(xi, yi)$ 在函数f(x, y)上的值。

**zi=interp2(x, y, z, xi, yi, method):** 其中method为可采用的插值方法。二维插值采用的方法只有4种，分别是'nearest'、'linear'、'spline'和'cubic'，其中线性插值为默认的插值方法。

```
% chap5_14.m

clc;
clear all;
[x, y]=meshgrid(-4:0.8:4); %原始数据
z=peaks(x, y);
[xi, yi]=meshgrid(-4:0.2:4); %插值数据
zi_nearest=interp2(x, y, z, xi, yi, 'nearest'); %临近点插值
zi_linear=interp2(x, y, z, xi, yi); %系统默认为线性插值
zi_spline=interp2(x, y, z, xi, yi, 'spline'); %三次样条插值
zi_cubic=interp2(x, y, z, xi, yi, 'cubic'); %三次多项式插值
figure; %数据显示
hold on;
subplot(321);
surf(x, y, z);
title('原始数据');

subplot(322);
surf(xi, yi, zi_nearest);
title('临近点插值');
subplot(323);
surf(xi, yi, zi_linear);
title('线性插值');
subplot(324);
surf(xi, yi, zi_spline);
title('三次样条插值');
```



## 样条插值

在MATLAB中，三次样条插值可以采用函数spline( )，该函数的调用格式为：

**yi=spline(x, y, xi):** 通过初始数据产生插值函数，然后对数据xi进行插值，返回值yi=f(xi)。采用这种调用方式时，其相当于yi=interp1(x, y, xi, 'spline')。

**pp=spline(x, y):** 该函数通过对初始数据x和y产生插值函数，并进行返回。然后利用函数ppval( )对数据xi进行插值计算，其调用方式为yi=ppval(pp, xi)，其中pp为插值函数。

```
% chap5_15.m

clc;
clear all;
x=0:10;%原始数据
y=sin(x);%
xi=0:.25:10;%插值数据
yi=spline(x, y, xi);%三次样条插值
pp=spline(x, y);%产生插值函数
y1=ppval(pp, xi); %结果相同y1=yi
y2=interp1(x, y, xi, 'spline'); %结果相同y2=yi
figure;%画图显示
plot(x, y, 'o', xi, yi);
legend('原始数据', '三次样条插值');
```

## 高维插值

在MATLAB中，采用函数interp3( )进行三维插值，该函数的调用格式为：

**vi=interp3(x, y, z, v, xi, yi, zi):** 该函数通过初始数据产生插值函数，然后对数据进行插值，返回值为。

**vi=interp3(x, y, z, v, xi, yi, zi, method):** 该函数中method为可以采用的插值方法，共有4种，分别是'nearest'、'linear'、'spline'和'cubic'，其中线性插值为默认的插值方法。

此外，在MATLAB中进行高维插值的函数还有interpn( )，可以进行n维插值，

```
% chap5_16.m

clc;
clear all;
[x, y, z, v]=flow(10);
[xi, yi, zi]=meshgrid(.1:.25:10, -3:.25:3, -3:.25:3);
vi = interp3(x, y, z, v, xi, yi, zi); % vi^
slice(xi, yi, zi, vi, [6 9.5], 2, [-2 .2]);
shading flat
```

## 函数的极限

极限理论是微积分学的理论基础。在MATLAB中，采用函数**limit( )**计算数列或函数的极限，可以非常方便的进行极限运算。下面介绍如何利用函数**limit( )**求极限。

### 极限的概念

数列的极限定义为：设  $\{y_n\}$  是数列， $A$ 是常数，若对于任意给定的正数  $\epsilon$ （无论它多么小），总存在正整数  $N$ ，使得当  $n > N$  时，都有

$$|y_n - A| < \epsilon$$

则称该数列以  $A$  为极限。有极限的数列称为收敛数列。

### 求极限的函数

函数的极限包括左极限和右极限。左极限和右极限统称为单侧极限。极限存在且等于  $A$  的充分必要条件是左极限与右极限都存在且相等。

**y=limit(f)**: 当  $x$  趋近于 0 时，该函数对函数  $f$  求极限。

**y=limit(f, x, a)** 或 **y=limit(f, a)**: 当  $x$  趋近于常数  $a$  时，对函数  $f$  求极限。

**y=limit(f, x, a, 'left')**: 当  $x$  从左侧趋近于常数  $a$  时，对函数  $f$  求极限，返回值为求得的左极限。

**y=limit(f, x, a, 'right')**: 当  $x$  从右侧趋近于常数  $a$  时，对函数  $f$  求极限，返回值为求得的右极限。

```
% chap5_20.m

clc;
clear all;
syms x;
f=(3*x^2-1)/(3*x^2-2*x+3);
y=limit(f, x, 1)
```

```
% chap5_21.m

clc;
clear all;
syms x;
f=x/(x-1)-2/(x^2-1);
y=limit(f, x, 1)|    I
```

```
% chap5_23.m

clc;
clear all;
syms x;
f=(1+2/x)^(2*x);    I
y=limit(f, x, inf)
```

```
% chap5_24.m

clc;
clear all;
syms x;
f=((x^x)^x);
y=limit(f, x, 0, 'right')
```

## 本章小结

本章介绍对一些基本的数据分析进行了详细的介绍，主要包括多项式、插值和函数的极限。多项式的操作方法，主要包括多项式的建立、多项式求值、多项式乘法和除法、多项式求导、多项式展开和拟合等等。插值函数，主要包括一维插值、二维插值、样条插值和高维插值等内容。利用**MATLAB**进行数据分析非常的灵活，需要读者熟练掌握。

# 积分和微分运算

积分和微分是微积分的重要内容，本章将对积分和微分的内容进行详细的介绍。积分包括单变量数值积分、双重积分和三重积分等。然后介绍了常微分方程的符号解和数值解。最后介绍了如何求函数的最小值和零点。

## 数值积分

**MATLAB**提供了一些进行定积分的专用函数，用户可以很方便的进行单变量数值积分、双重积分和三重积分等。下面将首先介绍定积分的概念，然后对这些函数进行详细的介绍。

### 定积分概念

函数 $f(x)$ 在区间 $[a, b]$ 上的定积分，记作：

$$\int_a^b f(x) dx = \lim_{\Delta x \rightarrow 0} \sum_{i=1}^n f(\varepsilon_i) \Delta x_i$$

其中 $f(x)$ 称为被积函数， $f(x)dx$ 称为被积表达式， $x$ 称为积分变量， $a$ 称为积分下限， $b$ 称为积分上限，区间 $[a, b]$ 称为积分区间。

### 利用梯形求面积

求解定积分的数值方法有很多，例如简单的梯形法、**Simpson**法和**Romberg**法等等都是数值分析课程中经常介绍的方法。它们的基本思想是将整个积分空间分割成若干个子空间。这样积分问题，就转换为求和问题。

求取定积分的最直观的方法是用梯形法，用直线将这些点连接起来，定积分近似为用这些折线和x轴间围成的面积。

在**MATLAB**中，采用函数**trapz( )**可以直接采用梯形法求解定积分的近似值。

```

% chap6_1.m

clear all;
x=[0:pi/100:pi]';
y=sin(x);
s=sum((2*y(1:end-1,:)+diff(y)).*diff(x))/2

% chap6_1.m

clear all;
X1=[1 2 3 4 5 6 7]
z1=trapz(X1)
X2=[1 2 3;3 6 8;2 5 9]
z2=trapz(X2)
z3=trapz(X2, 2)
x=[1 2 3]
z4=trapz(x, X2)|    I

//我不理解这个向量积分？？？

```

## 利用矩形求面积

**函数cumsum( )**采用矩形法求解定积分的近似值。该函数的调用格式为：

**y=cumsum(X):** 该函数采用矩形法计算定积分的近似值。

**y=cumsum(X, DIM):** 在第DIM维来计算定积分的近似值。

```

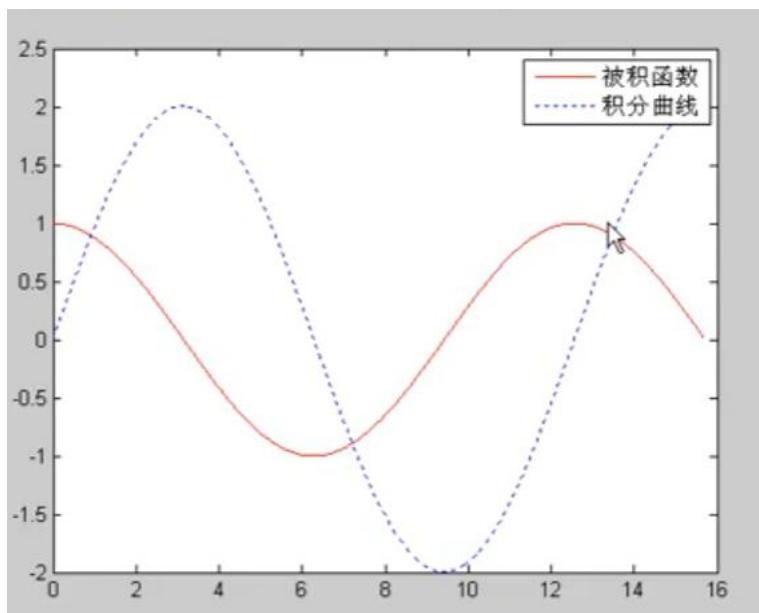
% chap6_4.m

clear all;
X1=[1 2 3 4 5 6 7]
z1=cumsum(X1)
X2=[1 2 3;3 6 8;2 5 9]
z2=cumsum(X2)
z3=cumsum(X2, 1)
z4=cumsum(X2, 2)
z5=cumsum(X2, 3)

% chap6_5.m

clear all;
x=0:0.01:5*pi;
y=cos(x./2)';
z=cumsum(y)*0.01;
z1=z(end)
z2=trapz(x, y)
figure;
plot(x, y, 'r-');
hold on;
plot(x, z, 'b:');
legend('被积函数', '积分曲线');

```



## 单变量数值积分求解

在MATLAB中，采用函数quad( )来求解定积分的数值解，该函数采用自适应变步长的方法。该函数的调用格式为：

**y=quad(Fun, a, b):** 求函数Fun的定积分，定积分的下限为a，定积分的上限为b。

**y=quad(Fun, a, b, ε):** 该函数的输入参数ε用来指定误差限，默认值为 $10^{-6}$ 。

此外，还有函数quadl( )，其调用格式和函数quad( )完全一致，但是求解的精度和速度高于函数quad( )。

```
% chap6_6.m
clear all;
syms x;
f=inline('1./(sin(x)+exp(-x.^2))');
y=quad(f, 0, 1.3)
y1=quad(@myfun1, 0, 1.3)
y2=quad(@myfun1, 0, 1.3, 1.e-10)
```

```
function y=myfun1(x)
y=1./(sin(x)+exp(-x.^2));
```

```
% chap6_8.m
clear all;
syms x;
f=inline('1./(sin(x)+exp(-x.^2))','x');
y=quadl(f, 0, 1.3)
y1=quadl(f, 0, 1.3, 1.e-20)
```

## 双重积分求解

在MATLAB中，可以通过函数**dblquad( )**计算双重积分的数值解，双重积分的格式为：

$$\int_{y_m}^{y_M} \int_{x_m}^{x_M} f(x, y) dx dy$$

该函数的调用格式为：

**q=dblquad(Fun, xm, xM, ym, yM)**: 该函数计算矩形区域的双重积分。

**q=dblquad(Fun, xm, xM, ym, yM, ε)**: 该函数限定双重积分的精度为。

```
% chap6_10.m  
clear all;  
syms x;  
f=inline('x*cos(y)+y*sin(x)', 'x', 'y');  
y=dblquad(f, pi, 2*pi, 0, pi)  
y1=dblquad(@integrand, pi, 2*pi, 0, pi)  
y2=dblquad(@(x, y) x*cos(y)+y*sin(x), pi, 2*pi, 0, pi)  
|
```

## 三重积分求解

在MATLAB中，可以通过函数**triplequad( )**计算三重积分的数值解，三重积分的格式为：

$$\int_{z_m}^{z_M} \int_{y_m}^{y_M} \int_{x_m}^{x_M} f(x, y, z) dx dy dz$$

**q=dblquad(Fun, xm, xM, ym, yM, zm, zM)**: 该函数计算三重积分。

**q=dblquad(Fun, xm, xM, ym, yM, zm, zM, tol)**: 该函数的输入参数**tol**是积分的精度控制量，默认为 $10^{-6}$ ，为了提高计算的精度，可以选择更小的值。

```
% chap6_11.m

clear all;
f=inline('z*cos(x)+y*sin(x)', 'x', 'y', 'z');
q=triplequad(f, 0, pi, 0, 1, -1, 1);
q1=triplequad(@(x, y, z) (y*sin(x)+z*cos(x)), 0, pi, 0, 1,
q2=triplequad(@(x, y, z) (y*sin(x)+z*cos(x)), 0, pi, 0, 1,
q3=triplequad(@(x, y, z) (y*sin(x)+z*cos(x)), 0, pi, 0, 1,
```

## 常微分方程

**常微分方程 (Ordinary Differential Equation, ODE)** 求解是高等数学的基础内容，在实际中有广泛的应用。  
MATLAB提供了很多函数用于求解常微分方程，包括常微分方程的符号解和常微分方程的数值解。

### 常微分方程符号解

在MATLAB中，采用函数`dsolve( )`求解常微分方程的符号解。  
。Dy表示一阶微分项，D2y表示二阶微分项。该函数的调用格式为：

`dsolve('equation')`: 该函数求微分方程的通解。  
`dsolve('equation', 'condition')`: 该函数求微分方程的特解。  
`dsolve('equation1', 'equation2', ..., 'condation1', 'condation2')`: 该函数求解微分方程组。  
`dsolve(..., 'v')`: 该函数求解微分方程，并表示为v的函数。

```
clear all;
f1=dsolve('Dy-y=sin(x)')
f2=dsolve('Dy-y=sin(x)', 'x')

f1 =
C2*exp(t) - sin(x)

f2 =
C4*exp(x) - sin(x)/2 - cos(x)/2

clear all;
dsolve('Dy=a*y', 'y(0)=b')
```

```

clear all;
z=dsolve('Dx=y','Dy=-x')
z.x
z.y

```

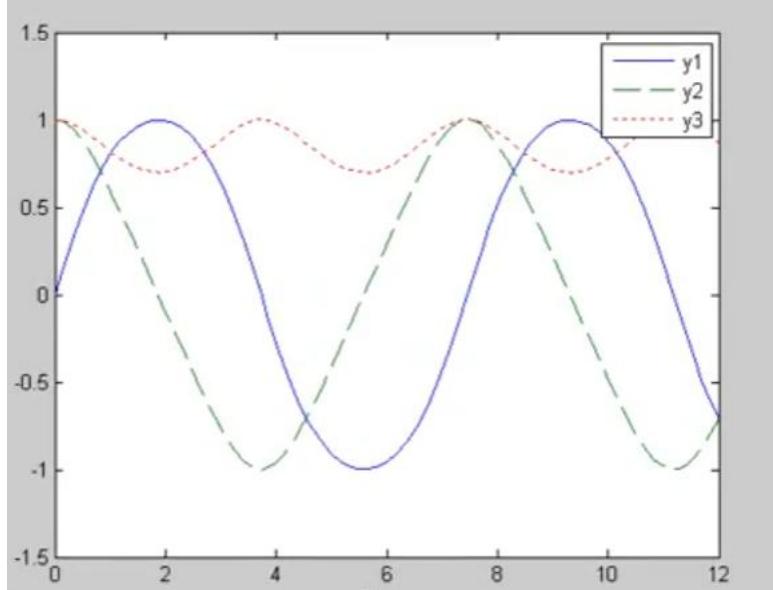
## 常微分方程的数值解

对于大多数的微分方程，很难求出一般解，可以求满足规定精度的近似解，即数值解。在MATLAB 2010a中，提供了常微分方程的7种解法，它们是函数ode45( )、ode23( )、ode113( )、ode15s( )、ode23s( )、ode23t( )、ode23tb( )。

```

clear all;
close all;
options=odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5
[t, y]=ode45(@rigid,[0 12],[0 1 1],options);
figure;
plot(t, y(:,1),'-',t, y(:,2),'--',t, y(:,3),':');
xlabel('Time');
legend('y1','y2','y3');

```



## 函数的极小值和零点

求函数的最小值和零点是工程上常见的问题，在MATLAB中提供了一些函数能够用于求解函数的最小值和零点。对于求解最大值的问题，可以在原函数上乘以-1，然后求最小值。下面介绍如何求函数的极小值和零点。

## 一元函数的最小值

在MATLAB中，采用函数fminbnd( )求解一元函数在给定的区间内的最小值，该函数的调用格式为：

**x=fminbnd(fun, x1, x2):** 该函数在区间[x1, x2]内寻找函数fun的最小值，fun为匿名函数或函数句柄，x为最小值的自变量取值。

**x=fminbnd(fun, x1, x2, options):** 该函数采用options进行优化器的设置。options可以采用函数optimset( )来设置。

**[x, fval]=fminbnd(...):** 该函数返回输出变量最小值fval。

```
clear all;
close all;
[x, fval]=fminbnd(@cos, 3, 4)
y=cos(x)

clear all;
close all;
f=@(x)x.^3-2*x-5
[x, fval]=fminbnd(f, 0, 2, optimset('Display','iter'))
y=f(x)|
```

## 多元函数的最小值

在MATLAB中，采用函数fminsearch( )求多元函数的最小值。该函数需要指定一个起始点x0，然后得到在点x0附近的局部最小值。该函数的调用格式为：

**x=fminsearch(fun, x0):** 该函数获取函数fun在x0点附近的最小值点，x为最小值的自变量的取值。

**x=fminsearch(fun, x0, options):** 该函数通过options对优化器的参数进行设置。

**[x, fval]=fminsearch(...):** 该函数返回最小值fval。

```
clear all;
close all;
f=@(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2; I
[x, fval]=fminsearch(f, [-1.2 1])
y=f(x)
```

## 一元函数的零点

在MATLAB中，利用函数fzero( )求一元函数的零点。用户可以指定一个区间或开始点。如果用户知道函数零点所在的区间，则可以使用一个包含两个元素的向量来指定区间。如果不知道取零点的区间，也可以指定一个开始点，函数在开始点附近寻找使函数值改变符号的区间，如果没有找到这样的区间，则返回值为NaN。

**x=fzero(fun, x0):** 该函数求一元函数fun的零点，返回值为零点的自变量的值。

**x=fzero(fun, x0, options):** 该函数通过options对优化器的参数进行设置。

**[x, fval]=fzero(...):** 该函数输出自变量为x时的函数值。

```
clear all;
close all;
f=@(x)sin(x)*cos(x);
figure;
fplot(f, [-5, 5]);
xlabel('x');
ylabel('f(x)');
grid on;
x1=fzero(f, [-5, -4.3])
x2=fzero(f, -4.2)
x3=fzero(f, -1)
[x4, fval]=fzero(f, [-3, -1])
```

## 本章小结

本章详细的介绍了利用MATLAB进行积分和微分运算，积分和微分运算是高等数学的核心内容。积分包括单变量积分、双重积分和三重积分。利用MATLAB可以非常方便的进行积分和微分运算。最后介绍了如何求函数的最小值和零点。

# 概率和数理统计

概率和数理统计是大学数学的重要内容，无论是在科学的研究还是在工程实际中都有着非常广泛的应用。在MATLAB中，提供了专门的统计工具箱Staticstics，该工具箱有几百个专用于求解概率和数理统计问题的函数。本章将详细的介绍随机数的产生，随机变量的概率密度函数和数字特征，以及假设检验、方差分析和统计绘图等。

## 随机数的产生

随机数的产生是概率统计的基础，概率论和数理统计就是对各种样本数据进行分析。在MATLAB中，各种样本数据可以用一些经典的随机分布数来表示。下面对常用的二项分布、均匀分布、指数分布、正态分布等随机数据进行详细的介绍。

### 二项分布随机数据的产生

在MATLAB中，使用函数binornd( )产生二项分布的随机数据。该函数的调用方式如下：

R=binornd(N, P): 在该函数中N和P为二项分布的两个参数，返回服从参数为N和P的二项分布随机数。

R=binornd(N, P, M): 在该函数中参数M指定随机数的个数，与返回结果R同维数。

```
% chap7_1.m  
clear all;  
r=binornd(10, 0.5)  
R=binornd(10, 0.5, 3, 4)    %产生一个3*4的矩阵
```

## 泊松分布

在MATLAB中，使用函数`poissrnd( )`产生泊松分布的随机数据。该函数的调用方式如下：

**R=poissrnd(LAMBDA):** 在该函数中`LAMBDA`为泊松分布的参数，返回服从参数为`LAMBDA`的泊松分布随机数，其中`R`和`LAMBDA`维数相同。

**R=poissrnd(LAMBDA, M, N):** 在该函数中`LAMBDA`为泊松分布的参数，返回服从参数为`LAMBDA`的泊松分布随机数矩阵，矩阵的大小为。

```
% chap7_2.m  
clear all;  
r=poissrnd(8)          %泊松分布  
R=poissrnd(8, 4, 4)    %产生一个4*4的矩阵  
|
```

## 指数分布随机数据的产生

在MATLAB中，使用函数`exprnd( )`生成具有指数分布的随机数据。其调用格式如下：

**R=exprnd(MU):** 该函数返回一个以`MU`为参数的指数分布的随机数，其中`R`和`MU`同维数。

**R=exprnd(MU, M, N):** 该函数返回一个以`MU`为参数的指数分布的随机数矩阵，该矩阵的大小为。

```
% chap7_3.m  
clear all;  
r=exprnd(10)           %指数分布  
R=exprnd(8, 4, 4)      %产生一个4*4的矩阵  
|
```

## 均匀分布随机数据的产生

### 1. 连续型均匀分布

在MATLAB中，使用函数`unifrnd( )`生成连续型均匀分布的随机数据。该函数的调用方式如下：

**R=unifrnd(A, B):** 返回区间`[A, B]`上的连续型均匀分布；

**R=unifrnd(A, B, M, N):** 返回区间`[A, B]`上的连续型均匀分布矩阵；

```
% chap7_4.m  
  
clear all;  
r=unifrnd(1, 3) %均匀分布  
R1=unifrnd(1, 3, 4, 4) %产生一个4*4的矩阵  
R2=unifrnd(1, 3, [4 4])
```

## 2. 离散型均匀分布

在MATLAB中，使用函数`unidrnd()`生成离散型均匀分布的随机数据。该函数的调用方式如下：

**R=unidrnd(N):** 返回一个离散型均匀分布，**R**和**N**同维数；

**R=unidrnd(N, MM, NN):** 返回一个离散型均匀分布矩阵，矩阵的大小为。

```
% chap7_5.m  
  
clear all;  
r=unidrnd(10) %离散型均匀分布  
R1=unidrnd(8, 4, 4) %产生一个4*4的矩阵  
R2=unidrnd(8, [4 4])
```

## 正态分布随机数据的产生

在MATLAB中，采用函数`normrnd()`生成参数为**MU**和**SIGMA**的正态分布随机数据。该函数的调用方式如下：

**R=normrnd(MU, SIGMA):** 该函数返回均值为**MU**，标准差为**SIGMA**的正态分布随机数据，其中**R**可以为向量，也可以为矩阵。

**R=normrnd(MU, SIGMA, M):** 该函数中参数**M**指定随机数的个数，与返回值**R**维数相同。

**R=normrnd(MU, SIGMA, M, N):** 该函数返回一个的矩阵。

```
% chap7_6.m  
  
clear all;  
r=normrnd(0, 1) %  
R1=normrnd(0, 1, [3 5]) %产生一个3*5的矩阵  
R2=normrnd(2, 4, [3 5])
```

## 其他常见分布随机数据的产生

对于其他常见分布的随机数据的产生，其函数调用方式与上面相同。表7.1列出了常用的随机数据产生函数。对于这些函数的详细使用情况用户可以通过MATLAB的帮助系统进行查看，或通过在命令行输入**help**和函数名查看某函数的功能和使用方法。

## 概率密度函数

在MATLAB中，计算某概率分布在指定点的概率密度的函数，都以代表特定概率分布的字母开头，以**pdf**（**Probability Density Function**）结尾。下面介绍常用的离散分布和连续分布的概率密度函数，以及三大抽样分布的概率密度函数。

### 常见离散分布的概率密度函数

#### 1. 二项分布

函数**binopdf( )**产生二项分布密度函数，调用格式为：

**Y=binopdf(X, N, P)**，产生参数为N和P的二项分布。

```
% chap7_7.m
```

```
clear all;
x=1:30;
y=binopdf(x, 300, 0.05);
figure;
plot(x, y, 'r*');
title('二项分布 (n=300, p=0.05)');
```

#### 2. 泊松（Poisson）分布

函数**poisspdf( )**产生泊松分布的密度函数，调用格式为：

**Y=poisspdf(X, LAMBDA)**，产生参数为LAMBDA的泊松分布。

```
% chap7_8.m
```

```
clear all;
x=1:50;
y=poisspdf(x, 25); %泊松分布
figure;
plot(x, y, 'r+');
title('泊松分布');
```

### 3. 几何分布

函数`geopdf( )`产生几何分布的密度函数，该函数的调用格式为：`Y=geopdf(X, P)`。

```
% chap7_9.m

clear all;
x=1:20;
y=geopdf(x, 0.5);           %几何分布
figure;
plot(x, y, 'rx');
title('几何分布');
```

### 4. 均匀分布（离散）

对于离散型均匀分布，采用函数`unidpdf( )`函数来产生概率密度函数，其调用格式为`Y=unidpdf(X, n)`，离散型均匀分布的参数为`n`，输出参数`Y`和输入参数`X`有相同的维数

```
% chap7_10.m

clear all;
n=20;
x=1:n;
y=unidpdf(x, n);           %均匀分布（离散）
figure;
plot(x, y, 'ro');
title('均匀分布（离散）');
```

常见连续分布的概率密度函数

#### 1. 均匀分布（连续）

在MATLAB中，采用函数`unifpdf( )`计算连续型均匀分布的密度函数，其调用格式为`Y=unifpdf(X, A, B)`，该函数产生区间`[A, B]`上的均匀分布。

```
% chap7_11.m

clear all;
x=-5:0.1:10;
y=unifpdf(x, 0, 5);         %均匀分布（连续）
figure;
plot(x, y, 'r:');
title('均匀分布（连续）');
```

## 2. 指数分布

在MATLAB中，采用函数`exppdf( )`产生指数分布概率密度函数，其调用格式为`Y=exppdf(X, MU)`，该函数产生参数为`MU`的指数分布，参数`MU`的默认值为1。输出参数`Y`和输入参数`X`可以为向量，也可以为矩阵，有相同的维数。

```
% chap7_11.m
```

```
clear all;
x=0:0.1:20;
y=exppdf(x, 5);      %指数分布
figure;
plot(x, y, 'r:');
title('指数分布');
```

## 3. 正态分布

在MATLAB中，采用函数`normpdf( )`产生正态分布的概率密度函数，其调用格式为`Y=normpdf(X, MU, SIGMA)`，该函数产生参数为`MU`和`SIGMA`的正态分布。参数`MU`和`SIGMA`的默认值分别为0和1，即标准正态分布。

```
% chap7_13.m
```

```
clear all;
x=-8:0.1:8;
y1=normpdf(x, 0, 1);      %标准正态分布
y2=normpdf(x, 2, 2);      %非标准正态分布
figure;
plot(x, y1, x, y2, ':');
legend('标准正态分布', '非标准正态分布');
x1=-4:0.1:8;
y3=normpdf(x1, 2, 1);      %SIGMA=1
y4=normpdf(x1, 2, 2);      %SIGMA=2
y5=normpdf(x1, 2, 3);      %SIGMA=3
figure;
plot(x1, y3, 'r-', x1, y4, 'b:', x1, y5, 'k--');
legend('SIGMA=1', 'SIGMA=2', 'SIGMA=3');
```

```
y6=normpdf(x1, 0, 1.5);      %MU=0
y7=normpdf(x1, 2, 1.5);      %MU=2
y8=normpdf(x1, 4, 1.5);      %MU=4
figure;
plot(x1, y6, 'r-', x1, y7, 'b:', x1, y8, 'k--');
legend('MU=0', 'MU=2', 'MU=4');
```

## 三大抽样分布的概率密度函数

### 1. 卡方分布

在MATLAB中，采用函数`chi2pdf( )`产生卡方分布，该函数的调用格式为`Y=chi2pdf(X, V)`，该函数产生自由度为V的卡方分布，输出参数Y和输入参数X有相同的维数。

```
% chap7_14.m

clear all;
x=0:0.1:15;
y1=chi2pdf(x, 4);      %卡方分布n=4
y2=chi2pdf(x, 8);      %卡方分布n=8
figure;
hold on;
plot(x, y1);           I
plot(x, y2, 'r');
legend('n=4', 'n=8');
title('卡方分布');
```

### 2. t分布

在MATLAB中，利用函数`tpdf( )`实现t分布的概率密度函数，其调用格式为`Y=tpdf(X, V)`，产生自由度为V的t分布。

```
% chap7_15.m

clear all;
x=-7:0.1:7;
y1=tpdf(x, 3);          %t分布 (n=3)
y2=tpdf(x, 30);         %t分布 (n=30)
figure;
plot(x, y1, 'r:', x, y2, 'b-');
legend('n=3', 'n=30');
title('t分布');
```

### 3. F分布

在MATLAB中，采用函数`fpdf( )`产生F分布的概率密度函数，其调用格式为：`Y=fpdf(X, V1, V2)`，其中V1为第一自由度，V2为第二自由度。

```
% chap7_16.m

clear all;
x=0.1:0.1:8;
y=fpdf(x, 5, 10);       %F分布
figure;
plot(x, y, 'r:');
title('F分布 (m=5, n=10)');
```

## 随机变量的数字特征

下面对随机变量的数字特征进行介绍，包括平均值和中位数，数据的排序，以及期望和方差。其中，期望和方差是随机变量的两个最重要的数字特征。

### 平均值和中位数

算术平均值

在MATLAB中，采用函数`mean( )`计算一组样本数据的算术平均值，其调用格式为：

**y=mean(X):** 该函数返回X的平均值，当X为向量时，返回X中各元素的算术平均值；当X为矩阵时，返回X中各列元素的算术平均值构成的向量。

**y=mean(X, DIM):** 该函数返回在给定的维数DIM上的算术平均值。

```
% chap7_17.m  
  
clear all;  
A=[1 2 3 4;2 0 1 4]  
m1=mean(A)           %对列元素求算术平均值  
m2=mean(A, 2)         %对行元素求算术平均值
```

忽略 NaN 计算算术平均值

在MATLAB中，函数`nanmean( )`用于求一组数据中除了NaN以外的数据的算术平均值，其调用格式为：

**Y=nanmean(X):** 该函数返回X中除NaN以外的算术平均值，当X为向量时，返回X中各元素的算术平均值；当X为矩阵时，返回X中各列元素的算术平均值构成的向量。

**Y=nanmean(X, DIM):** 该函数返回在给定的维数DIM上的除NaN以外的数据的算术平均值。

```
% chap7_18.m  
  
clear all;  
A=[1 2 nan 4;2 nan 1 nan]  
m1=mean(A)  
m2=nanmean(A)           %对列元素求算术平均值  
m3=nanmean(A, 2)         %对行元素求算术平均值
```

几何平均数

在MATLAB中，采用函数`geomean( )`来计算样本数据的几何平均数。

```
% chap7_19.m  
clear all;  
A1=[1 2 3 4]  
m1=geomean(A1)      %向量的几何平均数  
A2=[1 2 3;2 3 4]  
m2=geomean(A2)      %列元素的几何平均数  
m3=geomean(A2, 2)    %行元素的几何平均数
```

调和平均数

在MATLAB中，采用函数`harmmean( )`计算样本数据的调和平均数。

```
% chap7_20.m  
clear all;  
A1=[1 2 3 4]  
m1=harmmean(A1)      %向量的调和平均数  
A2=[1 2 3;2 3 4]  
m2=harmmean(A2)      %列元素的调和平均数  
m3=harmmean(A2, 2)    %行元素的调和平均数
```

中位数

中位数是样本数据中大小位于中间的那个数。在MATLAB中，使用函数`median( )`计算样本数据的中位数。该函数的调用格式为：

**Y=median(X):** 该函数返回X中数据的中位数，当X为向量时，返回X中各元素的中位数；当X为矩阵时，返回X中各列元素的中位数组成的向量。

**Y=median(X, DIM):** 该函数返回在给定的维数DIM上的中位数。

```
% chap7_21.m  
clear all;  
A1=[1 2 3 4]  
m1=median(A1)      %向量的中位数  
A2=[1 2 3;2 3 4;4 1 8]  
m2=median(A2)      %列元素的中位数  
m3=median(A2, 2)    %行元素的中位数
```

忽略 NaN 的中位数

忽略NaN的中位数，是指样本数据中除了NaN以外的数据中大小位于中间的数。在MATLAB中，利用函数nanmedian( )计算样本数据中除NaN以外的数据的中位数。该函数的调用格式为：

**Y=nanmedian(X)**: 该函数返回X中除NaN以外的数据的中位数，当X为向量时，返回X中除NaN以外各元素的中位数；当X为矩阵时，返回X中除NaN以外各列元素的中位数组成的向量。

**Y=nanmedian(X, DIM)**: 该函数返回在给定的维数DIM上的除NaN以外的中位数。

```
% chap7_21.m
```

```
clear all;
A1=[1 2 3 4]
m1=median(A1)          %向量的中位数
A2=[1 2 3;2 3 4;4 1 8]
m2=median(A2)          %列元素的中位数
m3=median(A2,2)         %行元素的中位数
```

## 数据的排序

数据的排序不仅包括数据的排序，还包括求数据的最大值和最小值等。下面分别进行介绍。

### 1. 数据的排序

在MATLAB中，采用函数sort( )对数据进行排序，

### 2. 按行进行数据排序

在MATLAB中，使用函数sortrows( )进行按行数据排序。

### 3. 求最大值和最小值

在MATLAB中，使用函数range( )求数据中最大值和最小值之差。使用函数minmax( )获取数据的最大值和最小值。

```
% chap7_23.m
```

```
clear all;
X=[1 3 4;8 3 5;2 7 4]
y1=sort(X)              %按列由小到大排序
y2=sort(X, 2)            %按行由小到大排序
y3=sort(X, 1, 'descend') %按列由大到小排序
[Y, I]=sort(X)
[Y, I]=sort(X, 2)
```

```
% chap7_24.m

clear all;
X=[1 3 4;8 3 5;2 7 4]
y1=sortrows(X)          %按行排序
y2=sort(X, 2)           %按指定的列排序
[Y, I]=sort(X)
```

```
% chap7_25.m
```

```
clear all;
X1=[1 3 4 10 3 5]        %向量
y1=range(X1)
X2=[1 3 5;2 4 6;8 4 3]    %矩阵
y2=range(X2)
y3=range(X2, 2)
```

```
% chap7_26.m
```

```
clear all;
X1=[2 5 4 12 3 15]        %向量
y1=minmax(X1)
X2=[1 3 5;2 4 6;8 4 3]    %矩阵
y2=minmax(X2)
```

## 期望和方差

期望和方差是刻画随机变量性质的两个最重要的数字特征，是概率论和数理统计中非常重要的内容。数字特征能够比较容易的估算出来，在理论和实践上都具有重要的意义。期望和方差的计算是概率论和数理统计的基础。

- 1、使用函数**mean( )**来计算样本数据的期望。
- 2、采用函数**var( )**来计算样本数据的方差。
- 3、采用函数**std( )**来计算样本数据的标准差。

```
% chap7_27.m
```

```
clear all;
X1=rand(1, 8)      %向量
y=mean(X1)
X2=rand(4, 7)      %矩阵
Y=mean(X2, 2)
```

```
% chap7_28.m

clear all;
A=rand(1, 8)          %向量
y=var(A)
B=rand(3, 5)          %矩阵
Y1=var(B)
Y2=var(B, 1)
W=[0.1 0.2 0.3]      %系数
Y3=var(B, W)

% chap7_29.m
```

```
clear all;
A=[2 3 4 1; 1 2 3 2; 8 1 2 3]
y1=std(A)
y2=std(A, 0)
y3=std(A, 1)
y4=std(A, 1, 2)
```

## 常见分布的期望和方差

下面对一些常见分布的期望和方差进行计算，主要包括二项分布、均匀分布、正态分布、指数分布、t分布和F分布等。

- 1、函数**binostat( )**计算二项分布的期望和方差。
- 2、函数**unifstat( )**计算均匀分布的期望和方差。
- 3、函数**normstat( )**计算正态分布的期望和方差。
- 4、函数**expstat( )**计算指数分布的期望和方差。
- 5、函数**tstat( )**计算t分布的期望和方差。
- 6、函数**fstat( )**计算F分布的期望和方差。



```
% chap7_30.m

clear all;           I
n1=100;
p1=0.3;
[m1, v1]=binostat(n1, p1)
n2=logspace(1, 4, 4)
[m2, v2]=binostat(n2, 1./n2)
```

```
% chap7_31.m

clear all;
a1=1;
b1=5;
[m1,v1]=unifstat(a1,b1)
a2=1:5;
b2=2.*a2;
[m2,v2]=unifstat(a2,b2)
```

```
% chap7_32.m
```

```
clear all;
n1=2;
n2=3;
[m1,v1]=normstat(n1,n2)
n3=1:4;
[m2,v2]=normstat(n3'*n3,n3'*n3)
```

```
% chap7_33.m
```

```
clear all;
mu1=2;
[m1,v1]=expstat(mu1)
mu2=1:5;
[m2,v2]=expstat(mu2)
```

```
% chap7_34.m
```

```
clear all;
n1=4;
[m1,v1]=tstat(n1)
n2=5:10;
[m2,v2]=tstat(n2)
```

```
% chap7_35.m
```

```
clear all;
n1=6;
n2=8;
[m1,v1]=fstat(n1,n2)
n3=5:10;
n4=7:12;
[m2,v2]=fstat(n3,n4)|
```

## 协方差和相关系数

### 1. 协方差

在MATLAB中，利用函数cov( )计算协方差，调用格式为：

C=cov(X)：输入参数可以为向量和矩阵。当X为向量时，该函数的返回值为该向量的方差。当输入参数X为矩阵时，矩阵的每一行代表一组样本数据，每一列代表一个变量，该函数的返回值为该矩阵的协方差矩阵。

C=cov(X, Y)：该函数计算输入参数X和Y的协方差矩阵，其中输入参数X和Y的行数和列数相同。

```
% chap7_36.m
```

```
clear all;
X1=rand(1, 5)
c1=cov(X1)
X2=rand(1, 5)
c2=cov(X2)
c3=cov(X1, X2)
X3=rand(4, 4)
c4=cov(X3)
c5=cov(X3(:, 1))|
```

## 2. 相关系数

在MATLAB中，采用函数corrcoef( )计算相关系数。

```
% chap7_37.m
```

```
clear all;
X=[1 2 3;3 4 6;7 4 2]
[R1, P1]=corrcoef(X)
x=[1 2 3 5 3 2];
y=[2 3 5 3 1 9];
R2=corrcoef(x, y)
```

## 偏斜度和峰度

### 1. 偏斜度

偏斜度是用来表示随机变量分布的形状与对称形式。

在MATLAB中，采用函数skewness( )计算样本数据的偏斜度。

```
% chap7_38.m
```

```
clear all;
X=randn([4, 5])
y1=skewness(X)
y2=skewness(X, 0)
y3=skewness(X, 0, 2)
```

## 2. 峰度

在MATLAB中，采用函数kurtosis( )计算随机变量的峰度。

```
% chap7_39.m
```

```
clear all;
X=randn([4, 5])
y1=kurtosis(X)
y2=kurtosis(X, 0)
y3=kurtosis(X, 0, 2)
```

## 参数估计

统计量中最重要、最常用的是均值和标准差，由于样本是随机变量，它们作为样本的函数自然也是随机变量，当用它们去推断总体时，有多大的可靠性就与统计量的概率分布有关，因此我们需要知道几个重要分布的简单性质。

参数估计包括点估计和区间估计。

### 点估计

用样本统计量来确定总体参数的一个数值。点估计分为矩估计法和极大似然估计法。评价估计好坏的标准有三个，分别是无偏性、有效性和一致性。

### 区间估计

区间估计会得到未知参数的置信区间，下面介绍均匀分布、指数分布、Beta分布和正态分布的参数估计。

- 1、利用函数unifit( )计算均匀分布的参数估计。
- 2、利用函数expfit( )计算指数分布的参数估计。
- 3、利用函数betafit( )计算Beta分布的参数估计。
- 4、采用函数normfit( )计算正态分布的参数估计。

```
% chap7_40.m

clear all;
X=unifrnd(2, 8, 20, 3)
[a, b, aci, bcil]=unifit(X, 0.05)

% chap7_41.m

clear all;
X=exprnd(5, 20, 4)
[parmhat1, parmcil1]=expfit(X, 0.05)
[parmhat2, parmcil2]=expfit(X, 0.01)

% chap7_42.m           I

clear all;
X=betarnd(7, 5, 100, 1);
[phat, pci]=betafit(X, 0.02)

% chap7_43.m

clear all;
X=normrnd(10, 2, 20, 3)
[mu1, sigma1, muci1, sigmacil1]=normfit(X, 0.05)
[mu2, sigma2, muci2, sigmacil2]=normfit(X, 0.01)
```

## 假设检验

在进行假设检验时，首先提出假设，然后在认为此假设成立的条件下，选择合适的统计量，最后做出判断。下面介绍常用的几种假设检验。

方差已知时正态总体均值的假设检验

在MATLAB中，如果已知总体的方差和均值时，使用函数 **ztest( )** 进行假设检验。

```
% chap7_45.m

clear all;
X=[490 513 514 513 511 499 515 512 491];
[H1, P1, CI1, STATS1]=ztest(X, 500, 10, 0.05, 0)    %ALPHA=0.05
[H2, P2, CI2, STATS2]=ztest(X, 500, 10, 0.1, 0)    %ALPHA=0.1
```

方差未知时正态总体均值的假设检验

在MATLAB中，如果总体的方差和均值未知时，使用函数  
**ttest( )**进行假设检验。

```
% chap7_46.m  
clear all;  
X=[300 324 305 290 295 291 310 315];  
[H, P, CI, STATS]=ttest(X, 305, 0.05, -1)      %ALPHA=0.05
```

两个正态分布均值差的检验

如果两个正态分布的方差未知但相等时，在MATLAB中用  
函数**ttest2( )**用于比较两个正态分布样本均值的假设检验

```
% chap7_47.m  
clear all;  
X=[302 304 305 310 320 299 298 301 315 313];  
Y=[305 314 320 315 313 308 318 325 301 312];  
[H, P, CI, STATS]=ttest2(X, Y, 0.05, -1)          %ALPH
```

两个分布一致性检验

在MATLAB中，利用函数**ranksum( )**函数进行两个分布一  
致性的检验，即秩和检验。

```
% chap7_48.m  
clear all;  
X=[3.2 7.0 2.8 4.5 7.5 2.3 5.8 1.4];  
Y=[2.3 5.6 9.2 1.2 5.9 3.4 2.8 8.4 1.5 3.2];  
[P, H, STATS]=ranksum(X, Y, 0.05)                %ALPH
```

## 方差分析

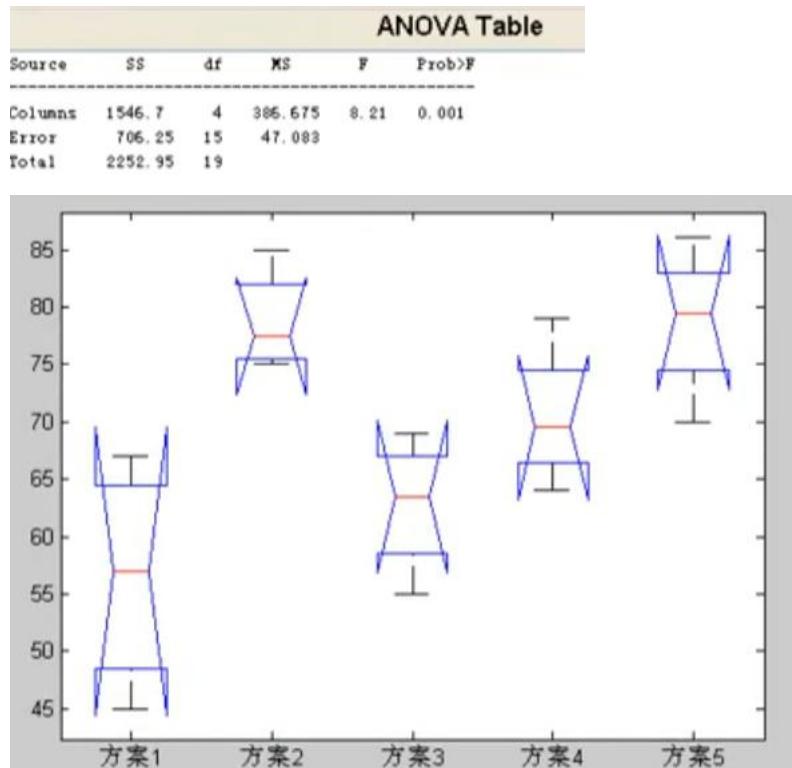
方差分析（Analysis of Variance, ANOVA）是数理统计中常用的数据处理方法之一。一个复杂的事物，例如产品的质量，受到很多因素的影响，这些因素既相互制约又相互依存。方差分析的目的是通过数据分析找到对该事物有显著影响的因素。本节只介绍一种或两种因素对实验结果有无显著影响的情况。只有一种因素的情形称为单因素方差分析，有两种因素的情形称为双因素方差分析。

### 单因素方差分析

单因素方差分析是指实验中只有一个因素发生改变。在 MATLAB 中，单因素方差分析可以采用函数 `anova1()` 来实现。

```
% chap7_49.m
```

```
clear all;
X=[62 75 62 79 80;67 85 69 64 70;...
    45 79 55 69 79;52 76 65 70 86]
group=['方案1';'方案2';'方案3';'方案4';'方案5']
[p, a, s]=anova1(X, group)
```



## 双因素方差分析

如果在某项实验中，实验结果受到两个因素的影响，则称该实验为双因素实验。对研究两种因素对实验结果的影响，需要进行双因素方差分析。

在MATLAB中，双因素方差分析可以采用函数anova2( )来实现。

```
% chap7_50.m  
clear all;  
X=[30 36 35.5 38.5;33.5 36.5 38 39.5;36 37.5 39.5 43];  
[p, a, s]=anova2(X)
```

ANOVA Table					
Source	SS	df	MS	F	Prob>F
Columns	79.063	3	26.3542	23.87	0.001
Rows	32.042	2	16.0208	14.51	0.005
Error	6.625	6	1.1042		
Total	117.729	11			

## 统计图绘制

MATLAB具有很强的数据可视化能力，能够将数理统计中的数据绘制成图形，用图形进行更加形象的描述。下面介绍如何利用MATLAB来绘制统计图。

## 正整数的频率表

在MATLAB中，采用函数tabulate( )产生正整数的频率表，该函数的调用格式为：T=tabulate(X)。输入参数为正整数构成的向量，返回值包含3列，第1列为向量的不重复的值；第2列为这些值出现的次数；第3列为这些值对应的频率。

```
% chap7_52.m  
clear all;  
X=[2 3 4 1 2 4 6 5 8 3 2 1 6];  
tabulate(X)|
```

Value	Count	Percent
1	2	15.38%
2	3	23.08%
3	2	15.38%
4	2	15.38%
5	1	7.69%
6	2	15.38%
7	0	0.00%
8	1	7.69%

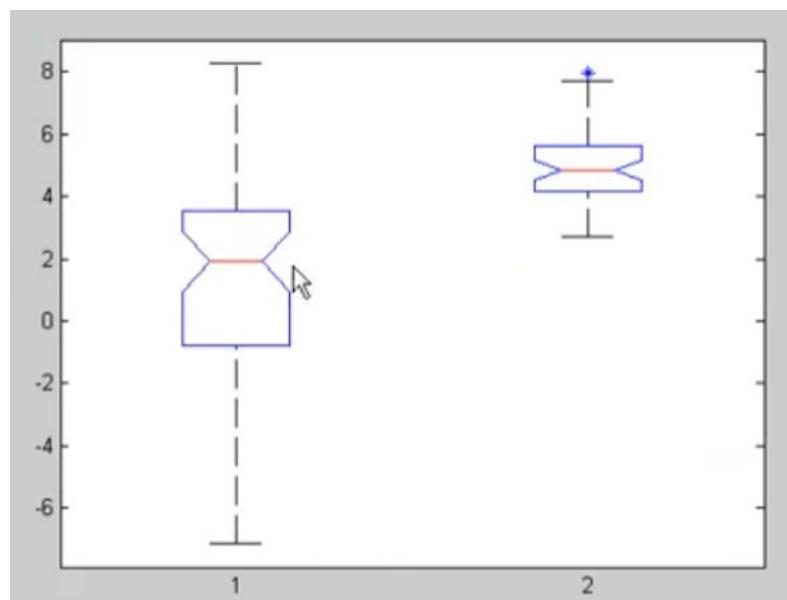
## 样本数据的盒图

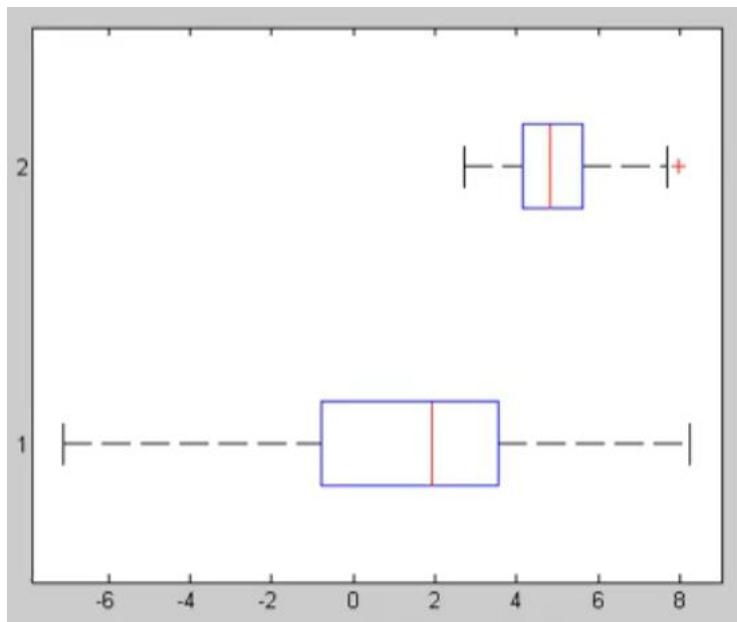
在MATLAB中，采用函数**boxplot( )**来绘制样本数据的盒图。该函数的调用格式为：**H=boxplot(X, 'param1', val1, 'param2', val2, ...)**。该函数的输入参数为矩阵，绘制每一列的盒图。

```
% chap7_53.m

clear all;
a=normrnd(1, 4, 50, 1);
b=normrnd(5, 1, 50, 1);
x=[a b];
figure;
boxplot(x, 'notch', 'on', 'symbol', 'b*', 'orientation'
figure;
boxplot(x, 'notch', 'off', 'symbol', 'r+', 'orientation

'orientation','vertical');
,'orientation','horizontal');
```





## 最小二乘拟合直线

在MATLAB中，使用函数lsline( )绘制最小二乘拟合直线，

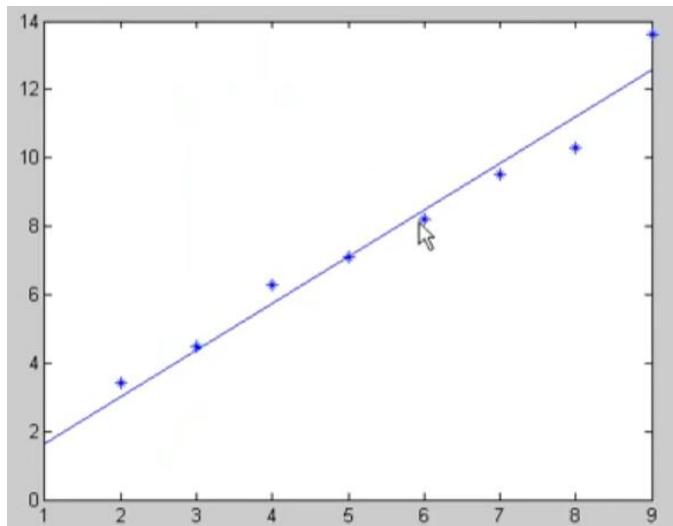
该函数的调用格式为：

**lsine:** 返回已知样本数据的最小二乘拟合直线，可以使用除'-'、'—'和'.'之外的其它线条样式绘制图形。

**h=lsine:** 该函数返回函数句柄。

```
% chap7_54.m
```

```
clear all;
X=[1.2 3.4 4.5 6.3 7.1 8.2 9.5 10.3 13.6];
figure;
plot(X, 'b*');
h=lsline
```



## 正态分布的概率图

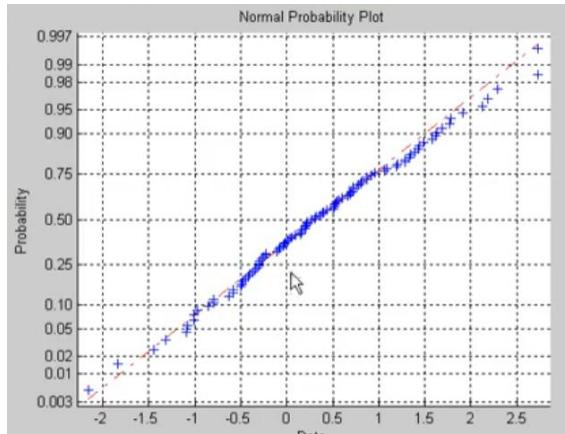
在MATLAB中，使用函数**normplot( )**绘制正态分布概率图，该函数的调用格式为：

**normplot(X)**: 如果输入参数为向量，则显示正态分布的概率图形；如果为矩阵，则显示每一列的正态分布概率图形。

**H=normplot(X)**: 该函数返回函数句柄。

```
% chap7_55.m
```

```
clear all;
X=normrnd(0, 1, 100, 1);
figure;
H=normplot(X)
```



## 经验累积分布函数图

在MATLAB中，使用函数**cdfplot( )**绘制经验累积分布函数图，该函数的调用格式为：

**cdfplot(X)**: 该函数绘制样本数据的累积分布函数图。输入参数可以为行向量或列向量，代表服从某种特定分布的随机样本。

**H=cdfplot(X)**: 该函数返回函数句柄。

**[H, stats]=cdfplot(X)**: 该函数返回样本数据的一些特征：

**stats.min**: 样本数据的最小值。

**stats.max**: 样本数据的最大值。

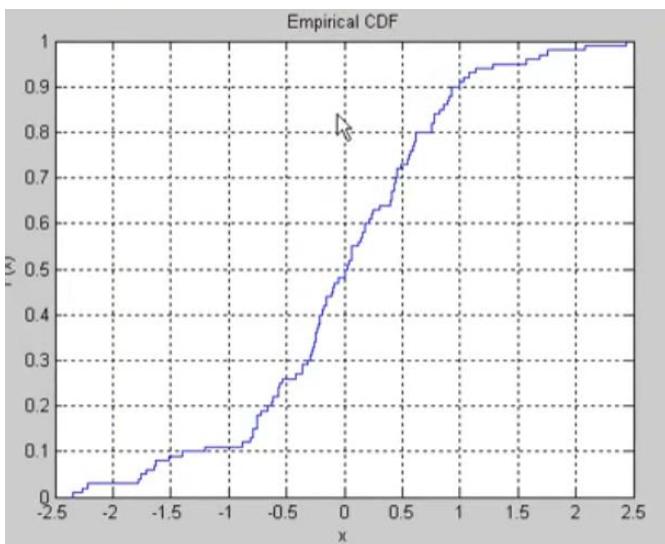
**stats.mean**: 样本数据的平均值。

**stats.median**: 样本数据的中位数。

**stats.std**: 样本数据的标准差。

```
% chap7_56.m

clear all;
X=normrnd(0, 1, 100, 1);
[h, stats]=cdfplot(X)
```



```
h =
174.0126

stats =
min: -2.3472
max: 2.4366
mean: -0.0133
median: 0.0193
std: 0.9040
```

//函数句柄

## 威布尔概率图

在MATLAB中，使用函数weibplot( )绘制威布尔概率图。

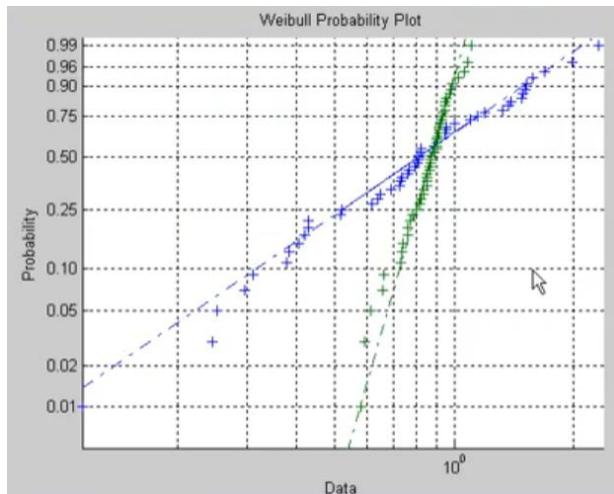
绘制威布尔概率图的目的是用图解的法估计来自威布尔分布的数据。该函数的调用格式为：

**weibplot(X):** 如果输入参数为向量，则显示威布尔概率图；如果为矩阵，则显示每一列的威布尔概率图。如果是威布尔分布数据，那么图形是直线；否则图形可能产生弯曲。

**H= weibplot (X):** 该函数返回所绘制直线的函数句柄。

```
% chap7_57.m

clear all;
a=weibrnd(1, 2, 50, 1);
b=weibrnd(3, 8, 50, 1);
X=[a b];
H=weibplot(X)
```



分位数-分位数图

在MATLAB中，使用函数`qqplot()`绘制分位数-分位数图。

该函数的调用格式为：

`qqplot(X)`: 该函数的输入参数为行向量或列向量，绘制样本数据和标准正态分布的分位数-分位数图。

`qqplot(X, Y)`: 该函数的输入参数和的分位数-分位数图。

`H= qqplot (X, Y)`: 该函数返回所绘制直线的函数句柄。

```
% chap7_58.m
```

```
clear all;
X=normrnd(1, 4, 50, 1);
Y=normrnd(5, 9, 50, 1);
figure;
H1=qqplot(X)
figure;
H2=qqplot(X, Y)
```

当前图形加一条参考线

在MATLAB中，使用函数`refline( )`给当前图形加一条参考线。该函数的调用格式为：

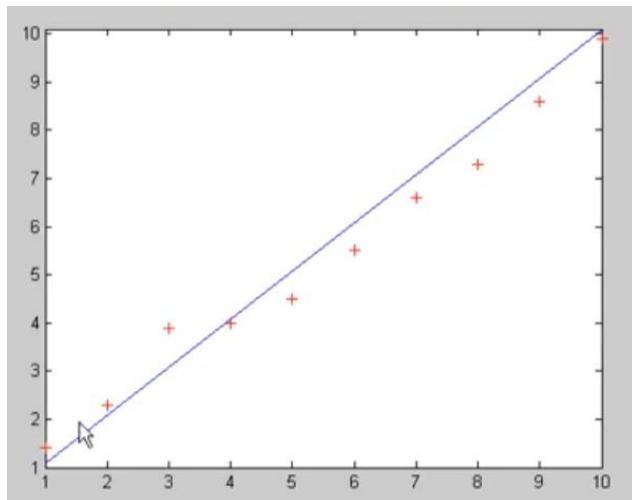
`refline(slope, intercept)`: 该函数的输入参数`slope`为直线的斜率，`intercept`为直线的截距。

`refline(slope)`: 该函数的输入参数`slope=[a, b]`，图中添加的直线为 $y=a*x+b$ 。

`H=refline(slope)`: 该函数返回所绘制直线的函数句柄。

```
% chap7_59.m
```

```
clear all;
X=[1.4 2.3 3.9 4.0 4.5 5.5 6.6 7.3 8.6 9.9];
figure;
plot(X, 'r+');
h=refline(1, 0.1)
```



当前图形加多项式曲线

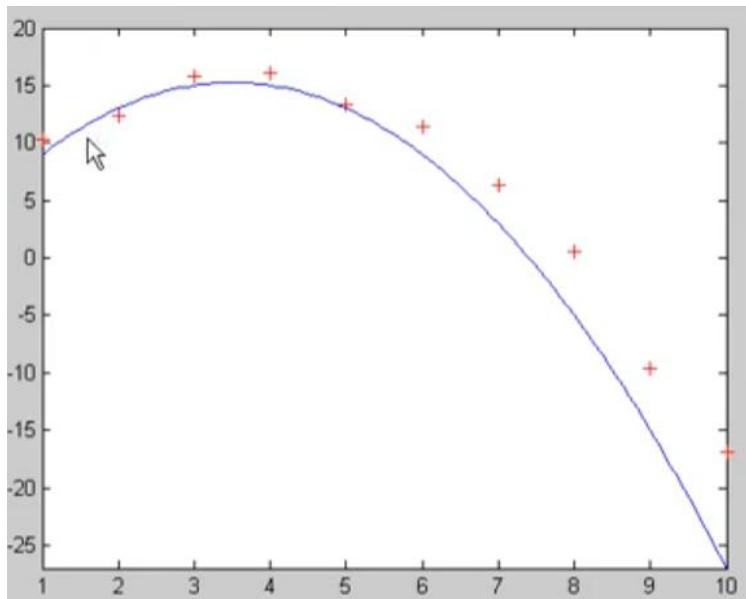
在MATLAB中，使用函数`refcurve( )`给当前图形加一条多项式曲线。该函数的调用格式为：

`refcurve(p)`: 该函数的输入参数`p`为多项式系数组成的向量，多项式的系数由高到低进行排列。

`H=refcurve(p)`: 该函数返回所绘制曲线的函数句柄。

```
% chap7_60.m

clear all;
X=[10.3 12.4 15.8 16.1 13.4 11.4 6.3 0.6 -9.6 -16.9];
figure;
plot(X, 'r+' );
p=[-1 7 3];
h=refcurve(p)
```



样本的概率图形

在MATLAB中，使用函数**capaplot( )**绘制样本的概率图形，该函数返回随机变量落入指定区间内的概率。该函数的调用格式为：

**capaplot(data, specs):** 该函数的输入参数**data**为所给的样本数据，参数**specs**为指定的区间，函数返回落入该区间的概率。

**[p, h]=capaplot(data, specs):** 该函数有两个返回值，**p**为指定区间的概率，**h**为所绘制图形的函数句柄。

```
% chap7_61.m
```

```
clear all;
data=normrnd(0, 1, 100, 1);
s1=[-2.5, +2.5];
s2=[-inf, 2];
figure;
[p, h]=capaplot(data, s1)
figure;
[p, h]=capaplot(data, s2)
```

## 带有正态密度曲线的直方图

在MATLAB中，使用函数**histfit( )**绘制带有正态密度曲线的直方图。该函数的调用格式为：

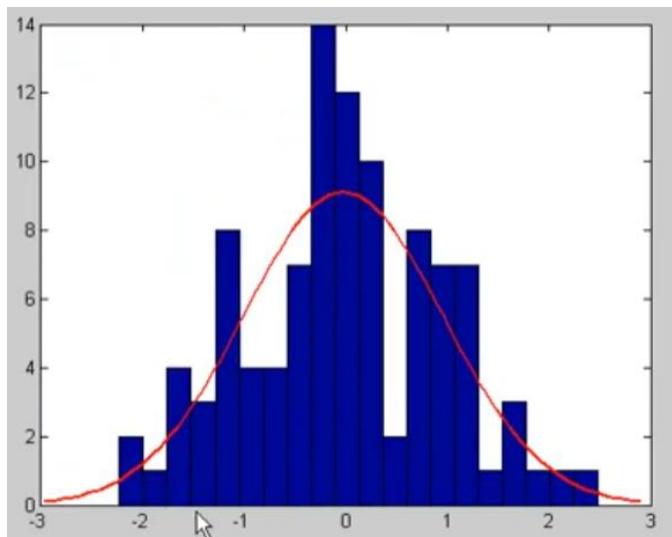
**histfit(data, nbins)**: 该函数绘制带有正态密度曲线的直方图，输入参数**data**为所给的样本数据的向量，参数**nbins**为指定的BAR个数，默认值为**data**中数据个数的平方根

。

**h=histfit(data, nbins)**: 该函数返回图形的函数句柄。

```
% chap7_62.m
```

```
clear all;
data=normrnd(0, 1, 100, 1);
figure;
h=histfit(data, 20)
```



## 在指定区间绘制正态密度曲线

在MATLAB中，使用函数**normspec( )**在指定的区间绘制正态密度曲线。该函数的调用格式为：

**normspec(specs)**: 该函数在指定的区间**specs**绘制标准正态密度曲线，期望**mu**和标准差**sigma**分别为0和1。

**normspec(specs, mu, sigma)**: 该函数对正态分布的期望和标准差进行设置。

**[p,h]= normspec(specs, mu, sigma)**: 该函数返回所绘制图形的函数句柄。

```
% chap7_63.m  
  
clear all;  
s=[-inf 7];  
mu=2;  
sigma=4;  
[p, h]=normspec(s, mu, sigma)
```

## 本章小结

本章详细介绍了利用**MATLAB**进行概率和数理统计分析，重点讲解了随机数的产生，随机变量的概率密度函数和数字特征（包括期望、方差、协方差、相关系数、偏斜度和峰度等），以及假设检验、方差分析等。最后详细介绍了如何绘制统计图。

# MATLAB 符号计算

在MATLAB中，提供了强大的符号运算功能，并且有专门的符号数学工具箱Symbolic Math Toolbox。此外，在MATLAB中，还可以通过maple.m和map.m两个接口和Maple相连。MATLAB的符号计算功能非常强大。在MATLAB中，符号计算的函数主要分为以下几类：

符号表达式

符号矩阵操作

符号微积分

符号积分变换



符号方程求解

符号函数的绘图

图形化符号函数计算器

## 符号运算入门

本节将详细介绍如何生成符号型数据变量，利用符号变量产生函数表达式和方程，以及符号变量的基本运算。下面将分别予以详细介绍。

### 符号变量的创建

符号数学工具箱中定义了一种新的数据类型：sym类。sym类的实例就是符号对象，用来存储代表符号的字符串。

在MATLAB中，提供了两个建立符号对象的函数：函数sym( )和函数syms( )，两个函数的用法不同。

### 符号变量的创建

在MATLAB中，可以利用函数sym( )建立单个的符号变量。  
采用函数syms( )一次定义多个符号变量。

此外，可以通过函数class( )获取符号变量的类型。

```
%chap8_1.m
```

```
clear all;
a1=sqrt(5)
a2=sqrt(sym(5))
a3=double(a2)
a4=sym(3)/sym(8)
a5=3*a4
a6='3/8'
ca1=class(a1)
ca2=class(a2)
ca3=class(a3)
ca4=class(a4)
ca5=class(a5)
ca6=class(a6)
```

```
%chap8_2.m
```

```
clear all;
t=0.13
y1=sym(t)
y2=sym(t,'r')
y3=sym(t,'f')
y4=sym(t,'e')
y5=sym(t,'d')
```

```
%chap8_3.m
```

```
clear all;
x=sym('x');
y=sym('y');
syms a b;
classx=class(x)
classy=class(y)
classa=class(a)
classb=class(b)
```

```
%chap8_4.m
```

```
clear all;
a=sym('a');
b=sym('b');
c=sym('c');
x=sym('x');
f1=a*x^2+b*x+c
g=f1^2+3*f1+5
f2=sym('a*x^2+b*x+c')
```

```
%chap8_5.m
```

```
clear all;
syms a b c x;
f1=a*x^2+b*x+c
g=f1^2+3*f1+5
```

## 符号函数和符号方程

在MATLAB中，符号表达式是由符号常量、符号变量、符号函数运算符以及专用函数连接起来的符号对象。符号表达式有两类：符号函数和符号方程。方程与函数的区别为：函数是一个代数式，而方程是一个等式。

```
%chap8_6.m
```

```
clear all;
syms x y;
f1=4*(x-2)^2+sin(x+y)+5^x
f2=f1/(7*x)
f3=sym(f1/(7*x))
```

```
% chap8_7.m
```

```
clear all;
syms x y;
e1=sym('4*(x-2)^2+5^y=0')
e2=sym('5*x^2+4*x+2=0')
|
```

## 简单实例分析

求方程的根、函数的微分和积分等是高等数学里面最基础的内容，下面介绍如何通过MATLAB的符号计算进行求解。

### 求解一元二次方程的根

对于一元二次方程， $ax^2+bx+c=0$ ，其中 $a \neq 0$ ，根据一元二次方程的求根公式，得到方程的两个根。当 $b^2-4ac \geq 0$ 时，该方程有实根；当 $b^2-4ac < 0$ 时，方程有两个共轭的复根。

```
%chap8_8.m  
clear all;  
y1=solve('a*x^2+b*x+c=0')  
y2=solve('3*x^2+8*x+5=0')  
y3=solve('x^2-2*x+6=0')
```

## 求导数

在MATLAB中，利用函数**diff( )**来求导数。根据复合函数的求导公式进行计算。

```
clear all;  
syms x;  
y=diff(x^5*cos(x)+cos(pi/6))
```

## 计算不定积分

在MATLAB中，利用函数**int( )**求不定积分。

```
% chap8_10.m  
clear all;  
syms x;  
y=int(exp(sin(x))*cos(x), 'x')
```

## 计算定积分

利用函数**int( )**也可以求定积分。

```
% chap8_11.m  
clear all;  
syms x;  
y=int('sin(x)', x, 0, pi)
```

## 求解一阶微分方程

求解微分方程 $dy/dt=ay$ ，该方程为可分离变量微分方程，进行变量分离，然后进行积分。采用函数**dsolve( )**求解微分方程。

```
% chap8_12.m  
clear all;  
syms a y;  
y=dsolve('Dy=a*y')  
y=dsolve('Dy=a*y', 'y(0)=2')
```

## 符号运算精度

在MATLAB中，采用函数`digits( )`和`vpa( )`来实现任意精度的符号运算。下面分别进行介绍：

`digits(d)`: 调用该函数后，符号对象的近似解的精度变成d位有效数字，参数d的默认值为32位。

`D=digits`: 调用该函数后，得到当前采用的数值计算的精度。

`R=vpa(A)`: 该函数计算符号矩阵A的近似解，精度为函数`digits(d)`指定的有效位数。

`R=vpa(A, d)`: 该函数计算符号矩阵A的近似解，有效位数由参数d决定。

```
% chap8_13.m
```

```
clear all;
digits 1
a1=sym(1.3, 'd')
digits(40)
digits
a2=sym(1.3, 'd')
```

```
% chap8_14.m
```

```
clear all;
a=vpa(hilb(2))
b=vpa(hilb(3), 6)
c=vpa(pi)
d=vpa(pi, 100)
```

## 符号表达式的操作

符号表达式可以进行加减乘除的四则运算，此外，还可以对符号表达式进行因式分解、展开、合并同类项、获取分子和分母等，这些操作都非常的简单方便，下面分别进行介绍。

## 符号表达式的基本运算

在MATLAB中，采用了函数重载技术，使得符号表达式的运算符和基本函数与数值计算中的运算符和基本函数几乎完全相同。符号表达式可以进行“+”、“-”、“\*”、“/”四则运算。

符号表达式的比较中，只有运算符“==”和“~=”代表“等于”和“不等于”。当结果为“真”时，返回值为1，否则返回值为0。

```
% chap8_15.m

clear all;
syms x y z;
f1=sin(x)+acos(y)
f2=sqrt(x)+exp(y)
f3=log2(x)+log(y)
g1=f1+f2
g2=f1*f2
g3=f1/f2
f1==f2
```

## 符号表达式的常用操作

符号表达式的常用操作，包括：

利用函数`findsym( )`寻找符号变量。

利用函数`factor( )`进行符号多项式的因式分解。

利用函数`expand( )`进行符号表达式的展开。

利用函数`collect( )`进行符号表达式中同类项的合并。

利用函数`horner( )`将符号多项式转换成嵌套形式。

利用函数`numden( )`获取符号表达式的分子和分母。

```
% chap8_16.m
```

```
clear all;
syms m n x y;
f=m^2+5*n+sin(x+5)+exp(5*y)+4+5*i
findsym(f)
findsym(f, 2)
findsym(f, 3)
```

```
% chap8_17.m
```

```
clear all;
syms x y;
factor(x^4-y^4+x^2-y^2)
for i=1:8
    disp(factor(x^i-1))
end
```

```
% chap8_18.m
```

```
clear all;
y1=factor(1234567890)
y2=factor(sym('12345678901234567890'))
```

```
% chap8_19.m

clear all;
syms x y;
f1=(x-y)^2+(x+y)^3          %多项式展开
g1=expand(f1)
f2=sin(x-3*y)                %三角函数展开
g2=expand(f2)
f3=exp(2*x^2+4*y)            %指数函数展开
g3=expand(f3)

% chap8_20.m

clear all;
syms x y;
f=-1/4*y*exp(-2*x)+2/5*x*exp(-2*x)+3*exp(-2*x);
y1=collect(f,exp(-2*x))
y2=collect(x^2*y+y*x+3*y^2*x^3-2*x^2-2*x)
y3=collect(x^2*y+y*x+3*y^2*x^3-2*x^2-2*x, y)

clear all;
syms x y;
f1=x^4+6*x^3+4*x^2-4
g1=horner(f1)
f2=[x^2-2*x+4 x^3-3*x^2+5; x^2-4*x+6 4*x^2+6*x]
g2=horner(f2)

% chap8_22.m

clear all;
syms x y;
[n1, d1]=numden(sym(3/4))
f=(x/y+3*y/x);           %
[n2, d2]=numden(f)
```

## 符号表达式的化简

在MATLAB中，采用函数simplify( )和函数simple( )进行符号表达式的化简，下面分别进行介绍。

**函数simplify( )**是一个具有普遍意义的工具，能够对包含和式、方根、分数的乘方、指数函数、对数函数、三角函数等的表达式进行化简。

**函数simple( )**也能进行符号表达式的化简，该方法比函数simplify( )要简单，所得到的结果也比较合理。

```
% chap8_23.m

clear all;
syms x y;
f1=(2*x^4-2*x)/(2*x^2+4*x+2);
sf1=simplify(f1)
f2=2*cos(x)^2-3*sin(x)^2-2;
sf2=simplify(f2, 100)
sf3=simplify((sin(x)^2+cos(x)^2)
```

```
% chap8_24.m
```

```
clear all;
syms x;
simple(cos(x)^2-sin(x)^2)
```

```
% chap8_25.m
```

```
clear all;
syms x;
[r1, how1]=simple(cos(3*acos(2*x)))
[r2, how2]=simple(cos(x)^2-sin(x)^2)
[r3, how3]=simple(cos(2*x)^2+sin(2*x)^2)
```

## 符号表达式的替换

在MATLAB中，可以采用函数**subexpr( )**和函数**subs( )**进行符号替换，让符号表达式的输出变的简单。

**函数subexpr( )**将符号表达式中重复出现的字符串用变量代替。

**函数subs( )**将符号表达式中重复出现的字符串用数值或字符串替换。

```
% chap8_26.m
```

```
clear all;
syms a x;
t=solve('a*x^3+a*x^2+3*x=0')
[y1, sigma1]=subexpr(t)|
sig=a;
[y2, sigma2]=subexpr(t, sig)
[y3, sigma3]=subexpr(t, 'abc')
```

```
% chap8_27.m

clear all;
syms x y;
f=x^2+3*x+5*x*y+6*y+2*y^2;
g1=subs(f, x, 3)
g2=subs(f, 3)
g3=subs(f, y, 5)    I
x=5;
g4=subs(f)
```

% chap8\_28.m

```
clear all;
syms x y;
f=x^2+3*x+5*x*y+6*y+2*y^2
g1=subs(f, {x, y}, {3, 5})
g2=subs(exp(x*y), x, magic(4))
```

## 反函数运算

在MATLAB中，采用函数`finverse()`进行反函数运算。该函数的调用格式为：

**g=finverse(f):** 该函数将会计算输入参数f的反函数，其中f为符号表达式，以默认的变量为自变量。函数的返回值g也是一个符号表达式。

**g=finverse(f, v):** 输入参数v是一个符号变量，是函数f中的变量，且该函数的返回值g以v为自变量。

% chap8\_29.m

```
clear all;
syms x y;
f=sin(x)+cos(y)+5
g1=finverse(f)
g2=finverse(f, x)
g3=finverse(f, y)    I
% fg2=simplify(compose(g2, f))
% fg3=simple(compose(g3, f))
```

## 复合函数运算

在数学计算中，经常会遇到求复合函数的情况。例如，函数 $z=f(y)$ ，而该函数的自变量y又是一个关于x的函数，即 $y=g(x)$ 。此时， $z=f(g(x))$ ，z是关于x的一个复合函数。

在MATLAB中，采用函数`compose()`进行符合函数运算。

```
% chap8_30.m

clear all;
syms x y z t u;
f=cos(x/t);
g=sin(y/u);
c1=compose(f, g);
c2=compose(f, g, z);
c3=compose(f, g, x, z);
c4=compose(f, g, t, z);
c5=compose(f, g, x, y, z);
c6=compose(f, g, x, u, z);
c7=compose(f, g, t, u, z);
```

## 符号矩阵的计算

符号矩阵也是一种符号表达式，前面介绍的符号表达式运算都可以用于符号矩阵，需要注意的是这些函数作用于符号矩阵时，是分别作用于矩阵的每一个元素。由于符号矩阵是一个矩阵，所以符号矩阵还能进行有关矩阵的运算。此外，曾介绍过的许多应用于数值矩阵的函数，也可直接应用于符号矩阵。

## 符号矩阵的生成

在MATLAB中，可以采用函数**sym( )**直接产生符号矩阵，也可以将数值矩阵转换为符号矩阵。下面介绍如何生成符号矩阵。

1. 采用函数**sym( )**直接生成符号矩阵
2. 利用函数**sym( )**将数值型变量转换为符号型变量

```
% chap8_31.m

clear all;
syms x;
A1=sym(' [1.5 x sin(x);cos(3)*2/4 4*x exp(x)]');
size(A1)
A2=sym(' [1/4, 2/6, 3/5;1/3, 8/9, 8/4;2/3, 3/5, 5/7];
size(A2)
```

```
% chap8_32.m

clear all;
syms x;
A1=sym(' [1.5 x sin(x);cos(3)*2/4 4*x exp(x) ]');
a=double(5);
A2=A1+a;
M1=magic(4);
A3=sym(M1);
M2=[0.4 0.5 0.6;3.12 pi sin(1);log(2) cos(1)];
A4=sym(M2)
```

## 符号矩阵的四则运算

对于符号变量和的四则运算有：

**A+B**和**A-B**: 实现符号矩阵的加法和减法。如果**A**和**B**为同类型的矩阵（具有相同的行数和列数）时，分别对对应的元素进行加减运算。如果**A**和**B**中有一个为标量，则矩阵中的每一个元素和该标量进行加法或减法。

**A\*B**: 实现矩阵**A**和**B**的乘法，要求矩阵**A**的列数必须等于矩阵**B**的行数。如果**A**和**B**中有一个为标量，则将矩阵中的每一个元素乘以该标量。

**A\B**: 实现矩阵的左除法。 $X = A \setminus B$ 为符号线性方程组 $A^*X = B$ 的解。另外， $A \setminus B$ 近似等于 $\text{inv}(A)^*B$ 。如果 $X$ 不存在或不唯一，则系统显示警告信息。 ↴

**B/A**: 实现矩阵的右除法。 $X = B / A$ 为符号线性方程组 $X^*A = B$ 的解。 $B / A$ 近似为 $B^*\text{inv}(A)$ 。如果 $X$ 不存在或不唯一，则系统显示警告信息。

```
% chap8_33.m

clear all;
syms x;
A1=sym(' [1/2 2/5 3/2 3/7 4;1.4 6 7]');
B1=sym(magic(3));
C1=A1+B1;
C2=A1-B1;
C3=A1+5;
D1=A1*B1;
D2=A1*5;
```

```
% chap8_34.m

clear all;
syms x;
A=sym(' [2 5 3;1 6 4;5 6 7]');
B=sym(magic(3));
C1=A\B | %左除
C2=inv(A)*B
D1=B/A %右除
D2=B*inv(A)
```

## 符号矩阵的线性代数运算

符号矩阵和数值矩阵非常的类似，也可以求符号矩阵的转置、秩、逆矩阵、行列式和特征值分解等。下面进行详细的介绍。

1. 符号矩阵的转置
2. 符号方阵的幂运算
3. 符号矩阵的秩
4. 符号方阵的逆矩阵和行列式计算
5. 符号方阵的特征值分解
6. 约当标准型
7. 符号矩阵的奇异值分解



```
% chap8_35.m

clear all;
A1=sym(magic(4))
B1=A1';
C1=A1.';
A2=sym([2+3i, 3; 2-5i, 8])
B2=A2';
C2=A2.'|
```

```
% chap8_36.m
```

```
clear all;
a=sym(' [x 2*x 3;3 8 x;2.0 x 6]');
y1=a^2
b=sym(' [2 4;3 1]')
y2=2^b
```

```
% chap8_37.m

clear all;
syms x y;
f1=sym(' [1, x^2, 3; exp(x), x+y, y; 3+x, sin(x), cos(
f2=sym(' [1, x^2, 3; exp(x), x+y, y]')

g1=rank(f1)
g2=rank(f2)
```

```
% chap8_38.m
```

```
clear all;
syms x;
A1=sym(magic(3))
Y1=inv(A1)
det(A1)
A2=sym([1-x, x, x-1; x, x-2, x+3; x, x, 0])
Y2=inv(A2)
det(A2) I
```

```
% chap8_39.m
```

```
clear all;
syms x;
A1=sym(magic(3))
E1=eig(A1)
[V1, D1]=eig(A1)
A2=sym([1 2 3; 2 3 4; 3 4 5])
E2=eig(A2)
[V2, D2]=eig(A2)|
```

```
% chap8_40.m
```

```
clear all;
A=sym([1, 2, 1; 1, 2, 1; 3, 1, 1])
[V, J]=jordan(A)
V1=vpa(V, 8)
J1=vpa(J, 8)
J2=vpa(V\A*V, 8)| I
//只能符号? ? ?
```

```
% chap8_41.m
```

```
clear all;
A=sym(magic(4))
svd(A)
vpa(svd(A))
svd(vpa(A))
[U, S, V]=svd(A)
```

## 符号微积分

### 符号表达式的微分运算

在MATLAB中，采用函数**diff( )**进行微分和求导运算，利用**Jacobian**函数**jacobian( )**实现对多元符号函数的求导。

```
% chap8_42.m
```

```
clear all;
syms x y;
f=5*x^4+y*sin(x)+x*cos(y)+6
g1=diff(f)
g2=diff(f, 4)           I
g3=diff(f, x, 4)
g4=diff(f, y, 4)
```

```
% chap8_43.m
```

```
clear all;
syms x y;
f=[5*x^4 y*sin(x);4*x x*cos(y)]
g1=jacobian(f, x)
g2=jacobian(f, [x; y])
```

### 符号表达式的极限

在MATLAB中，采用函数**limit( )**求符号表达式的极限。

```
% chap8_44.m
```

```
clear all;
syms x h;
y1=limit((cos(x+h)-cos(x))/h, h, 0)
y2=limit(((x+h)^3-x^3)/h, h, 0)
```

## 符号表达式的积分

在MATLAB中，采用函数int( )来计算符号表达式的不定积分和定积分。该函数的调用格式为：

**R=int(S):** 该函数计算S的不定积分，输入参数S可以是符号表达式或符号矩阵。没有指定积分变量和积分阶数，系统按函数findsym( )得到的默认变量对S求不定积分。

**R=int(S, v):** 该函数对S中指定的符号变量v求不定积分。需要注意的是，函数的返回值R只是其中的一个原函数，后面没有带任意常数C。

**R=int(S, a, b):** 该函数计算S在闭区间[a, b]上的定积分。由于没有指定积分变量，系统按函数findsym( )得到的默认变量计算定积分。a和b分别表示定积分的下限和上限，可以是两个具体的数，也可以是一个符号表达式，还可以是无穷大（inf）。

**R=int(S, v, a, b):** 该函数计算S对于变量v在区间[a, b]上的定积分。

```
% chap8_45.m
```

```
clear all;
syms x y;
f1=cos(x)+cos(y)
g1=int(f1)
g2=int(f1, x)
g3=int(f1, y)
```

```
% chap8_46.m
```

```
clear all;
syms x;
f1=1/x^2+sin(x)
g1=int(f1, 1, 2)
g2=int(f1, x, 1, 2)
f2=1/x^2;
g3=int(f2, x, 1, +inf)
```

## 级数的求和

在MATLAB中，采用函数symsum( )进行级数符号的求和。

该函数的调用格式为：

**symsum(S):** 该函数没有指定求和的符号变量，参数S表示级数的通项。

**symsum(S, v):** 该函数对变量v进行级数求和。

**symsum(S, a, b):** 该函数对默认变量从a到b进行级数求和。

**symsum(S, v, a, b):** 该函数对于变量v从a到b进行级数求和

```
% chap8_47.m

clear all;
syms x k;
y1=symsum(k)| I
y2=symsum(k, 1, 5)
y3=symsum(sin(x)+cos(k), x, 1, 5)
y4=symsum(k, 5, 4)
y5=symsum(1/k^2)
y6=symsum(1/k^2, 1, Inf)
```

泰勒级数

在MATLAB中，采用函数taylor( )求符号表达式的泰勒级数展开式，该函数的调用格式为：

**taylor(f):** 计算函数f在默认变量等于0处作默认为5阶的泰勒展开式。

**taylor(f, n):** 计算函数f在默认变量为0处做n-1阶的泰勒展开式，  
， 的默认值为6。

**taylor(f, a):** 计算函数f在默认变量等于a处做默认为5阶的泰勒展开式，a的默认值为0。

**taylor(f, n, a):** 计算函数f在默认变量等于a处做n-1阶的泰勒展开式。

**taylor(f, n, v, a):** 计算函数f在变量等于a处做n-1阶的泰勒展开式，变量变为v。

```
% chap8_48.m
```

```
clear all;
syms x y;
f1=taylor(exp(x))
f2=taylor(exp(x), 8)
f3=taylor(exp(x), 3, 6)
f4=taylor(exp(x), y)
```

符号表达式积分变换

在MATLAB中，提供了丰富的积分变换函数，本节将介绍MATLAB提供的常见积分变换，主要包括傅立叶变换及其反变换、拉普拉斯变换及其反变换和Z变换及其反变换。

## Fourier 变换及其反变换

时域中的信号  $f(t)$  与它在频域中的 Fourier 变换  $F(\omega)$  之间存在如下的关系。

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-j\omega t} dt$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{j\omega t} d\omega$$

在 MATLAB 中，采用函数 `fourier( )` 计算傅立叶变换，采用函数 `ifourier( )` 计算傅立叶变换的反变换。

```
% chap8_49.m

clear all;
syms t v w x;
F1=fourier(1/t)
F2=fourier(exp(-x^2), x, t)
F3=fourier(exp(-t)*sym('heaviside(t)'), v)
F4=fourier(diff(sym('F(x)'), x, w)

% chap8_50.m

clear all;
syms u v w x;
f1=ifourier(w*exp(-3*w)*sym('heaviside(w)'), w)
f2=ifourier(1/(1+w^2), u)
f3=ifourier(v/(1+w^2), v, u)
f4=ifourier(fourier(sym('f(x)'), x, w), w, x)
```

## Laplace 变换及其反变换

对于函数  $f(x)$  进行拉普拉斯 (Laplace) 变换的公式为：

$$L(s) = \int_0^{\infty} f(t) e^{-st} dt$$

对函数  $L(s)$  进行拉普拉斯反变换的公式为：

$$F(t) = \int_{c-i\infty}^{c+i\infty} L(s) e^{st} ds$$

在 MATLAB 中，实现拉普拉斯变换的函数为 `laplace( )`，进行拉普拉斯反变换的函数为 `ilaplace( )`。

```
% chap8_51.m

clear all;
syms s t w x a;
f1=laplace(x^3)
f2=laplace(sin(w*x), t)
f3=laplace(exp(a*x), s)
f4=laplace(cos(w*x), w, t)|

% chap8_52.m

clear all;
syms s t a w x y;
f1=ilaplace(1/(s-1))
f2=ilaplace(1/(t^2+1), y)
f3=ilaplace(1/(s-a), y)
f4=ilaplace(y/(y^2+w^2), y, x)
f5=ilaplace(sym(' laplace(F(x), x, s)'), s, x)
```

## Z 变换及其反变换

对于离散序列 $f(n)$ 的Z变换为

$$F(z) = \sum_{n=0}^{\infty} f(n)z^{-n}$$

对于 $F(z)$ 的Z反变换为：

$$f(n) = Z^{-1}\{F(z)\}$$

在MATLAB中，利用函数`ztrans( )`进行Z变换，利用函数`iztrans( )`进行Z反变换。

```
% chap8_53.m

clear all;
syms k n w z;
f1=ztrans(2^n)
f2=ztrans(sin(k*n), w)
f3=ztrans(cos(k*n), k, z)

% chap8_54.m

clear all;
syms x k n w z;
f1=iztrans(z/(z-2))
f2=iztrans(exp(x/z), z, k)
f3=iztrans(sym(' ztrans(f(n), n, z)'), z, n)
```

## 符号方程的求解

符号运算不仅能够求解方程，还可以求解方程组。下面介绍如何利用函数**solve( )**求解符号代数方程组，函数**dsolve( )**求解微分方程组。

### 符号代数方程组的求解

在MATLAB中，利用函数**solve( )**求解一般符号代数方程组。

该函数的调用格式为：

**solve('eqn1')**: 该函数求解单个的方程eqn1。

**solve('eqn1', 'eqn2', ..., 'eqnN')**: 该函数用来求解由N个方程组成的方程组。

**solve('eqn1', 'eqn2', ..., 'eqnN', 'var1', 'var2', ..., 'varN')**:

该函数中var1等表示符号变量，以这些变量为自变量，进行方程的求解。

```
% chap8_55.m
```

```
clear all;
syms x y;
e1=sym('y*sin(x)=5')
g1=solve(e1)
g2=solve(e1, 'y')
g3=solve('y*sin(x)=5')
g4=solve('y*sin(x)=5', 'y')
```

```
% chap8_56.m
```

```
clear all;
syms x y;
e1=sym('x^2+4*x+y^2+5*y-6=0')
e2=sym('x^2+2*x+y^2-4=0')
g=solve(e1, e2)
g.x
g.y
[x, y]=solve('x^2+4*x+y^2+5*y-6=0', 'x^2+2*x+y
```

```
% chap8_57.m
```

```
clear all;
syms x y a b;
e1=sym('a*x+b*y-6=0')
e2=sym('b*x+a*y-4=0')
g1=solve(e1, e2)
g.x
g.y
g2=solve(e1, e2, 'a', 'b')
g.a
g.b
```

```
% chap8_58.m

clear all;
syms x y a b;
e1=sym('a*x+y-6')
e2=sym('5*x+a*y-4=0')
e3=sym('4*x+6*y-8=0')
[a, x, y]=solve(e1, e2, e3)
g=solve([e1, e2, e3])

% chap8_59.m

clear all;
syms x y;
e1=sym('5*cos(x+y)+x^2+y^2=6')
e2=sym('5*x+y-4=0')
[x, y]=solve(e1, e2)
```

## 微分方程的求解

在MATLAB中，采用函数`dsolve( )`进行微分方程的求解。

该函数的调用格式为：

`dsolve('eq1')`: 该函数对单个的微分方程`eq1`求解。

`dsolve('eq1, eq2, ...', 'cond1, cond2, ...', 'v')`: 该函数对由`eq1, eq2`等组成的微分方程组求解，初始条件为`cond1, cond2`等，自变量为`v`。如果不指定参数`v`，则系统默认以`t`为自变量。

```
% chap8_60.m

clear all;
dsolve('Dx=-a*x')
dsolve('Dx=-a*x', 'x(0)=1')
dsolve('Dx=-a*x', 'x(0)=1', 's')
```

% chap8\_61.m

```
clear all;
g1=dsolve('D2y=-a^2*y')
g2=dsolve('D2y=-a^2*y', 'y(0)=1')
g3=dsolve('D2y=-a^2*y', 'y(0)=1', 'y(pi)=1')
```

% chap8\_62.m

```
clear all;
g1=dsolve('D2y+2*Dy+2*y=0')
g2=dsolve('D2y+2*Dy+2*y=0', 'y(0)=1')
g3=dsolve('D2y+2*Dy+2*y=0', 'y(0)=1', 'Dy(0)=0')
```

```
% chap8_63.m

clear all;
[x1, y1]=dsolve('Dx=y', 'Dy=-x', 't')
[x2, y2]=dsolve('Dx=y', 'Dy=-x', 'x(0)=3', 'y(0)=0')

% chap8_64.m

clear all;
S=dsolve('Du=v, Dv=w, Dw=-u', 'u(0)=0, v(0)=0, w(0)=0')
[u, v, w]=dsolve('Du=v, Dv=w, Dw=-u', 'u(0)=0, v(0)=0, w(0)=0')
```

## 符号函数的图形绘制

在MATLAB中，对于符号方程，可以非常方便的绘制其图形。本节将介绍符号函数的图形绘制，常用的绘图函数如表8.1所示。

表 8.1 符号函数的绘图函数<sup>②</sup>

函数名 <sup>③</sup>	函数的功能 <sup>④</sup>
fplot <sup>⑤</sup>	一元函数绘图 <sup>⑥</sup>
ezplot <sup>⑤</sup>	二元函数绘图 <sup>⑥</sup>
ezpolar <sup>⑤</sup>	极坐标系下的绘图 <sup>⑥</sup>
ezplot3 <sup>⑤</sup>	三维图形 <sup>⑥</sup>
ezmesh <sup>⑤</sup>	三维网格图 <sup>⑥</sup>
ezmeshc <sup>⑤</sup>	带等值线的三维网格图 <sup>⑥</sup>
ezcontour <sup>⑤</sup>	等值线图 <sup>⑥</sup>
ezcontourf <sup>⑤</sup>	带填充的等值线图 <sup>⑥</sup>
ezsurf <sup>⑤</sup>	三维彩色曲面图 <sup>⑥</sup>
ezsurfc <sup>⑤</sup>	带等值线的三维彩色曲面图 <sup>⑥</sup>

## 符号函数曲线的绘制

对于一元函数，可以采用函数fplot( )绘制该函数的图形。

对于多元函数，可以采用函数ezplot( )绘制。

此外，还可以采用函数ezpolar( )在极坐标下绘图。

```
% chap8_65.m
```

```
clear all;
x=-2:0.1:4;
figure;
plot(x, humps(x));
title('plot');
figure;
fplot(@humps, [-2 4]);
title('fplot');
```

```
//fplot 会自动缩小自变量的步长
```

```
% chap8_66.m
```

```
clear all;
figure;
fplot(' [1/x, sin(x), cos(x)]', 2*pi*[-1 1 -1 1])
legend('1/x', 'sin(x)', 'cos(x)');
```

```
% chap8_67.m
```

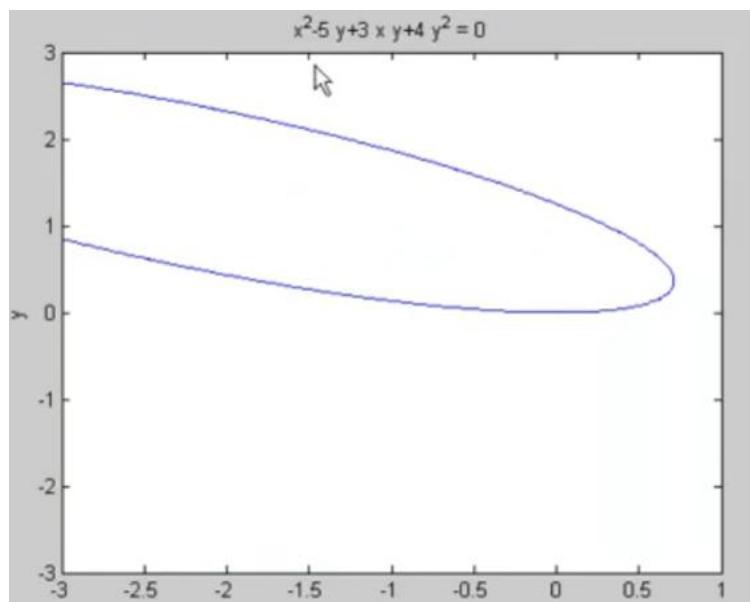
```
clear all;
figure;
fplot('cos(1/x)', [0.01, 0.1], 1e-3);
```

```
% chap8_68.m
```

```
clear all;
figure;
ezplot('x^2-6*x+9*sin(x)+4', [-2*pi, 2*pi]);
```

```
% chap8_69.m
```

```
clear all;
figure;
ezplot('x^2-5*y+3*x*y+4*y^2', [-3, 1, -3, 3]);
```



```
//隐函数
```

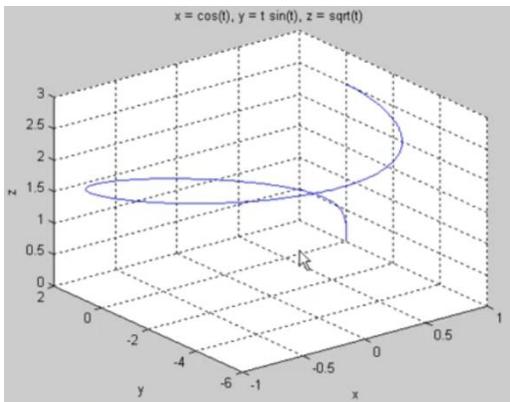
```
% chap8_70.m
```

```
clear all;
figure;
ezplot('sin(4*t)*cos(2*t)', 'cos(4*t)*sin(2*t)')
```

```
% chap8_71.m
clear all;
figure;
ezpolar('sin(2*t).*cos(3*t)', [0, pi]);

% chap8_72.m
clear all;
figure;
ezpolar('1+cos(3*t)', [0, 2*pi]);

% chap8_73.m
clear all;
figure;
ezpolar(@cos, [0, pi]);
```



```
% chap8_74.m
clear all;
figure;
ezplot3('cos(t)', 't*sin(t)', 'sqrt(t)', [0, 2*pi])
```

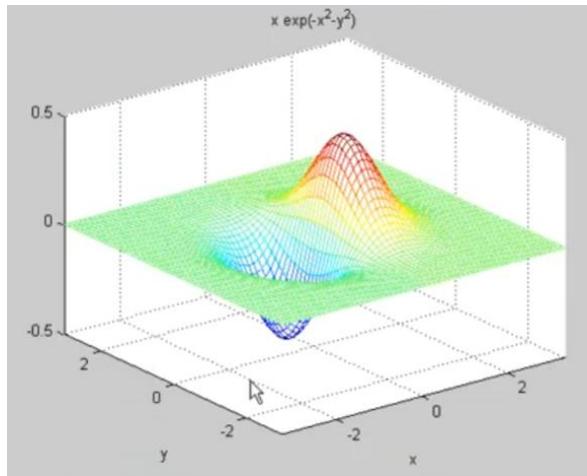
## 符号函数的三维网格图

在MATLAB中，可以采用函数ezmesh( )绘制符号函数的三维网格图，利用函数ezmeshc( )绘制带有等值线的三维网格图。

1. 函数ezmesh( )
2. 函数ezmeshc( )

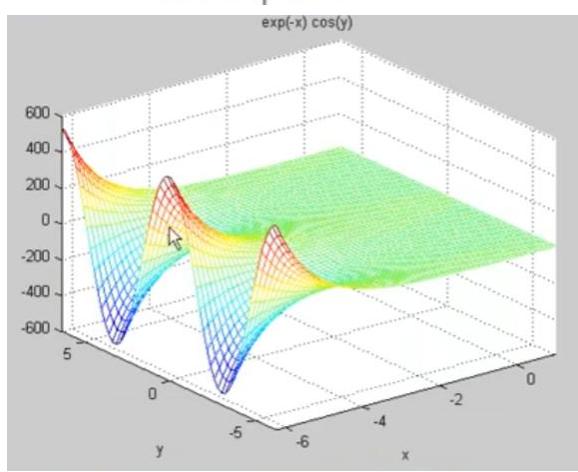
```
% chap8_76.m
```

```
clear all;
figure;
ezmesh('x.*exp(-x.^2-y.^2)');
```



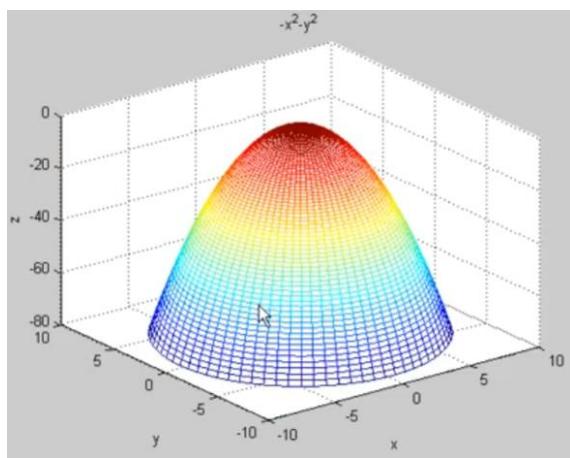
```
% chap8_77.m
```

```
clear all;
figure;
ezmesh('exp(-x)*cos(y)');
```



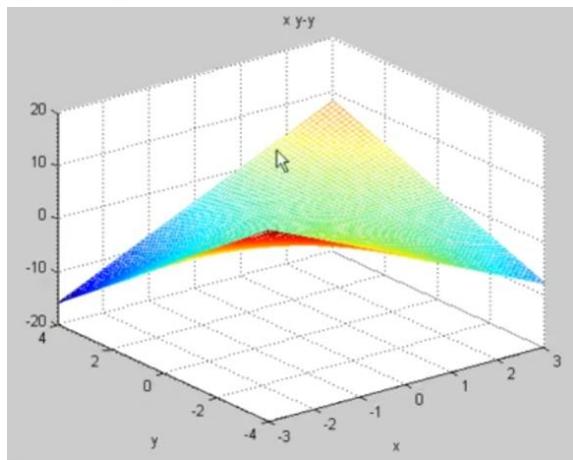
```
% chap8_78.m
```

```
clear all;
figure;
ezmesh('-x.^2-y.^2','circ');
```



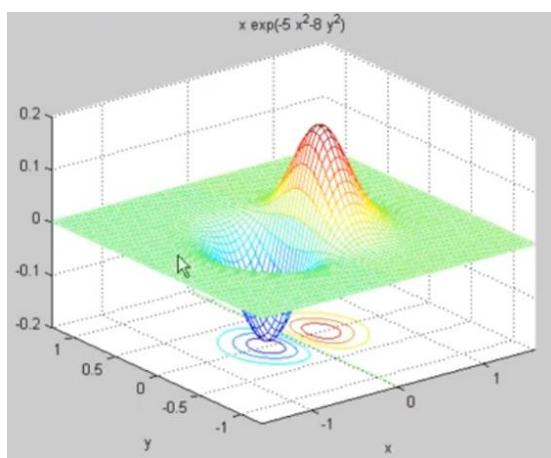
```
% chap8_79.m
```

```
clear all;
figure;
ezmesh('x*y-y', [-3, 3, -4, 4]);
```

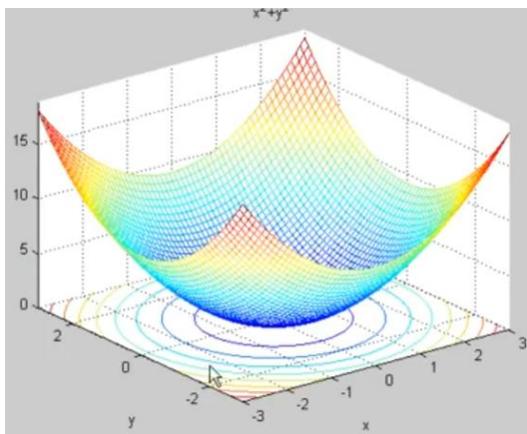


```
% chap8_80.m
```

```
clear all;
figure;
ezmeshc('x.*exp(-5*x.^2-8*y.^2)');
```



```
% chap8_81.m
clear all;
figure;
ezmeshc('x.^2+y.^2', [-3 3 -3 3]);
```



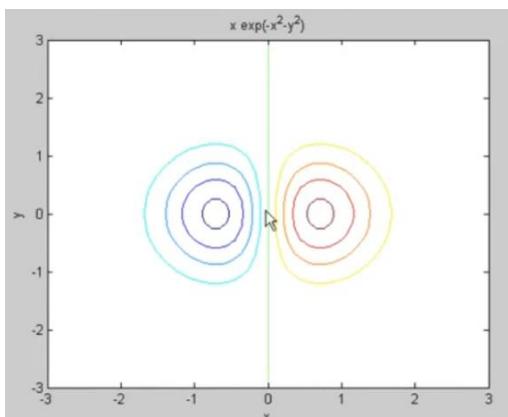
符号函数的等值线图

在MATLAB中，可以采用函数ezcontour( )绘制等值线图，  
利用函数ezcontourf( )绘制经过填充的等值线图。

1. 函数ezcontour( )
2. 函数ezcontourf( )

```
% chap8_82.m
```

```
clear all;
figure;
ezcontour('x.*exp(-x.^2-y.^2)', [-3 3]);
```

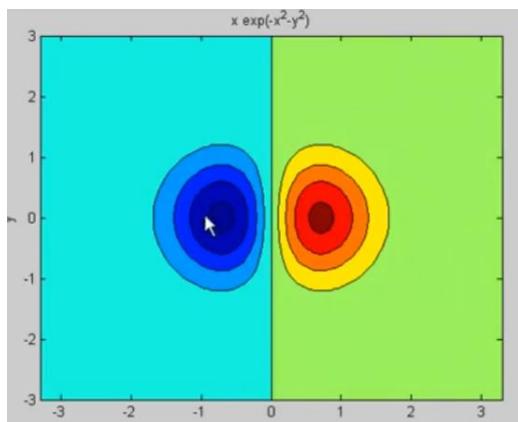


```
% chap8_83.m
```

```
clear all;
figure;
ezcontour('x*sin(y)', [-5 5 -5 5]);
```

```
% chap8_84.m
```

```
clear all;
figure;
ezcontourf('x*exp(-x.^2-y.^2)');
```



```
% chap8_85.m
```

```
clear all;
figure;
ezcontourf('x.^2-y.^2-1', [-4 4 -5 5]);
```

## 符号函数的三维彩色曲面图

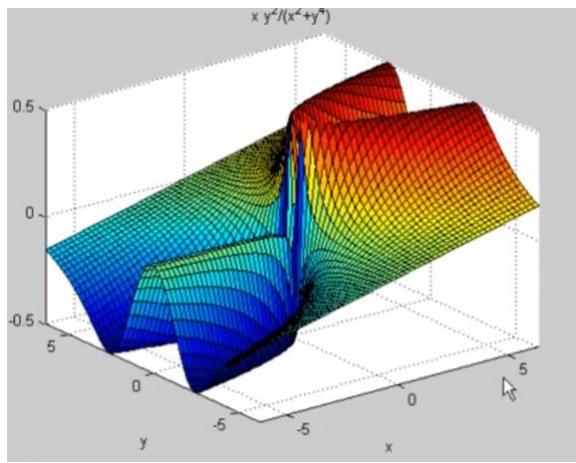
在MATLAB中，采用函数ezsurf( )绘制三维彩色曲面图，利用函数ezsurfc( )绘制带有等值线的三维彩色曲面图。

### 1. 函数ezsurf( )

### 2. 函数ezsurfc( )

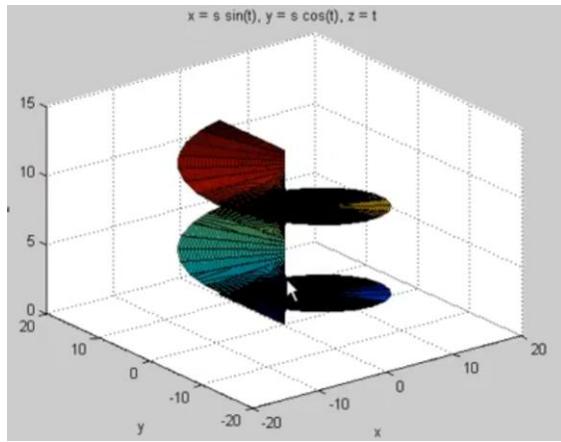
```
% chap8_86.m
```

```
clear all;
figure;
ezsurf('x.*y.^2/(x.^2+y.^4)');
```



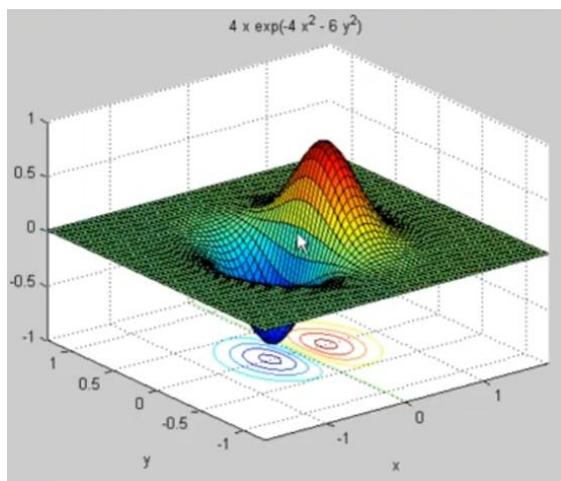
```
% chap8_87.m
```

```
clear all;
figure;
ezsurf('s*sin(t)', 's*cos(t)', 't', [0, 4*pi]);
```



```
% chap8_88.m
```

```
clear all;
figure;
ezsurfc('4*x.*exp(-4*x.^2 - 6*y.^2)');
```



```
% chap8_89.m

clear all;
figure;
ezsurf('x.*y.^2/(x.^2+y.^4)');
```

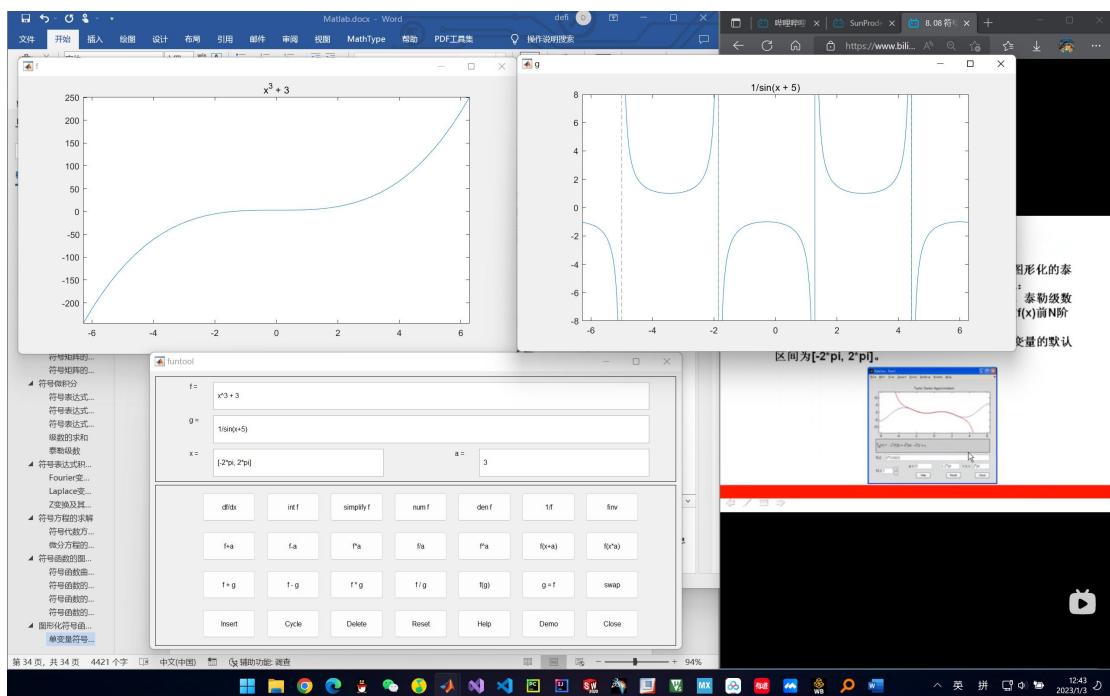
## 图形化符号函数计算器

在MATLAB中，提供了图形化的符号函数计算器。图形化的符号函数计算器功能虽然不是十分的强大，但是操作非常简单和方便，用户可以对符号运算和函数图形有个直观的了解。在MATLAB中，有两种符号函数计算器，单变量符号函数计算器和泰勒级数逼近计算器，下面分别进行介绍。

### 单变量符号函数计算器

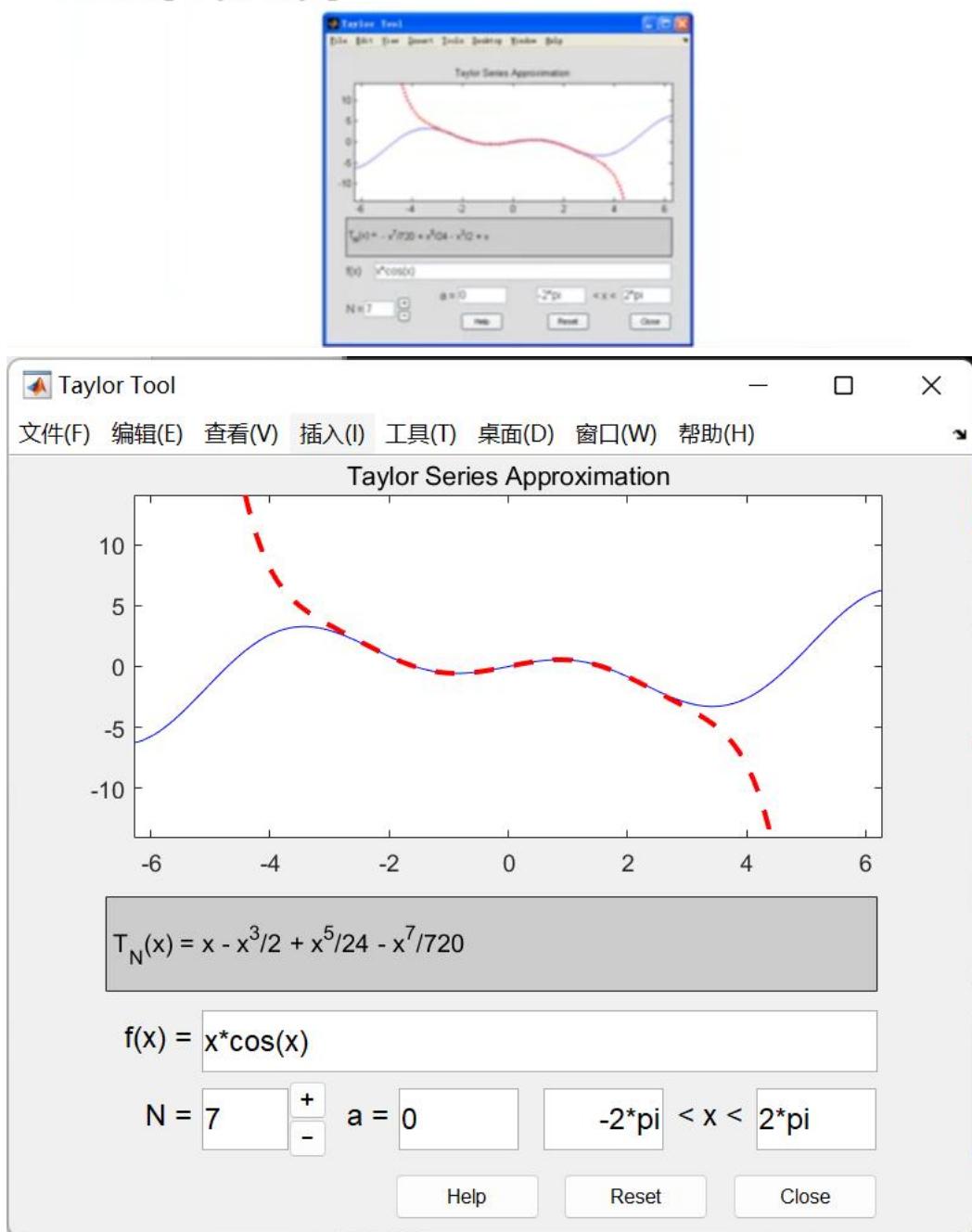
在MATLAB中，可以使用函数`funtool`来调用图形化的单变量符号函数计算器。

用户在命令行窗口中输入：`funtool`，可将单变量符号函数计算器调出，包括3个图形窗口。



## 泰勒级数逼近计算器

在MATLAB中，可以使用函数taylortool来调用图形化的泰勒级数逼近计算器。用户在命令行窗口中输入：taylortool，即可将泰勒级数逼近计算器调出。泰勒级数逼近计算器是一个交互的界面，可以绘制函数 $f(x)$ 前N阶的泰勒级数。在图8.29中，系统默认的函数为 $f(x)=x*\cos(x)$ ，泰勒级数展开的阶数为7，自变量的默认区间为 $[-2\pi, 2\pi]$ 。



## Maple 接口

为了在MATLAB中进一步利用Maple的符号计算能力，MATLAB的符号数学工具箱提供了两个函数：函数sym( )和函数maple( )。这两个函数能够实现对Maple中绝大多数符号计算函数的调用，利用它们可以扩充MATLAB的符号计算能力，下面分别进行介绍。

### 利用 sym 函数调用 Maple 函数

在MATLAB中，可以采用函数sym( )来调用Maple函数，来扩充MATLAB的符号计算功能。然后，通过函数subs( )进行变量的替换。

```
% chap8_90.m
```

```
clear all;
syms k n;
f=sym('k!');
y1=subs(f, k, 9)
y2=subs(f, k, 20)
y3=subs(f, k, n)
```

### 利用 maple 函数调用 Maple 函数

在MATLAB中，可以采用函数maple( )调用Maple中的函数。

在MATLAB中可以通过函数mhelp来获得Maple函数的帮助信息。例如，在命令行输入：mhelp gcd，将获取Maple中的函数gcd的幫助信息。函数gcd( )用于求两个多项式或整数的最大公约数。

```
% chap8_91.m
```

```
clear all;
syms x y a b;
y1=maple('gcd', 6, 9)
y2=maple('gcd', x^2-y^2, x^3-y^3)
y3=gcd(6, 9)
y4=gcd(x^2-y^2, x^3-y^3)
```

## 本章小结

本章详细的介绍了**MATLAB**的符号运算功能，重点讲解了符号表达式、符号矩阵操作、符号微积分，以及符号积分变换和方程求解。此外，介绍了**MATLAB**中符号函数的绘图功能，图形化符号函数计算器，以及**Maple**接口等。  
**MATLAB**中的符号运算功能强大，需要读者多联系，熟练掌握。

# 二维数据可视化

MATLAB受到控制界广泛接受的一个重要原因是因为它提供了方便的绘图功能。Matlab提供了一系列的绘图函数，用户不需要过多的考虑绘图的细节，只需要给出一些基本参数就能得到所需图形，这类函数称为高层绘图函数。二维图形的绘制是其它绘图操作的基础，本章主要介绍二维图形的绘制，主要包括二维绘图、图形的标注、特殊图形的绘制，以及交互式绘图等。

## MATLAB 绘图

二维图形是将平面坐标上的数据点连接起来的平面图形。可以采用不同的坐标系，如直角坐标、对数坐标、极坐标等。在Matlab中，最基本而且应用最为广泛的绘图函数为**plot**，利用它可以在二维平面上绘制出不同的曲线。下面介绍利用MATLAB绘制二维图形。

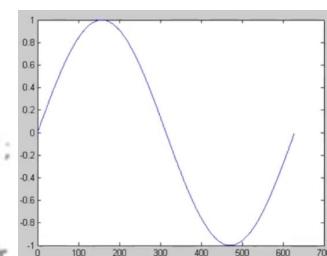
### 基本绘图函数

在MATLAB中，绘制二维图形最常用的函数是**plot( )**，该函数的功能非常强大，通过不同的输入，能够绘制不同的图形。

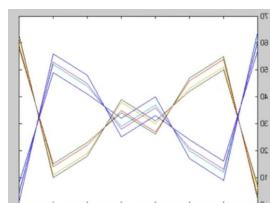
该函数的调用格式为：

1. 函数**plot(y)**
2. 函数**plot(x, y)**
3. 函数**plot(x, y, s)**
4. 函数**plot(x1, y1, s1, x2, y2, s2, ...)**

```
% chap9_1.m  
  
clear all;  
t=0.01:0.01:2*pi;  
y=sin(t);  
figure;  
plot(y);
```

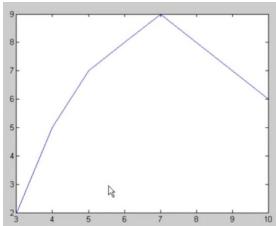


```
% chap9_2.m  
  
clear all;  
y=magic(8);  
figure;  
plot(y);
```



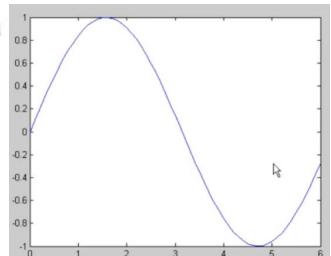
```
% chap9_3.m
```

```
clear all;
y=[3+2i, 4+5i, 5+7i, 6+8i, 7+9i, 10+6i];
figure;
plot(y);
```



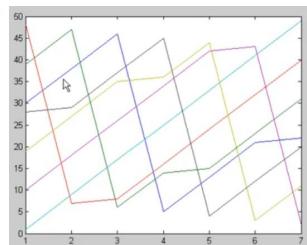
```
% chap9_4.m
```

```
clear all;
x=0:0.1:6;
y=sin(x);
figure;
plot(x, y);
```



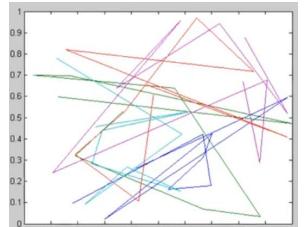
```
% chap9_5.m
```

```
clear all;
x=1:7;
y=magic(7);
figure;
plot(x, y);
```



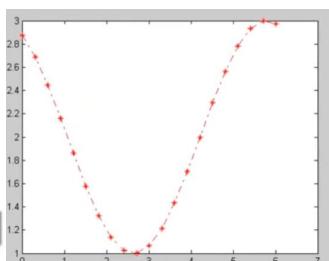
```
% chap9_6.m
```

```
clear all;
x=rand(9, 5);
y=rand(9, 5);
figure;
plot(x, y);
```



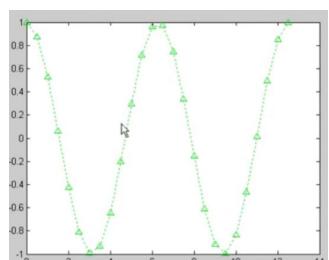
```
% chap9_7.m
```

```
clear all;
x=0.01:0.3:2*pi;
y=cos(x+0.5)+2;
figure;
plot(x, y, 'r-.*');
```



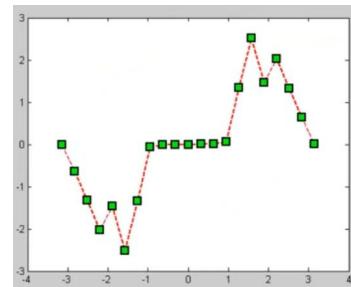
```
% chap9_8.m
```

```
clear all;
x=0.01:0.5:4*pi;
y=cos(x);
figure;
plot(x, y, 'g:^');
```



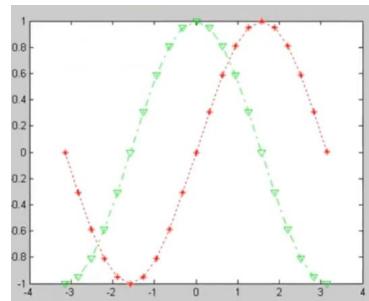
```
% chap9_9.m
```

```
clear all;
x=-pi:pi/10:pi;
y=tan(sin(x))-sin(tan(x));
plot(x, y, '--rs', 'LineWidth', 2, ...
    'MarkerEdgeColor', 'k', ...
    'MarkerFaceColor', 'g', ...
    'MarkerSize', 10);
```



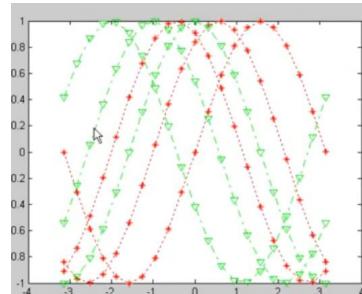
```
% chap9_11.m
```

```
clear all;
x=-pi:pi/10:pi;
y=sin(x);
z=cos(x);
figure;
plot(x, y, 'r-*', x, z, 'g-.v');
```



```
% chap9_12.m
```

```
clear all;
x=-pi:pi/10:pi;
y=[sin(x);sin(x+1);sin(x+2)];
z=[cos(x);cos(x+1);cos(x+2)];
figure;
plot(x, y, 'r-*', x, z, 'g-.v');
```



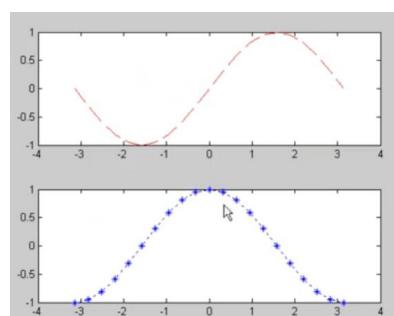
## 子图绘制

在MATLAB中，采用函数**subplot( )**进行子图绘制。该函数能够将几个图形在同一个图形窗口中绘制出来，且采用不同的坐标系。

**subplot(m, n, p):** 该函数将当前图形窗口分成 $m \times n$ 个绘图区，即共m行，每行n个，子绘图区的编号按行优先从左到右编号。该函数选定第p个子图为当前活动区。在每一个子绘图区允许以不同的坐标系单独绘制图形。

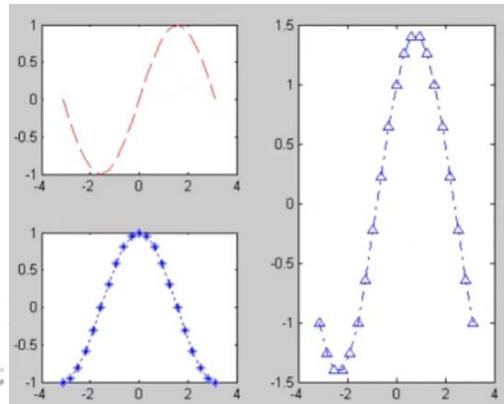
```
% chap9_13.m
```

```
clear all;
x=-pi:pi/10:pi;
figure;
subplot(2, 1, 1);
plot(x, sin(x), 'r--');
subplot(212);
plot(x, cos(x), 'b-*');
```



```
% chap9_14.m
```

```
clear all;
x=-pi:pi/10:pi;
figure;
subplot(2,2,1);
plot(x, sin(x), 'r--');
subplot(2,2,2);
plot(x, cos(x), 'b-*');
subplot(2,2,[2 4]);
plot(x, sin(x)+cos(x), 'b-.~');
```



//神奇！还能这样!!! (学到了 dog)

## 叠加图绘制

在实际应用中，有时需要在已经绘制好的图形上叠加绘制新的图形。在MATLAB中，采用函数**hold**启动或关闭图形保持功能，该函数的调用格式为：

**hold on**: 启动图形保持功能，允许在同一坐标轴上绘制多个图。

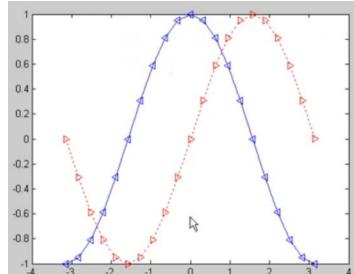
**hold off**: 关闭图形保持功能，不能在当前坐标轴上再绘制图形。

**hold**: 在两种状态**hold on**和**hold off**之间进行切换。

**hold all**: 实现**hold on**功能，并且使新的绘图函数依然按顺序循环使用当前坐标系中**ColorOrder**和**LineStyleOrder**两个属性。

```
% chap9_15.m
```

```
clear all;
x=-pi:pi/10:pi;
figure;
plot(x, sin(x), 'r:>');
hold on;
plot(x, cos(x), 'b-<');
```



## 设置坐标轴

在MATLAB中，可以设置函数axis的参数实现对坐标轴进行控制，该函数的调用格式为：

**axis([xmin xmax ymin ymax]):** 该函数用于定义x轴和y轴的范围。

**axis([xmin xmax ymin ymax zmin zmax]):** 该函数用于定义坐标轴的范围，包含x轴、y轴和z轴的三维图形。

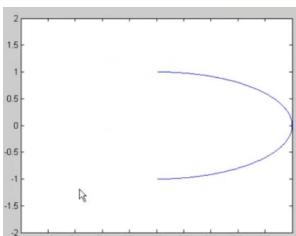
**axis([xmin xmax ymin ymax zmin zmax cmin cmax]):** 该函数用于定义三维坐标轴的范围，以及图形的颜色信息。

```
% chap9_16.m
```

```
clear all;
t=0.01:0.01:pi;
figure;           ans =
plot(sin(t),cos(t));
axis[          0    1    -1    1
```

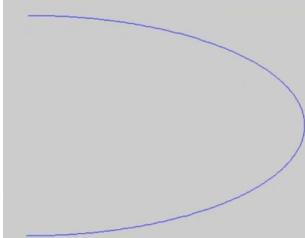
```
% chap9_17.m
```

```
clear all;
t=0.01:0.01:pi;
figure;
plot(sin(t),cos(t));
axis([-1 1 -2 2]);
```



```
% chap9_18.m
```

```
clear all;
t=0.01:0.01:pi;
figure;
plot(sin(t),cos(t));
axis off
```



## 网格线和边框

在MATLAB中，通过函数grid( )给坐标轴加网格线，该函数的调用格式为：

**grid on:** 给当前坐标轴添加网格线。

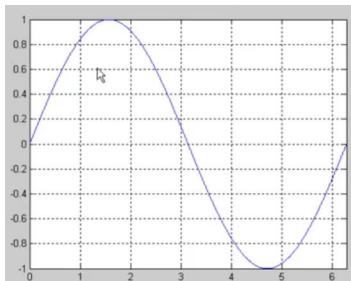
**grid off:** 取消当前坐标轴的网格线。

**grid minor:** 设置网格线间的间距。

**grid:** 不带参数时，在grid on和grid off之间进行切换。

```
% chap9_19.m
```

```
clear all;
t=0.01:0.01:2*pi;
figure;
plot(t,sin(t));
axis([0 2*pi -1 1]);
grid on;
```



在MATLAB中，通过函数**box( )**给坐标轴加边框，该函数的调用格式为：

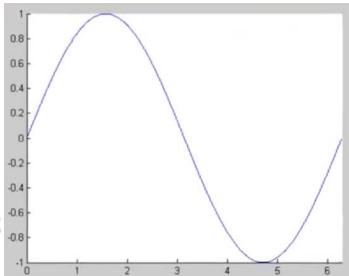
**box on**: 该函数给当前坐标轴添加边框线。

**box off**: 该函数取消当前坐标轴的边框线。

**box**: 不带参数时，在**box on**和**box off**之间进行切换。

```
% chap9_20.m
```

```
clear all;
t=0.01:0.01:2*pi;
figure;
plot(t,sin(t));
axis([0 2*pi -1 1]);
box off; |
```



## 坐标轴的缩放

在MATLAB中，通过函数**zoom**实现图形的缩放，该函数的调用格式为：

**zoom (factor)**: 该函数以作为缩放因子进行坐标轴的缩放。

**zoom on**: 允许对坐标轴进行缩放。

**zoom off**: 禁止对坐标轴进行缩放。

**zoom out**: 恢复到最初的坐标轴设置。

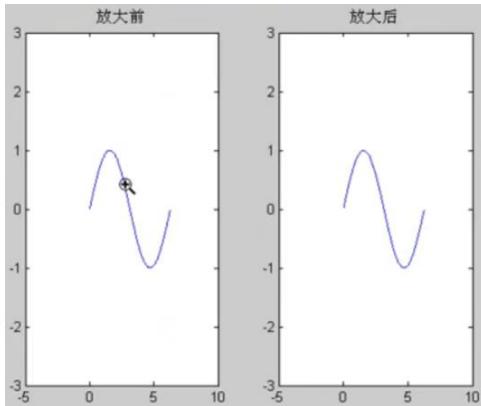
**zoom reset**: 对当前的坐标轴进行复位，恢复到初始值。

**zoom xon**: 允许对轴进行缩放。

**zoom yon**: 允许对轴进行缩放。

**zoom**: 在**zoom on**和**zoom off**之间进行切换。

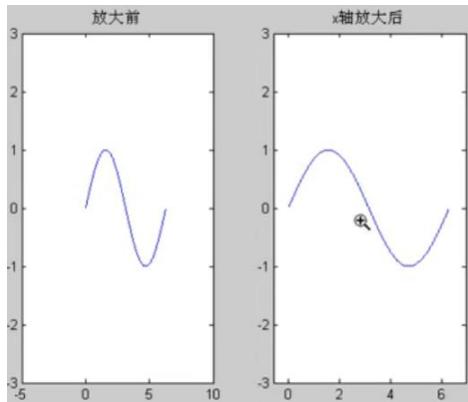
```
clear all;
t=0.01:0.01:2*pi;
figure;
subplot(121);
plot(t,sin(t));
axis([-5 10 -3 3]);
title('放大前');
subplot(122);
plot(t,sin(t));
axis([-5 10 -3 3]);
zoom on;
title('放大后');
```



```

clear all;
t=0.01:0.01:2*pi;
figure;
subplot(121);
plot(t,sin(t));
axis([-5 10 -3 3]);
title('放大前');
subplot(122);
plot(t,sin(t));
axis([-5 10 -3 3]);
zoom xon;
title('x轴放大后');

```



## 图形的拖拽

在MATLAB中，采用函数pan()进行图形的拖拽。在进行图形的拖拽时，鼠标变为手的形状。该函数的调用格式为：

**pan on:** 该函数打开图形的拖拽功能。

**pan xon:** 该函数打开图形在轴方向的拖拽功能。

**pan yon:** 该函数打开图形在轴方向的拖拽功能。

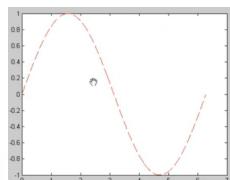
**pan off:** 该函数关闭图形的拖拽功能。

**pan:** 该函数在**pan on**和**pan off**之间进行切换。

```

clear all;
t=0.01:0.01:2*pi;
figure;
plot(t,sin(t), '--r'); %打开拖拽功能
pan on;

```



## 数据光标

在MATLAB中，通过函数datacursormode显示鼠标所选的图形上某点的坐标值，该函数的调用格式为：

**datacursormode on:** 该函数打开图形的数据光标功能。

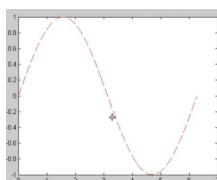
**datacursormode off:** 该函数关闭图形的数据光标功能。

**datacursormode:** 该函数在**datacursormode on**和**datacursormode off**之间进行切换。

```

clear all;
t=0.01:0.01:2*pi;
figure;
plot(t,sin(t), '--r');
datacursormode on;

```

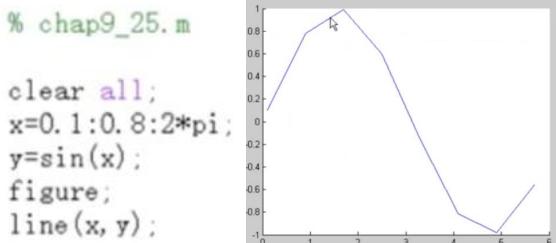


## 绘制直线

在MATLAB中，通过函数line( )绘制简单的线，该函数的调用格式为：

**line(x, y):** 该函数利用参数x和y绘制直线。如果x和y都为向量，必须具有相同的长度，将对应的元素作为横、纵坐标进行画线。如果x和y为矩阵，则以对矩阵的每一列作为坐标绘制多条线。

**line(x, y, z):** 该函数在三维坐标中进行画线，参数x、y和z为三维坐标系中的坐标值。



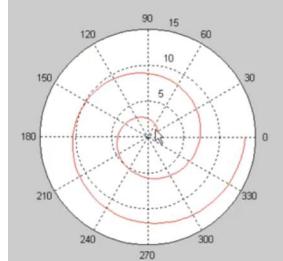
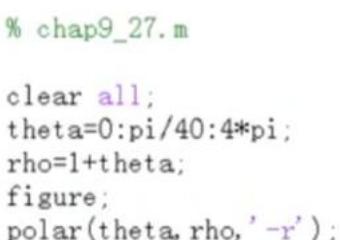
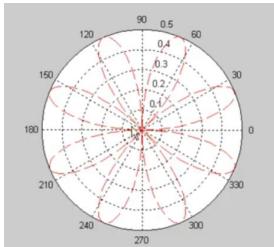
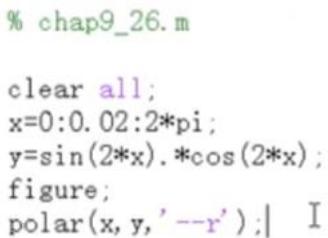
## 极坐标绘图

在MATLAB中，采用函数polar( )进行极坐标系的绘图，该函数的调用格式为：

**polar(theta, rho):** 该函数以弧度为theta和半径为rho在极坐标系中进行绘图，极坐标系下的函数为rho=f(theta)。

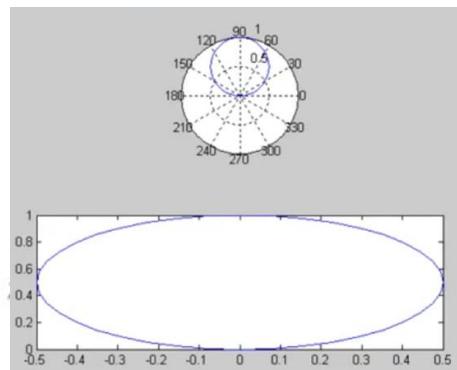
**polar(theta, rho, s):** 该函数通过参数对曲线的线型、标记和颜色等信息进行设置，具体设置信息如表9.1所示。

在MATLAB中，可以通过函数pol2cart( )将极坐标系中的坐标值转换为直角坐标系下的坐标值，函数cart2pol( )将直角坐标系下的坐标值转换为极坐标系下的坐标值。



```
% chap9_28.m
```

```
clear all;
theta=0:pi/40:4*pi;
rho=sin(theta);
figure;
subplot(211)
polar(theta, rho);
[x, y]=pol2cart(theta, rho);
subplot(212);
plot(x, y);
```



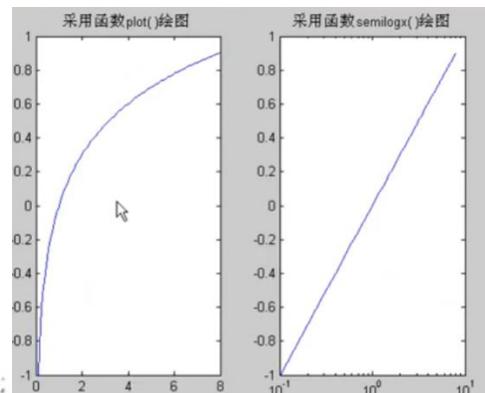
对数和半对数坐标系绘图

在MATLAB中，除了采用等比例刻度坐标系绘图，还可以采用对数和半对数坐标系进行绘图。下面对这些函数进行介绍：

1. 函数semilogx()
2. 函数semilogy()
3. 函数loglog()

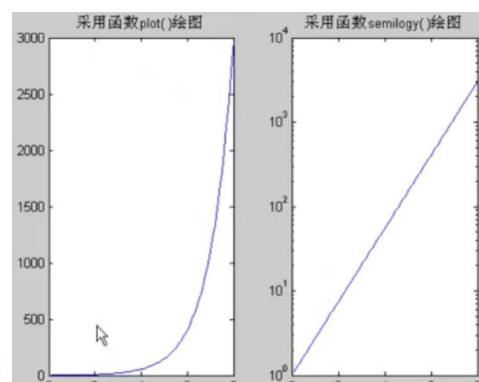
```
% chap9_29.m
```

```
clear all;
x=0.1:0.1:8;
y=log10(x);
figure;
subplot(121);
plot(x, y);
title('采用函数plot()绘图');
subplot(122);
semilogx(x, y);
title('采用函数semilogx()绘图');
```



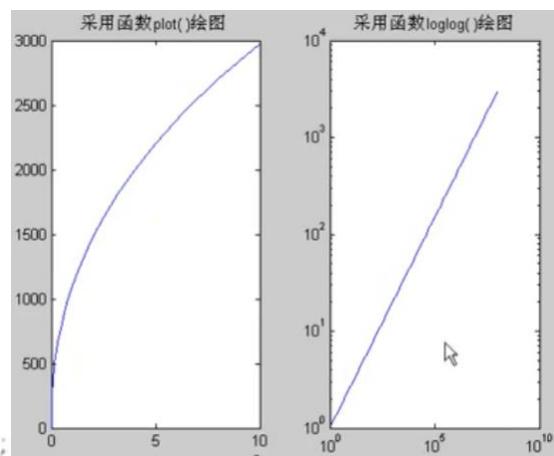
```
% chap9_28.m
```

```
clear all;
x=0.1:0.1:8;
y=exp(x);
figure;
subplot(121);
plot(x, y);
title('采用函数plot()绘图');
subplot(122);
semilogy(x, y);
title('采用函数semilogy()绘图');
```



```
% chap9_31.m
```

```
clear all;
t=0.1:0.1:8;
x=10.^t;
y=exp(t);
figure;
subplot(121);
plot(x,y);
title('采用函数plot( )绘图');
subplot(122);
loglog(x,y);
title('采用函数loglog( )绘图');
```

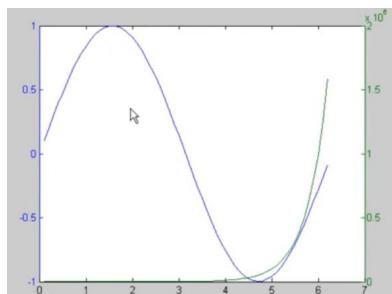


## 双纵轴绘图

在MATLAB中，采用函数`plotyy()`进行双y轴坐标系绘图，横坐标的标度相同。对于两组数据分别采用左侧y轴和右侧y轴，它们的坐标轴范围各自独立，这样就能在一幅图中很好的观察两组数据的变化趋势。

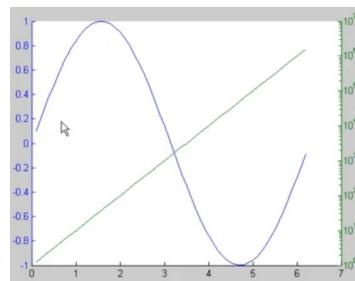
```
% chap9_32.m
```

```
clear all;
x=0.1:0.1:2*pi;
y=sin(x);
z=10.^x;
figure;
plotyy(x,y,x,z);
```



```
% chap9_33.m
```

```
clear all;
x=0.1:0.1:2*pi;
y=sin(x);
z=10.^x;
figure;
plotyy(x,y,x,z,'plot','semilogy');
```



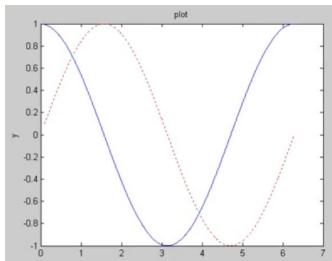
## 图形的窗口和标注

MATLAB所绘制的图像都显示在图形窗口，提供了丰富的绘图工具。MATLAB中图形的标注可以采用绘图工具，也可以采用函数。下面分别进行介绍。

## 图形窗口

在MATLAB中，图形窗口由标题栏、菜单栏、工具条和图形区组成。在标题栏的左侧显示该图形的名称，标题栏的右侧是图形的最大化、最小化和关闭按钮。菜单栏包括：**File**（文件）、**Edit**（编辑）、**View**（视图）、**Insert**（插入）、**Tools**（工具）、**Desktop**（桌面）、**Window**（窗口）和**Help**（帮助）。在菜单栏右侧的箭头可以将图形窗口显示在MATLAB的桌面中。

```
>> clear all;
t=0.1:0.01:2*pi;
figure;
plot(t,sin(t),'r:');
hold pn;
plot(t,cos(t));
xlabel('x');
ylabel('y');
title('plot');
```



通过绘图工具绘制的图形，可以转换为MATLAB程序。通过单击菜单**File|Generate Code**，就可以产生MATLAB程序代码。

//这个可以的!!

## 图形标注概述

在MATLAB中，可以通过函数进行图形的标注，也可以直接在图形的编辑模式下进行图形的标注。图形编辑工具条默认情况下是不显示的，通过单击图形窗口中菜单**View|Plot Edit Toolbar**显示图形编辑工具条。通过图形编辑工具条可以在图形中插入文字、线条和箭头等，还可以对标注进行对齐。



## 图形标题

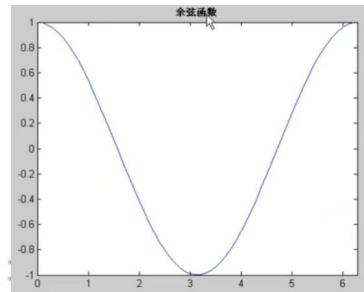
在MATLAB中，可以通过函数title( )给图形添加标题，也可以绘图后通过单击菜单Insert|Title来插入标题。图形的标题位于图形的顶端，默认是居中对齐的。函数title( )的调用格式为：

title('string'): 设置当前图形的标题为字符串。

title(..., 'PropertyName', 'PropertyValue', ...): 该函数对标题的属性进行设置，例如采用的字体、字体的大小和颜色等。

```
% chap9_34.m
```

```
clear all;
x=0.1:0.1:2*pi;
y=cos(x);
figure;
plot(x, y);
axis([0, 2*pi, -1, 1])
title('余弦函数', 'fontname', '宋体', 'fontsize',
```



在进行文字标注时，可以对文字进行控制，常用的设置有：

\bf: 采用的字体加粗。

\it: 采用的字体为斜体。

\rm: 采用标准形式。

\fontname {fontname}: 设置文字的字体。

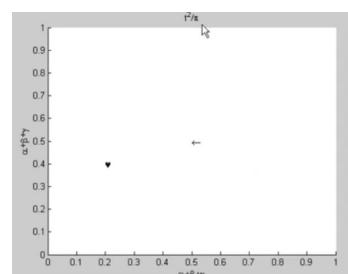
\fontsize {fontsize}: 设置字体的大小。

\color {colorname}: 对字体的颜色进行设置

此外，还可以采用函数texlabel( )将MATLAB的表达式转化为Tex格式的字符串，非常方便的在图形中添加希腊字母或数学公式。

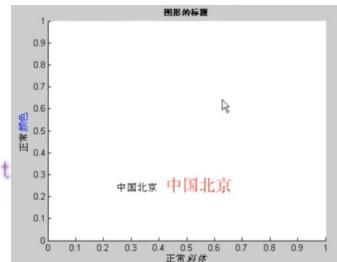
```
% chap9_35.m
```

```
clear all;
title(texlabel('t^2/pi'));
text(0.5, 0.5, '\leftarrow')
text(0.2, 0.4, '\heartsuit');
xlabel('\alpha+\beta+\gamma');
ylabel(texlabel('alpha+beta+gamma'));
```



```
% chap9_36.m

clear all;
title(' \bf\fontname{黑体} 图形的标题');
text(0.25, 0.25, '中国北京' ,\fontname{宋体}\font
xlabel(' 正常\it斜体');
ylabel(' \rm正常\color{blue}颜色');
```



## 坐标轴标题

在MATLAB中，可以通过函数 xlabel( )和 ylabel( )给图形添加坐标轴标题，也可以绘图后通过单击菜单 Insert|XLabel 来插入横轴的标题，单击菜单 Insert|YLabel 来插入纵轴的标题。横轴的标题默认位于横轴的下方中间位置，纵轴的标题默认位于纵轴左方并垂直居中。

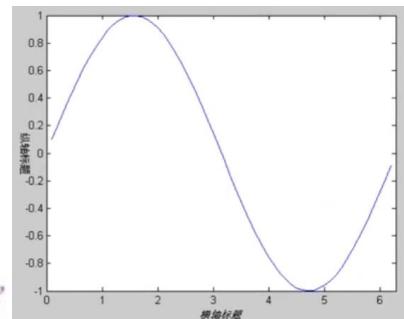
**xlabel('string')**: 设置图形横轴的标题为字符串。

**xlabel(..., 'PropertyName','PropertyValue', ...)**: 该函数对横轴标题的属性进行设置，例如采用的字体、颜色等

函数 ylabel( )的调用格式和 xlabel( )完全一样。

```
% chap9_37.m

clear all;
x=0.1:0.1:2*pi;
y=sin(x);
figure;
plot(x, y);
axis([0, 2*pi, -1, 1])
xlabel(' \it横轴标题', 'fontname', '宋体');
ylabel(' 纵轴标题', 'fontname', '宋体', 'rotation'
```



## 图例

图例（Legend）用来标注图形中不同颜色、线型的数据的实际意义，可以绘图后通过单击菜单Insert|Legend插入图例，图例的各项文字初始值为data1、data2等。也可以通过函数legend( )对图例进行设置，非常的方便。

**legend('string1', 'string2', ...)**或**legend 'string1'**

‘string2’：添加图例，顺序给出各个图形的描述。

**legend('off')**或**legend off**：清除图例。

**legend('toggle')**或**legend toggle**：在显示和清除间切换。

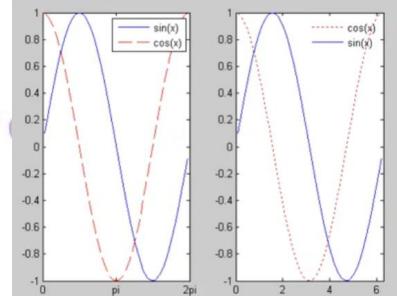
**legend('hide')**或**legend hide**：隐藏图例。

**legend('show')**或**legend show**：显示图例

**legend('boxoff')**或**legend boxoff**：隐藏图例的边框。

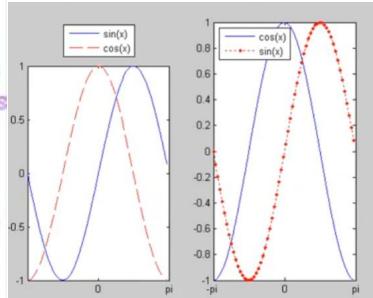
**legend('boxon')**或**legend boxon**：显示图例的边框。

```
clear all;
x=0.1:0.1:2*pi;
y=sin(x);
z=cos(x);
figure;
subplot(121);
plot(x, y, '-b');
hold on;
plot(x, z, '--r');           I
axis([0, 2*pi, -1, 1])
set(gca, 'XTick', [0 pi 2*pi], 'XTickLabel', {'0', 'pi', '2pi'});
legend('sin(x)', 'cos(x)');
subplot(122);
plot(x, z, ':r');
hold on;
plot(x, y, '-b');
axis([0, 2*pi, -1, 1])
legend('cos(x)', 'sin(x)');
legend boxoff;
```



```
% chap9_39.m
```

```
clear all;
x=-pi:0.1:pi;
figure;
subplot(121);
plot(x,sin(x),'-b',x,cos(x),'--r');
axis([-pi,pi,-1,1])
set(gca,'XTick',[0 pi 2*pi],'XTickLabel',{'0','pi','2*pi'});
legend('sin(x)','cos(x)','Location','NorthOutside');
subplot(122);
plot(x,cos(x),'-b',x,sin(x),':r');
axis([-pi,pi,-1,1])
set(gca,'XTick',[-pi 0 pi],'XTickLabel',{'-pi','0','pi'});
legend('cos(x)','sin(x)',2);
```



//可以拖动图例哦

## 颜色条

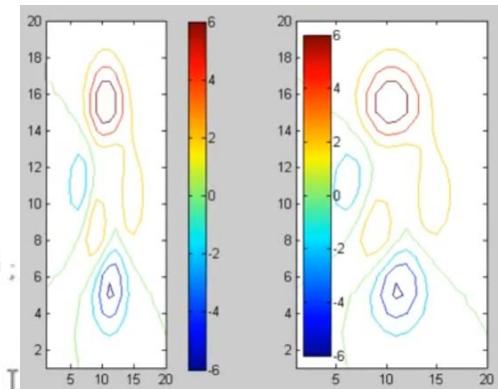
颜色条用于显示图形中颜色和数值的对应关系，方便用户理解图形，主要用于三维绘图和二维等高线图形中。用户可以通过单击菜单Insert|Colorbar显示颜色条，也可以通过函数colorbar( )来添加颜色条。

**colorbar:** 该函数为图形添加颜色条，默认在图形外侧的右部。

**colorbar('Location'):** 该函数对颜色条的位置设置为字符串**Location**。

```
% chap9_40.m
```

```
clear all;
z=peaks(20);
figure;
subplot(121);
contour(z)
colorbar('EastOutside');
subplot(122);
contour(z);
colorbar('West');
```



## 文本框标注

在MATLAB中，可以采用文本框对图形进行标注。添加文本框可以通过绘图后单击Insert|Textbox来添加，也可以通过函数text( )和gtext( )来添加。

函数text( )添加文本框时，需要对文本框放置的位置进行设置，调用格式为：

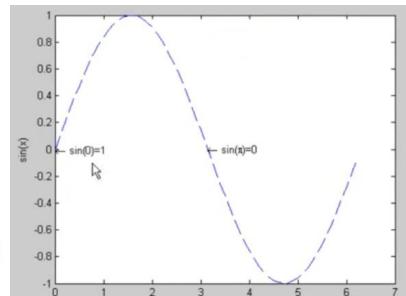
**text(x, y, 'string')**: 该函数在二维平面的坐标(x, y)处添加文本框标注，文本的内容为字符串。

**text(x, y, z, 'string')**: 该函数在三维平面的坐标(x, y, z)处添加文本框标注，文本的内容为字符串。

**text(..., 'PropertyName', 'PropertyValue')**: 该函数在添加文本框进行标注的同时，对文本的格式进行设置。

```
% chap9_41.m
```

```
clear all;
x=0:0.1:2*pi;
y=sin(x);
figure;
plot(x, y, '--b');
xlabel('x');
ylabel('sin(x)');
text(pi, sin(pi), '\leftarrow sin(\pi)=0');
text(0, 0, '\leftarrow sin(0)=1');
```



函数gtext( )是交互式的文本标注工具，可以通过鼠标来选择标注文字放置的位置，该函数的调用格式为：

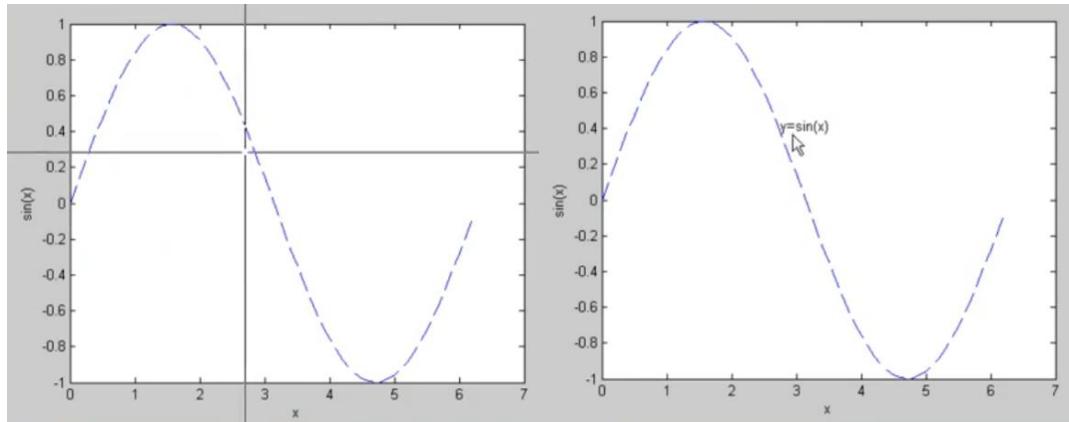
**gtext('string')**: 该函数通过点击鼠标来确定文本框的位置，文本的内容为字符串。

**gtext('string1', 'string2', ...)**: 该函数在鼠标点击的位置标注多行文本和等，字符串之间采用逗号分割。

**gtext('string1'; 'string2'; ...)**: 该函数通过多次鼠标点击添加多个文本框，字符串之间采用分号进行分割。

```
% chap9_42.m
```

```
clear all;
x=0:0.1:2*pi;
y=sin(x);
figure;
plot(x, y, '--b');
xlabel('x');
ylabel('sin(x)');
gtext('y=sin(x)', 'fontsize', 10);
```



## 获取和标记数据点

在MATLAB中，可以通过函数`ginput()`和鼠标方便的读取二维平面上任一点的坐标值，该函数的调用格式为：

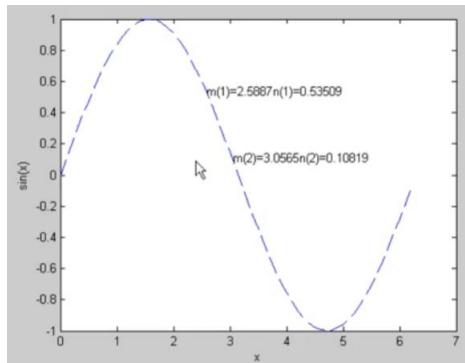
`[x, y]=ginput(n)`: 该函数通过鼠标选择个点，它们的坐标值保存在变量和中，可以通过Enter键结束。

`[x, y]=ginput`: 该函数通过鼠标选择多个点，坐标值保存在 和 中，通过Enter键结束。

`[x, y, button]=ginput(...)`: 该函数的返回值记录了在用鼠标选点时的相关信息。

```
% chap9_43.m
```

```
clear all;
x=0:0.1:2*pi;
y=sin(x);
figure;
plot(x, y, '--b');
xlabel('x');
ylabel('sin(x)');
[m, n]=ginput(2)
text(m(1), n(1), ['m(1)=', num2str(m(1)), ' n(1)=', num2str(n(1))]);
text(m(2), n(2), ['m(2)=', num2str(m(2)), ' n(2)=', num2str(n(2))]);
```



## 特殊图形的绘制

在MATLAB中，除了可以通过函数`plot()`等绘制图形外，还有一些函数可以绘制特殊的图形，例如柱状图、饼状图和直方图等。

表 9.6 二维特殊图形常用函数<sup>①</sup>

函数 <sup>②</sup>	意义 <sup>③</sup>	函数 <sup>②</sup>	意义 <sup>③</sup>
<code>bar()</code>	柱状图 <sup>④</sup>	<code>fill()</code>	多边形填充图 <sup>⑤</sup>
<code>barh()</code>	水平柱状图 <sup>④</sup>	<code>feather()</code>	羽毛图 <sup>⑥</sup>
<code>area()</code>	填充图 <sup>⑦</sup>	<code>comet()</code>	彗星图 <sup>⑧</sup>
<code>pie()</code>	饼状图 <sup>⑨</sup>	<code>quiver()</code>	向量场图 <sup>⑩</sup>
<code>errobar()</code>	图形加上误差范围 <sup>⑪</sup>	<code>compass()</code>	罗盘图 <sup>⑫</sup>
<code>hist()</code>	直方图 <sup>⑬</sup>	<code>rose()</code>	极坐标下的柱状图 <sup>⑭</sup>
<code>stairs()</code>	阶梯图 <sup>⑮</sup>	<code>pareto()</code>	Pareto图 <sup>⑯</sup>
<code>stem()</code>	火柴杆图 <sup>⑰</sup>	<code>convhull()</code>	凸壳图 <sup>⑱</sup>
<code>contour()</code>	等高线绘图 <sup>⑲</sup>	<code>polyarea()</code>	数组确定的多边形填充图 <sup>⑳</sup>

## 函数绘图

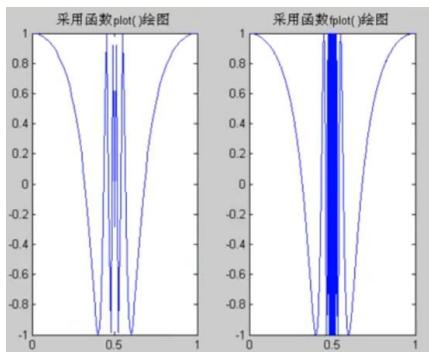
如果只知道某个函数的表达式，也可以绘制将该函数的图形。函数`fplot()`用于绘制一元函数的图形，函数`ezplot()`用于绘制二元函数的图形，函数`ezpolar()`用于绘制极坐标图形。

下面分别进行介绍。

1. 函数`fplot()`
2. 函数`ezplot()`
3. 函数`ezpolar()`
4. 函数`ezcontour()`和`ezcontourf()`

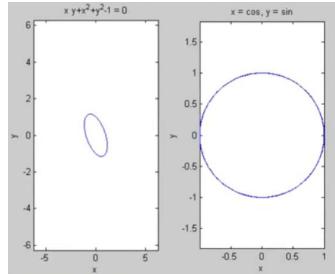
% chap9\_44.m

```
clear all;
x=0:0.01:1;
y=cos(tan(pi*x));
figure;
subplot(121);
plot(x, y, '-b');
title('采用函数plot()绘图');
subplot(122);
fplot('cos(tan(pi*x))', [0 1], 1e-4);
title('采用函数fplot()绘图');
```



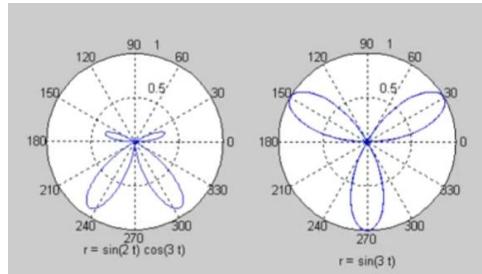
```
% chap9_45.m
```

```
clear all;
figure;
subplot(121);
ezplot('x.*y+x.^2+y.^2-1');
subplot(122);
ezplot(@cos, @sin, [-2*pi 2*pi]);
```



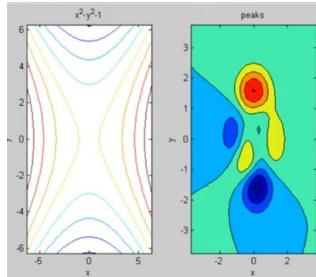
```
% chap9_46.m
```

```
clear all;
figure;
subplot(121);
ezpolar('sin(2*t).*cos(3*t)', [0 pi]);
subplot(122);
ezpolar(@(t) sin(3*t), [0 2*pi]);
```



```
% chap9_47.m
```

```
clear all;
figure;
subplot(121);
ezcontour('x^2-y^2-1');
subplot(122);
ezcontourf(@peaks);
```



## 柱状图

在MATLAB中，通过函数**bar( )**绘制柱状图，该函数的调用格式为：

**bar(y)**: 该函数以y中的值为长度画一个长柱。

**bar(x, y)**: 该函数在指定的横坐标x上画出y，参数x为单调增加的。如果y为矩阵，则将每个行向量画出。

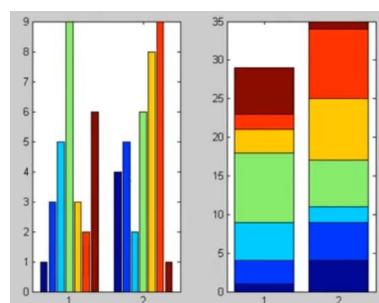
**bar(..., width)**: 该函数设置柱的宽度，默认值为0.8。如果width大于1，则一组内的柱互相重叠。

**bar(..., 'style')**: 该函数的参数style定义形状，取值可以为group或stack，默认值为group。

**bar(..., 'bar\_color')**: 该函数定义柱的颜色为bar\_color。

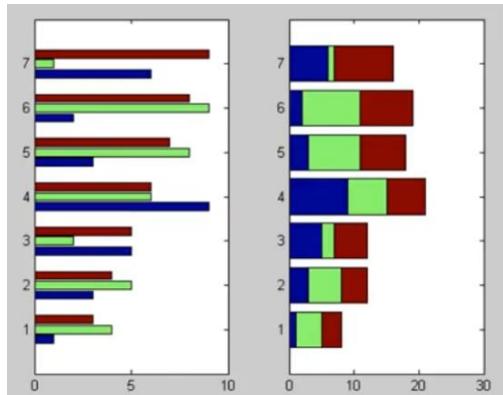
```
% chap9_48.m
```

```
clear all;
figure;
k=[1 3 5 9 3 2 6;4 5 2 6 8 9 1];
subplot(121);
bar(x);
subplot(122);
bar(x, 'stack')
```



```
% chap9_49.m

clear all;
figure;
x=[1 3 5 9 3 2 6;4 5 2 6 8 9 1;3 4 5 6 7 8 9]';
subplot(121);
barh(1:7, x);
subplot(122);
barh(1:7, x, 'stack')
```



## 饼状图

在MATLAB中，通过函数`pie( )`绘制饼状图，该函数的调用格式为：

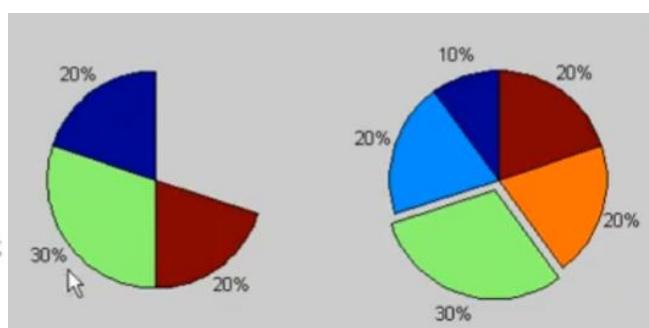
`pie(x)`: 该函数以向量来绘制饼状图。如果`sum(x)<1`，则以向量x的值为各个分量画饼状图。如果`sum(x)>1`，则进行归一化处理。

`pie(x, explode)` : 该函数中参数`explode`与x有相同的维数，其中非零元素对应的向量x的分量在饼状图中向外突出一些。

`pie(..., labels)`: 该函数用于定义每个分量的标签，和向量x有相同的维数。

```
% chap9_50.m
```

```
clear all;
figure;
x=[0.2 0.3 0.2];
subplot(121);
pie(x);
subplot(122);
y=[0.1 0.2 0.3 0.2 0.2];
explode=[0 0 1 0 0];
pie(y,explode);
```



## 直方图

在MATLAB中，通过函数**hist( )**绘制直方图，该函数的调用格式为：

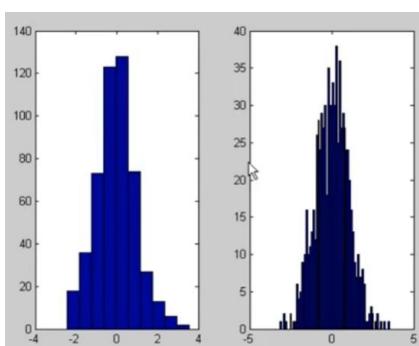
**h=hist(y):** 该函数将向量y中的元素放到10个柱的直方图中，返回值h为包含每个柱的元素个数组成的向量。如果参数y为矩阵，则按照矩阵的列来画图。

**h=hist(y, m):** 参数m为标量，用于指定柱的个数。

**h=hist(y, x):** 该函数中参数x为向量，将参数中的元素放到length(x)个由x中元素指定的位置为中心的直方图。

```
% chap9_51.m
```

```
clear all;
figure;
x=randn(500, 1);
subplot(121);
hist(x);
subplot(122);
y=randn(800, 1);
hist(y, -4:0.1:4);
```



## 面积图

在MATLAB中，通过函数**area( )**绘制面积图，该函数的调用格式为：

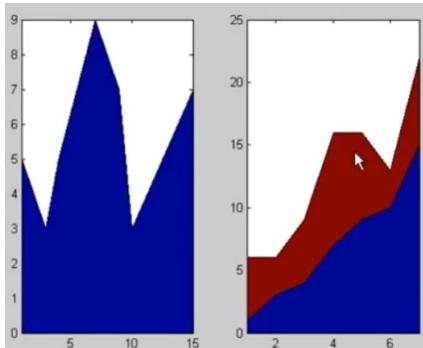
**area(x, y):** 该函数以参数x和y绘制面积图。如果x和y为向量，则相当于函数**plot(x, y)**，并将0到y之间进行了填充。如果参数y为矩阵，则将y的每一列绘制面积图并进行累计求和。

**area(y):** 如果参数y为向量，则绘制面积图；如果y为矩阵则绘制每一列的面积图之和。

**area(..., level):** 该函数绘制到y=level的面积图，参数level的默认值为0。

```
% chap9_52.m
```

```
clear all;
figure;
x=[1 3 4 7 9 10 15]';
y=[5 3 5 9 7 3 7]';
subplot(121);
area(x,y);
subplot(122);
area([x y])
```



## 散点图

在MATLAB中，通过函数scatter( )绘制散点图，该函数的调用格式为：

**scatter(x, y, s, c):** 该函数以x和y分别为横坐标和纵坐标，绘制散点，参数s设置散点的大小，参数c设置散点的颜色。

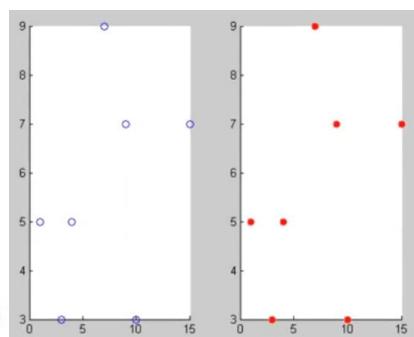
**scatter(x, y):** 该函数以x和y分别为横坐标和纵坐标，绘制散点，采用系统默认的大小和颜色。

**scatter(..., m):** 该函数设置散点为m，而不是默认的“o”。

**scatter(..., 'filled'):** 该函数将空的散点进行填充。

```
% chap9_53.m
```

```
clear all;
figure;
x=[1 3 4 7 9 10 15];
y=[5 3 5 9 7 3 7];
subplot(121);
scatter(x,y);           I
subplot(122);
scatter(x,y,[],[1 0 0], 'fill')
```



在MATLAB中，还可以利用函数plotmatrix( )绘制多个散点图，该函数的调用格式为：

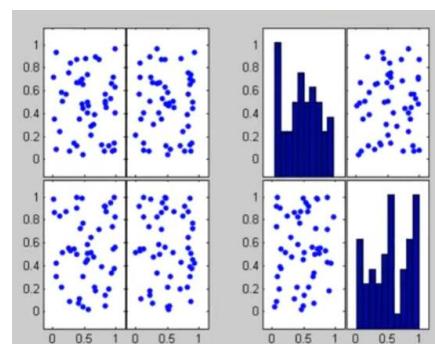
**plotmatrix(X, Y):** 该函数中参数X为 $p \times m$ 的矩阵，参数Y为 $n \times p$ 的矩阵，绘制 $n \times m$ 个散点图，其中第(i,j)个散点图是根据矩阵Y的i第列和矩阵X的第j列的数据绘制的。

**plotmatrix(Y):** 该函数相当于plotmatrix(Y, Y)，参数Y为 $p \times n$ 的矩阵时，绘制出 $n \times n$ 个散点图。该图的对角块画出的是矩阵Y的每列数据的频数直方图。

**plotmatrix(..., 'LineSpec'):** 该函数采用字符串LineSpec设置曲线的线型和颜色等信息。

```
% chap9_54.m
```

```
clear all;
X=rand(50, 2);
Y=rand(50, 2);
figure;
subplot(121);
plotmatrix(X,Y);           %绘制多个散点图
subplot(122);
plotmatrix(Y);             %绘制多个散点图
```

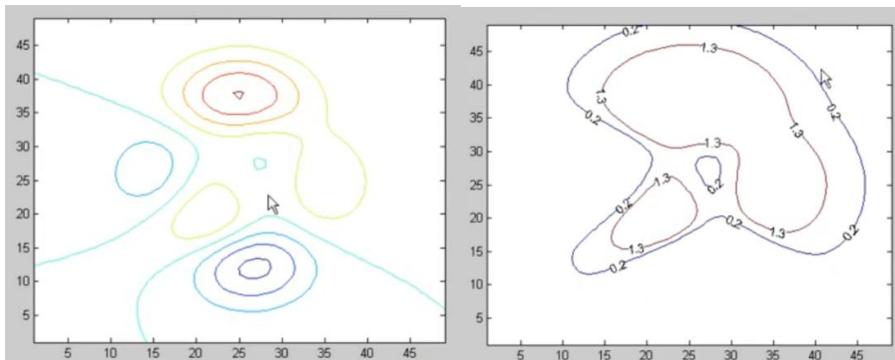


## 等高线绘图

在MATLAB中，通过函数**contour( )**绘制等高线图，函数**contourf( )**绘制经过填充的等高线图。该函数的调用格式为：  
**contour(z)**: 该函数绘制矩阵z的等高线。  
**contour(x, y, z)**: 该函数在坐标(x, y)的值为矩阵z的等高线。  
**contour(z, n)**或**contour(x, y, z, n)**: 该函数绘制n条等高线。  
**contour(x, y, z, [v v])**: 该函数绘制高度为v的等高线。  
**[c, h]=contour(...)**: 该函数返回值c和h，可以采用函数**clabel(c, h)**对等高线进行标注。

```
% chap9_55.m
```

```
clear all;
z=peaks;
figure;
contour(z); %绘制等高线图
figure;
[c, h]=contour(z, [0.2 1.3]); %绘制两条等高线
clabel(c, h); %标记等高线
```

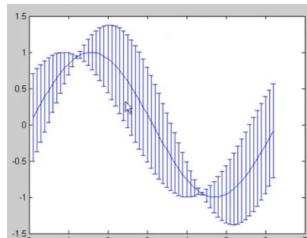


## 误差图

在MATLAB中，通过函数**errorbar( )**绘制误差图，该函数的调用格式为：  
**errorbar(y, e)**: 该函数绘制向量y，以及向量y的误差e。  
**errorbar(x, y, e)**: 该函数中x、y和e为同类型的变量。  
**errorbar(x, y, e, LineSpec)**: 该函数中利用参数**LineSpec**来设定线型和颜色等。

```
% chap9_56.m
```

```
clear all;
x=0.1:0.1:2*pi;
y=sin(x+rand);
figure;
errorbar(x, sin(x), y-sin(x));
```



## 填充图

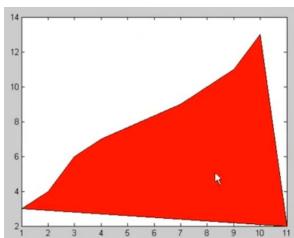
在MATLAB中，通过函数fill( )绘制多边形填充图，该函数的调用格式为：

**fill(x, y, c):** 该函数绘制以向量x和y组成的多边形的填充图，参数c用于设定颜色，可以为表示颜色的字符串，例如：'r'、'g'、'b'、'y'等，也可以是RGB色彩向量，例如：[1, 0, 0]。如果x和y为相同维数的矩阵，则绘制矩阵每一列的多边形填充图。

**fill(x1, y1, c1, x2, y2, c2, ...):** 该函数绘制多个多边形填充图。

```
% chap9_57.m
```

```
clear all;
x=[1 2 3 4 7 9 10 11];
y=[3 4 6 7 9 11 13 2];
figure;
fill(x, y, 'r');
```



## 火柴杆图

在MATLAB中，通过函数stem( )绘制火柴杆图，该函数的调用格式为：

**stem(y):** 该函数绘制向量y的火柴杆图。如果y为矩阵，则绘制的每一列的火柴杆图。

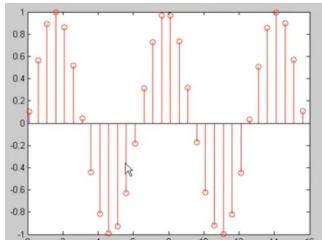
**stem(x, y):** 该函数绘制横坐标为x的火柴杆图。

**stem(..., 'filled'):** 该函数将火柴杆进行填充。

**stem(..., 'LineSpec'):** 该函数用于设置火柴杆图的线型和颜色等。

```
% chap9_58.m
```

```
clear all;
x=0.1:0.5:5*pi;
y=sin(x);
figure;
stem(x, y, 'r');
```



## 阶梯图

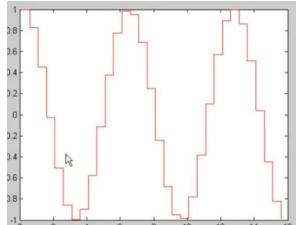
在MATLAB中，通过函数stairs( )绘制阶梯图，该函数的调用格式为：

**stairs(y):** 该函数绘制向量y的阶梯图。

**stairs(x,y):** 该函数绘制横坐标为x的阶梯图。

```
% chap9_59.m
```

```
clear all;
x=0.1:0.5:5*pi;
y=cos(x);
figure;
stairs(x, y, 'r');
```



## 罗盘图

在MATLAB中，通过函数**compass( )**绘制罗盘图，该函数的调用格式为：

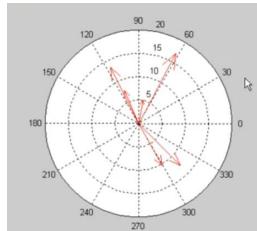
**compass(u, v)**: 该函数绘制罗盘图，从坐标原点指向坐标( $u, v$ )的箭头。

**compass(z)**: 该函数中 $z$ 为复数，相当于**compass(real(z), imag(z))**。

**compass(..., 'LineSpec')**: 该函数利用参数**LineSpec**来设定线型和颜色等。

```
% chap9_60.m
```

```
clear all;
x=[1 -3 5 -6 8 9];
y=[5 7 -9 12 15 -9];
figure;
compass(x, y, 'r'); %绘制罗盘图
```



## 羽毛图

在MATLAB中，通过函数**feather( )**绘制羽毛图，该函数的调用格式为：

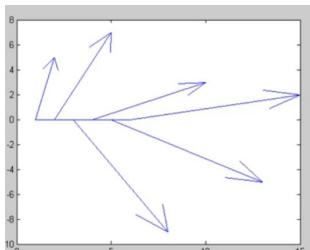
**feather(u, v)**: 该函数绘制向量 $u$ 和 $v$ 的羽毛图。

**feather(z)**: 该函数中 $z$ 为复数，相当于**feather(real(z), imag(z))**。

**feather(..., 'LineSpec')**: 该函数利用参数**LineSpec**来设定线型和颜色等。

```
% chap9_61.m
```

```
clear all;
x=[1 3 5 6 8 9];
y=[5 7 -9 3 -5 2];
figure;
feather(x, y);
```



## 向量场图

在MATLAB中，通过函数quiver( )绘制向量场图，该函数的调用格式为：

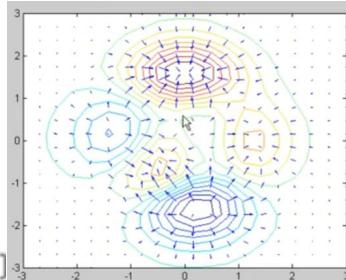
**quiver(x, y, u, v)**: 该函数在坐标(x, y)处绘制向量场图，  
(u, v)为速度分量。

**quiver(u, v)**: 该函数绘制向量场图。

**quiver(..., 'LineSpec')**: 该函数利用参数**LineSpec**来设定线型和颜色等。

**quiver(..., 'LineSpec','filled')**: 该函数设置为填充模式。

```
% chap9_62.m
clear all;
[x, y, z]=peaks(20);
contour(x, y, z, 15);
[u, v]=gradient(z);
hold on;
quiver(x, y, u, v);
```



```
>> help peaks
```

**peaks** – peaks 函数

此 MATLAB 函数 返回在一个  $49 \times 49$  网格上计算的 peaks 函数的 z 坐标。

```
Z = peaks
Z = peaks(n)
Z = peaks(Xm, Ym)
[X, Y, Z] = peaks(__)
```

```
peaks(__)
```

```
>> help gradient
```

**gradient** – 数值梯度

此 MATLAB 函数 返回向量 F 的一维数值梯度。输出 FX 对应于  $\partial F / \partial x$ ，即 x (水平) 方向上的差分。点之间的间距假定为 1。

```
FX = gradient(F)
[FX, FY] = gradient(F)
[FX, FY, FZ, ..., FN] = gradient(F)
[__] = gradient(F, h)
[__] = gradient(F, hx, hy, ..., hn)
```

## 彗星图

在MATLAB中，通过函数comet( )绘制彗星图，该函数的调用格式为：

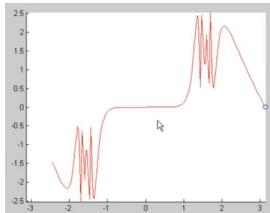
comet(y): 该函数绘制由向量y确定路线的彗星图。

comet(x, y): 该函数绘制由向量x和y确定路线的彗星图。

comet(x, y, p): 该函数设置彗星体的长度为p\*length(y)，参数p的默认值为0.1。

```
% chap9_63.m
```

```
clear all;
t=-pi:pi/100:pi;
y=tan(sin(t))-sin(tan(t));
comet(t, y);
```



//动态

## 伪彩色绘图

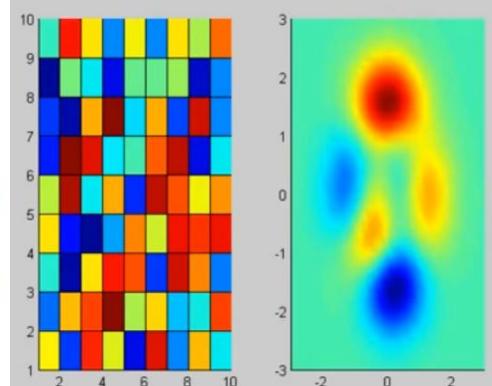
在MATLAB中，通过函数pcolor( )进行伪彩色绘图，该函数的调用格式为：

pcolor(C): 该函数绘制矩阵C的伪彩色图。

pcolor(X, Y, C): 该函数采用参数X确定横坐标，参数Y确定纵坐标，绘制伪彩色图。

```
% chap9_64.m
```

```
close all;
C=rand(10);
figure;
subplot(121);
pcolor(C); %绘制伪彩色图
subplot(122);
[X, Y, Z]=peaks(40);
pcolor(X, Y, Z); %绘制伪彩色图
colormap jet; I
shading interp;
```



## 图形句柄

MATLAB图形句柄是指图形中最基本、最底层的单元。在MATLAB中，对图形对象进行修改和控制时，采用图形对象句柄（Handle）。下面介绍常用的图形句柄和坐标轴句柄。

## 图形句柄

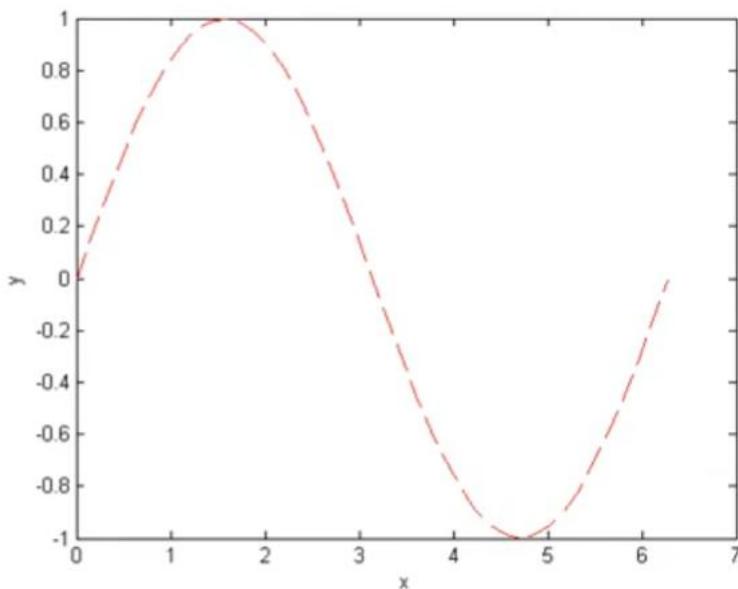
在MATLAB中，利用函数set( )设置图形句柄的属性值，利用函数get( )获取图形句柄的属性值。

函数set( )的调用格式为：set(H, 'PropertyName', PropertyValue)，该函数设置句柄H为的图形对象的属性值，将属性设置为PropertyValue。

函数get( )的调用格式为：get(H, 'PropertyName')，该函数获取句柄H为PropertyName的图形对象的属性的值。

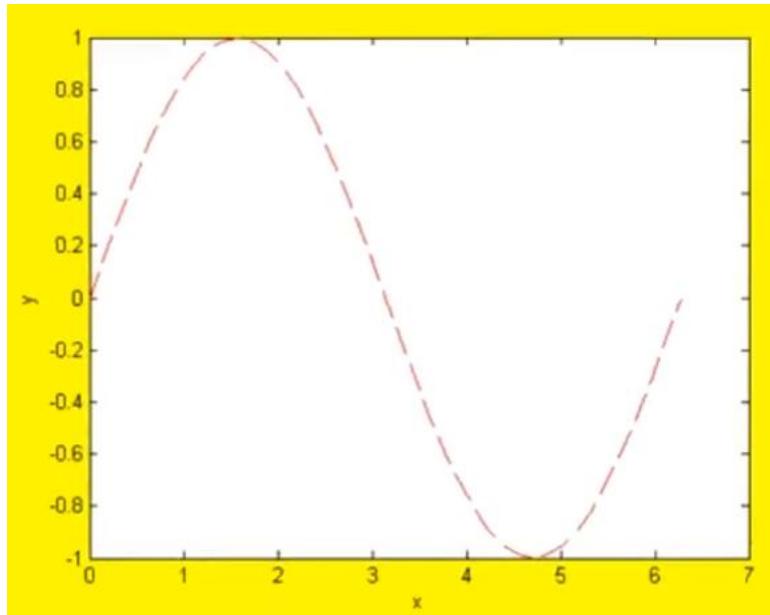
```
% chap9_65.m

close all;
x=0:pi/20:2*pi;
h1=figure %获取图形句柄
plot(x, sin(x), '--r'); %画图
xlabel('x');
ylabel('y');
get(h1, 'MenuBar')
set(h1, 'Name', '图形1');
set(h1, 'Color', [1, 1, 1]); %将图形窗口的底色设置
```



```
% chap9_66.m

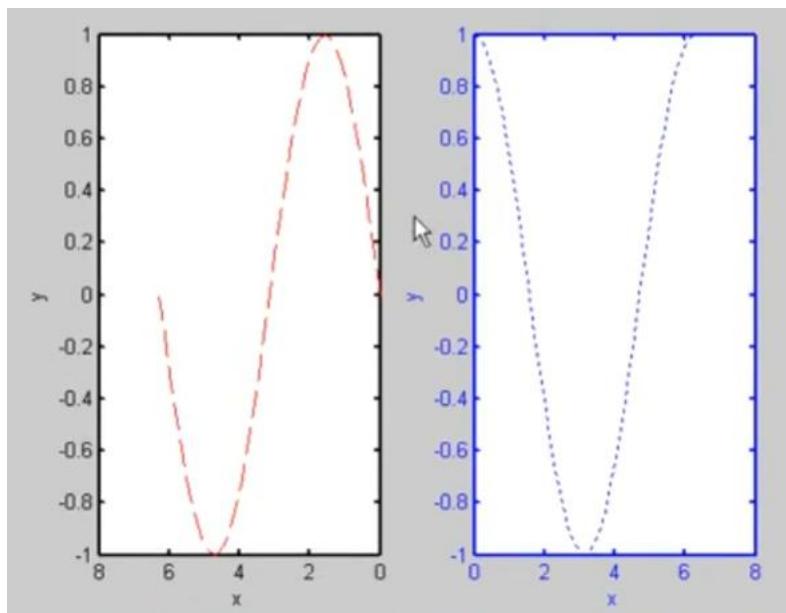
close all;
x=0:pi/20:2*pi;
figure(3);
plot(x,sin(x),'-r'); %绘图
xlabel('x');
ylabel('y');
set(gcf,'Name','图形1'); %通过函数gcf获得
set(gcf,'Color',[1,1,0]); %通过函数gcf获得
```



## 坐标轴句柄

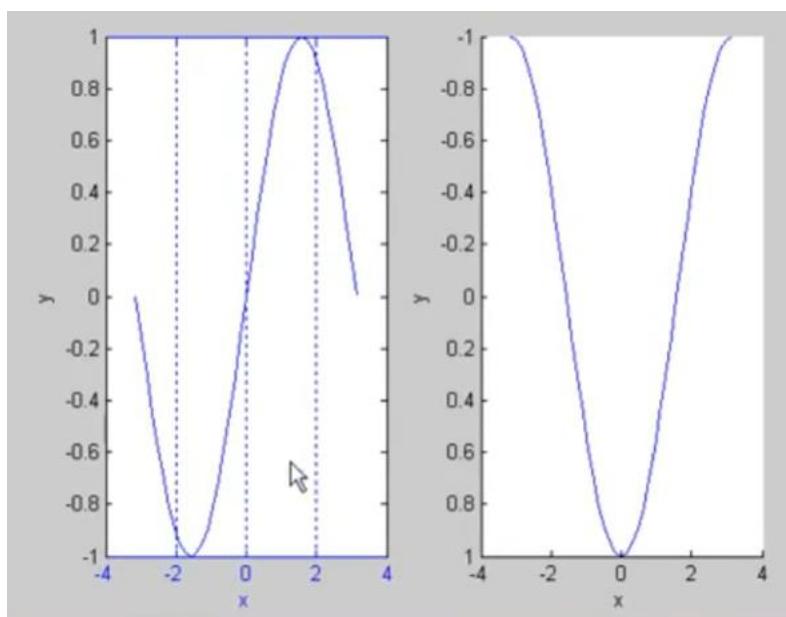
在MATLAB中，通过函数`gca`获得当前图形中坐标轴的句柄。  
通过坐标轴句柄，可以利用函数`get()`获取坐标轴的属性值  
，也可以通过函数`set()`对坐标轴的属性值进行设置。

```
close all;
x=0:pi/20:2*pi;
figure(3);
subplot(121);
plot(x,sin(x),'-r');
xlabel('x');
ylabel('y');
set(gca,'Xdir','reverse'); %设置x轴反向
set(gca,'LineWidth',2); %x轴宽度为2
subplot(122);
plot(x,cos(x),':b');
xlabel('x');
ylabel('y');
set(gca,'Xcolor',[0 0 1]); %x轴颜色为蓝色
set(gca,'Ycolor',[0 0 1]); %y轴颜色为蓝色
set(gca,'LineWidth',1.5);
```



```
% chap9_68.m
```

```
close all;
x=-pi:pi/20:pi;
figure;
subplot(121);
plot(x,sin(x));
set(gca,'Xcolor','b','Xgrid','on');
xlabel('x');
ylabel('y');
subplot(122); I
plot(x,cos(x));
set(gca,'Box','off','Xgrid','off','Ydir','reverse');
xlabel('x');
ylabel('y');
```



## 本章小结

本章详细介绍了**MATLAB**中数据的二维可视化，包括二维绘图函数、图形的标注，以及特殊的二维图形的绘制。

最后，介绍了图形句柄。通过图形句柄，可以实现图形进行更加深入的控制和设置。本章是**MATLAB**绘图中最基础的内容，需要读者仔细的阅读，深入的理解，观察不同图形的差别，最好每个程序都编程和运行。

# 三维数据可视化

在MATLAB中，三维图形包括三维曲线、三维网格图和三维曲面图，分别采用函数plot3()、函数mesh()和surf()进行绘制。本章还将介绍一些特殊的三维图形绘制，重点讲解三维图形的视角、色彩和光照等控制工具，以及图形的打印和输出等。下面分别进行介绍。

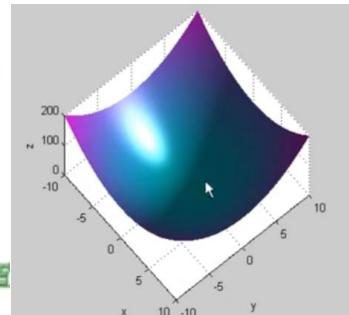
## 创建三维图形

在实际的工程计算中，经常需要将结果表示成三维图形，下面介绍如何利用MATLAB创建三维图形，包括三维曲线图、三维曲面图和特殊三维图形等，以及非网格数据三维绘图等。

### 三维图形概述

MATLAB语言提供了三维绘图功能，这些功能与二维图形的绘制有很多的相似之处，例如曲线的属性设置完全相同。最常用的三维绘图包括三维曲线图、三维网格图和三维曲面图等。完整的三维绘图流程，通常包括绘图的数据，设置图形窗口，图形的视角、颜色和光照等效果，以及坐标轴和图形标注的设置等。

```
clear all;                                %绘图数据
x=-10:0.1:10;
y=-10:0.1:10;
[X, Y]=meshgrid(x, y);
z=X.^2+Y.^2;
figure;                                     %图形窗口
surf(x, y, z);                            %绘图
view([50 70]);                            %视角
colormap('cool');                           %颜色
shading interp;                           %光照
light('Position', [1 0.4 0.4]);          %光照
axis square;                             %坐标轴设置
xlabel('x');                               %图形标注
ylabel('y');
zlabel('z');
```



## 三维曲线图

在MATLAB中，利用函数plot3( )绘制三维曲线图，该函数的调用格式为：

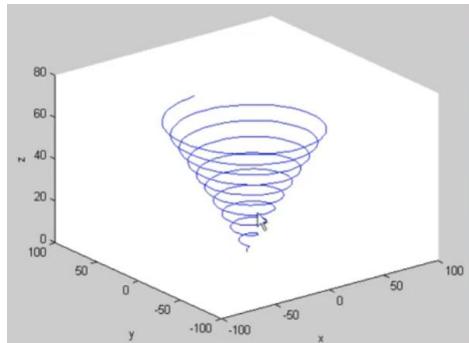
**plot3(x, y, z):** 该函数绘制三维曲线，参数x、y和z是有相同的维数的向量。

**plot3(X, Y, Z):** 该函数中参数X、Y和Z为具有相同维数的矩阵，将参数X、Y和Z的每一列绘制一条曲线，同时绘制多条曲线。

**plot3(X, Y, Z, s):** 该函数和二维绘图函数plot( )类似，可以设置曲线的线型和颜色等信息，以及数据点的标记等。

```
% chap10_2.m
```

```
clear all;
t=linspace(0, 20*pi, 500);
x=t.*sin(t);
y=t.*cos(t);
z=t;
figure.| I
plot3(x, y, z);
xlabel('x');
ylabel('y');
zlabel('z');
```



## 三维曲面图

在MATLAB中，提供了一个peaks( )函数，可产生一个凹凸有致的曲面，包含了三个局部极大点及三个局部极小点。

**peaks:** 该函数绘制三维曲面，默认大小为49\*49。

**peaks(n):** 该函数绘制三维曲面，三维曲面数据为n\*n。

**z=peaks:** 产生默认大小为49\*49的矩阵，不绘制三维曲面。

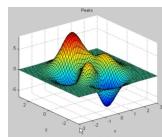
**z=peaks(x, y):** 该函数计算输入参数为x和y的方程的值，不绘制三维曲面。

**[x, y, z]=peaks:** 该函数产生3个矩阵x、y和z，大小都为49\*49，不绘制三维曲面。

**[x, y, z]=peaks(n):** 该函数产生3个矩阵x、y和z，大小都为n\*n，不绘制三维曲面。

```
% chap10_3.m
```

```
clear all;
[a, b, c]=peaks(20);
peaks
```



```
z = 3*(1-x).^2.*exp(-(x.^2)-(y+1).^2) ...
- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
- 1/3*exp(-(x+1).^2 - y.^2)
```

在MATLAB中，采用函数**meshgrid()**产生平面区域内的网格坐标矩阵。

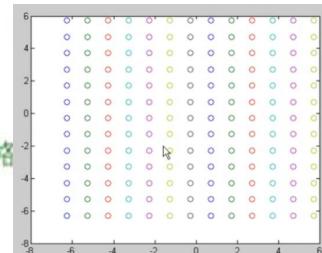
在MATLAB中，采用函数**mesh()**绘制三维网格图，该函数将曲面上的点用线条连接起来，网格对应的曲面区域则显示为空白。该函数的调用格式为：**mesh(X, Y, Z, c)**：该函数绘制彩色的三维网格图，其中矩阵X和Y控制x轴和y轴，矩阵Z为z轴数据，图形的颜色采用参数c设置。

在MATLAB中，采用函数**meshc()**绘制带有等高线的三维网格图，采用函数**meshz()**绘制带有底座的三维网格图。这两个函数的用法与函数**mesh()**类似。

此外，还可以采用函数**waterfall()**绘制三维网格图。

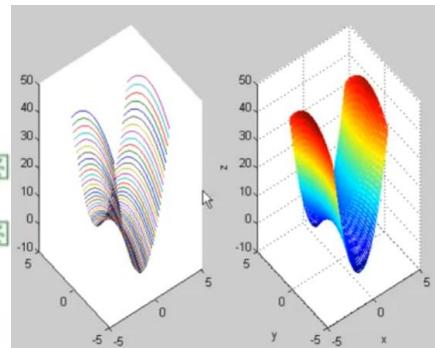
```
% chap10_4.m
```

```
clear all;
x=-2*pi:2*pi;
y=-2*pi:2*pi;
[X, Y]=meshgrid(x, y); %产生矩形网格
figure;
plot(X, Y, 'o'); %绘制网格数据
```



```
clear all;
[X, Y]=meshgrid(-3:0.1:3);
Z=5*X.^2-Y.^2+3;
figure;
subplot(121);
plot3(X, Y, Z); %绘制三维曲线图
subplot(122);
mesh(X, Y, Z); %绘制三维网格图
xlabel('x');
ylabel('y');
zlabel('z');
```

%绘制三维网格图

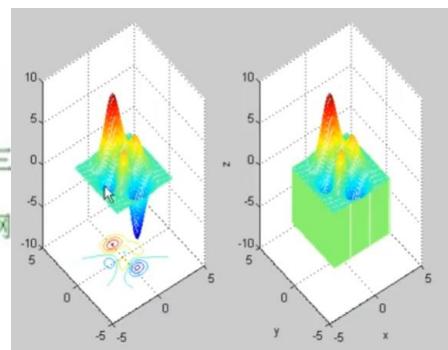


```

clear all;
[X, Y]=meshgrid(-3:0.1:3);
Z=peaks(X, Y);
figure;
subplot(121);
meshc(X, Y, Z); %绘制带有等高线的三维网
subplot(122);
meshz(X, Y, Z);
xlabel('x');
ylabel('y');
zlabel('z');

```

%绘制带有等高线的三  
%绘制有底座的三维网

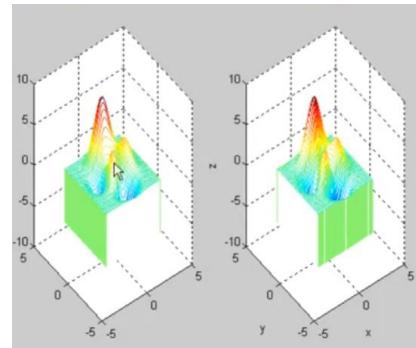


```

clear all;
[X, Y]=meshgrid(-3:0.1:3);
Z=peaks(X, Y);
figure;
subplot(121);
waterfall(X, Y, Z); %x方向流水
subplot(122);
waterfall(X', Y', Z'); %y方向流水
xlabel('x');
ylabel('y');
zlabel('z');

```

%x方向流水  
%y方向流水



在MATLAB中，采用函数surf( )绘制三维曲面图。该函数的调用格式为：surf(X, Y, Z, c): 该函数绘制彩色的三维曲面图，其中矩阵X和Y控制x轴和y轴，矩阵Z为z轴数据，图形的颜色采用参数c设置。

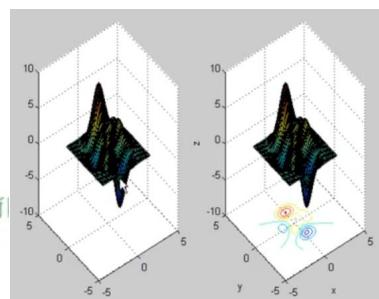
在MATLAB中采用函数surfc( )绘制带有等高线的三维曲面图，通过函数surfl( )添加三维曲面的光照效果，调用方式和surf( )相同。

```

clear all;
[X, Y]=meshgrid(-3:0.1:3);
Z=peaks(X, Y);
figure;
subplot(121);
surf(X, Y, Z); %绘制三维曲面图
subplot(122);
surfc(X, Y, Z); %绘制带等高线的三维曲面
xlabel('x');
ylabel('y');
zlabel('z');

```

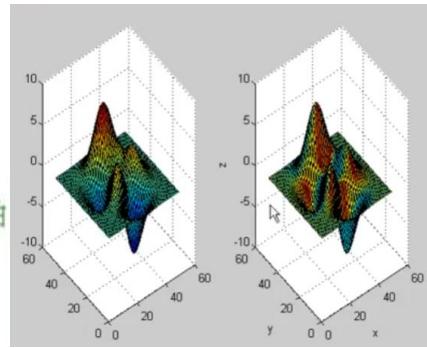
%绘制带等高线的三维曲面



```

clear all;
z=peaks;
figure;
subplot(121);
surf(z); %绘制三维曲面图
subplot(122);
surfl(z); %绘制有光照效果的三维曲面图
xlabel('x');
ylabel('y');
zlabel('z');

```



## 特殊三维绘图

### 1. 柱面图

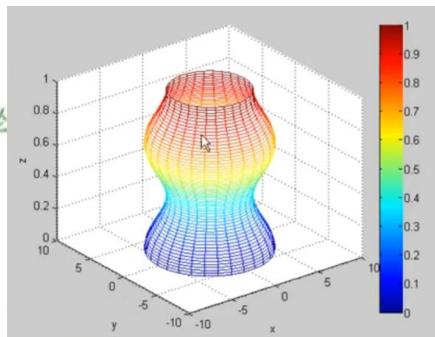
在MATLAB中，可以通过函数cylinder()产生柱面图的数据，然后采用函数mesh()方便的绘制出三维旋转体。

```
% chap10_10.m
```

```

clear all;
t=0:pi/20:3*pi;
r=5+cos(t);
[x, y, z]=cylinder(r, 30); %获取柱面坐标
figure;
mesh(x, y, z); %绘图
xlabel('x');
ylabel('y');
zlabel('z');
colorbar;

```

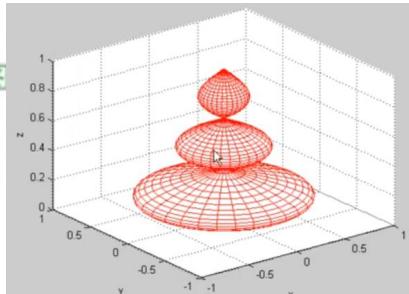


```
% chap10_11.m
```

```

clear all;
t=0:pi/20:3*pi;
r=abs(exp(-0.2*t).*sin(t));
[x, y, z]=cylinder(r, 30); %产生柱面图
figure;
mesh(x, y, z); %采用函数mesh()
xlabel('x');
ylabel('y');
zlabel('z');
colormap([1 0 0]);

```



### 2. 球面图

在MATLAB中，可以通过函数sphere()绘制球面图。该函数的调用格式为：

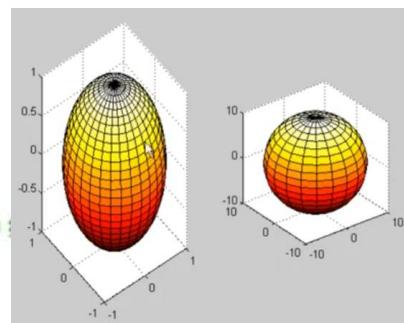
[X, Y, Z]=sphere(N): 该函数产生3个(N+1)\*(N+1)的矩阵，利用函数surf(X, Y, Z)可产生单位球面。

[X, Y, Z]=sphere: 该函数产生3个的矩阵，默认值。

sphere(N): 该函数只绘制球面图而不返回任何值。

```
% chap10_12.m

clear all;
figure;
subplot(121);
sphere(30); %绘制球面图
subplot(122);
[x, y, z]=sphere;
surf(10*x, 10*y, 10*z); %利用函数surf();
axis('equal');
colormap('hot');
```

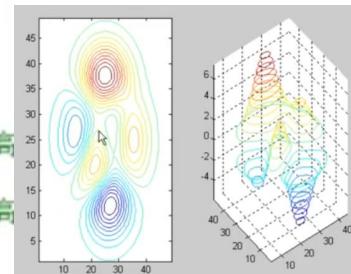


### 3. 三维等高线图

在MATLAB中，可以通过函数`contour3()`绘制三维等高线图。该函数的调用格式和绘制二维等高线的函数`contour()`非常相近，用户可以在命令行窗口输入：`help contour3`查询该函数的详细使用方法。

```
% chap10_13.m
```

```
clear all;
Z=peaks;
figure;
subplot(121);
contour(Z, 20); %绘制二维等高线
subplot(122);
contour3(Z, 20); %绘制三维等高线
axis([-inf inf -inf inf -inf inf]);
```

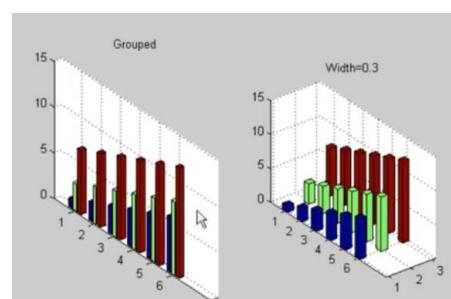


### 4. 三维柱状图

在MATLAB中，采用函数`bar3()`绘制三维柱状图，该函数的调用格式和函数`bar()`非常相近。用户可以在命令行窗口输入：`help bar3`来查看该函数的详细介绍。此外，还可以采用`bar3h()`绘制水平三维柱状图，该函数的调用方式和函数`bar3()`相同。

```
% chap10_14.m
```

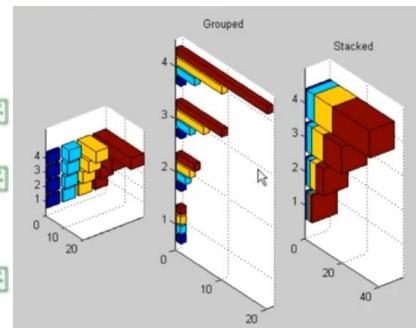
```
clear all;
x=[1:6,3:8;7:12];
figure;
subplot(121);
bar3(x, 'grouped'); %绘制三维柱状图
title('Grouped');
subplot(122);
bar3(x, 0.3); %绘制三维柱状图
title('Width=0.3');
```



```

clear all;
x=pascal(4);
figure;
subplot(131); %绘制水平三维柱状图
bar3h(x);
subplot(132); %绘制水平三维柱状图
bar3h(x, 'grouped');
title('Grouped');
subplot(133); %绘制水平三维柱状图
bar3h(x, 'stacked');
title('Stacked');

```



## 5. 三维饼状图

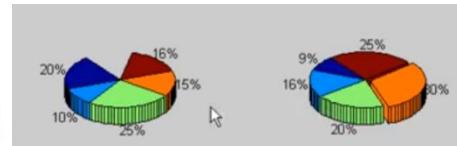
在MATLAB中，采用函数`pie3()`绘制三维饼状图，该函数的调用格式和二维饼状图绘图函数`pie()`基本相同。如果用户需要查看函数`pie3()`的详细使用情况，可以在命令行窗口输入：`help pie3`。

```

% chap10_16.m

clear all;
x=[0.2 0.1 0.25 0.15 0.16];
y=[3 5.5 7 10.5 8.5];
figure;
subplot(121); %绘制三维饼图
pie3(x)
subplot(122); %绘制三维饼图
explode=[0 0 0 1 0];
pie3(y, explode);

```



## 6. 三维散点图

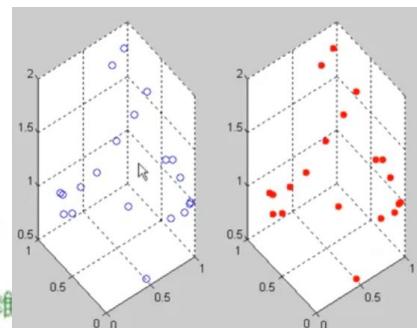
在MATLAB中，采用函数`scatter3()`绘制三维散点图，该函数的调用格式和二维散点图函数`scatter()`非常相近。如果用户需要查看函数`scatter3()`的详细使用情况，可以在命令行窗口输入：`help scatter3`。

```

% chap10_17.m

clear all;
x=rand(1, 20);
y=rand(1, 20);
z=x+y;
figure;
subplot(121); %绘制三维散点图
scatter3(x, y, z)
subplot(122); %绘制三维散点图
scatter3(x, y, z, 'r', 'filled');

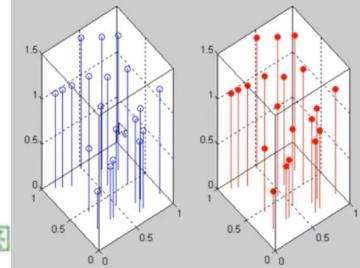
```



## 7. 三维火柴杆图

在MATLAB中，采用函数`stem3( )`绘制三维火柴杆图，该函数的调用格式和二维火柴杆绘图函数`stem( )`非常相近。  
用户可以在MATLAB的命令行窗口输入：`help stem3`查看该函数的详细使用情况。

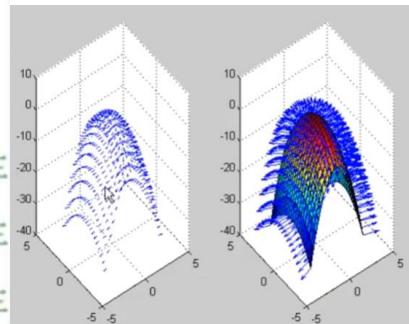
```
% chap10_18.m  
  
clear all;  
x=rand(1, 20);  
y=rand(1, 20);  
z=x+y;  
figure;  
subplot(121); %绘制三维火柴杆图  
stem3(x, y, z)  
subplot(122);  
stem3(x, y, z, 'r', 'filled'); %绘制三维火柴杆图
```



## 8. 三维向量场图

在MATLAB中，采用函数`quiver3( )`绘制三维向量场图，该函数的调用格式和二维向量场图绘制函数`quiver( )`非常相近。用户可以在命令行窗口输入：`help quiver3`，查看该函数的详细使用情况。

```
clear all;  
[X, Y]=meshgrid(-3:0.3:3);  
Z=-3*X.^2-Y.^2;  
[U, V, W]=surfnorm(X, Y, Z);  
figure;  
subplot(121); %绘制三  
quiver3(X, Y, Z, U, V, W, 0.2);  
subplot(122); %绘制三  
quiver3(X, Y, Z, U, V, W);  
hold on; %绘制三  
surf(X, Y, Z);
```



## 9. 三维彗星图

在MATLAB中，采用函数`comet3( )`绘制三维彗星图，该函数的调用格式和二维彗星图的绘图函数`comet( )`非常相近。  
用户可以在MATLAB的命令行窗口输入：`help comet3`查询该函数的详细使用情况。

```
% chap10_20.m

clear all;
t=-pi:pi/400:pi;
x=sin(5*t);
y=cos(3*t);
z=t;
figure(15);
comet3(x, y, z);      %绘制三维彗星图

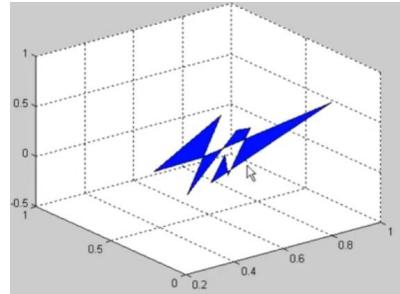
//动态的哦 hh
```

## 10. 三维填充图

在MATLAB中，采用函数fill3( )绘制三维填充图，该函数的调用格式和二维填充图函数fill( )基本相同。用户可以在MATLAB的命令行窗口输入：[help fill3](#)查询该函数的详细介绍。

```
% chap10_21.m

clear all;
x=rand(1, 10);
y=rand(1, 10);
z=x-y;
figure;
fill3(x, y, z, 'b');      %绘制三维填充图
grid on;
```



## 11. 三维彩带图

在MATLAB中，采用函数ribbon( )绘制三维彩带图，该函数的调用格式为：

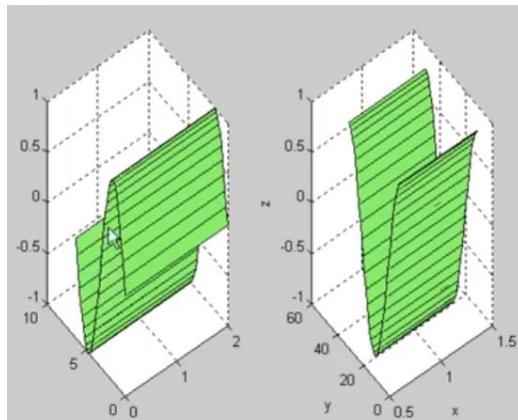
**ribbon(X, Y)**: 该函数绘制三维彩带图，和函数plot( )类似。

**ribbon(Y)**: 该函数绘制三维彩带图，X默认为X=1:size(Y,1)。

**ribbon(X, Y, width)**: 该函数绘制三维彩带图，采用参数width设置线条的宽度。

```
% chap10_22.m
```

```
clear all;
x=0:pi/20:2*pi;
y=sin(x);
figure;
subplot(121);
ribbon(x, y, 2);
subplot(122);
ribbon(cos(x));
xlabel('x');
ylabel('y');
zlabel('z');
```



## 非网格数据绘图

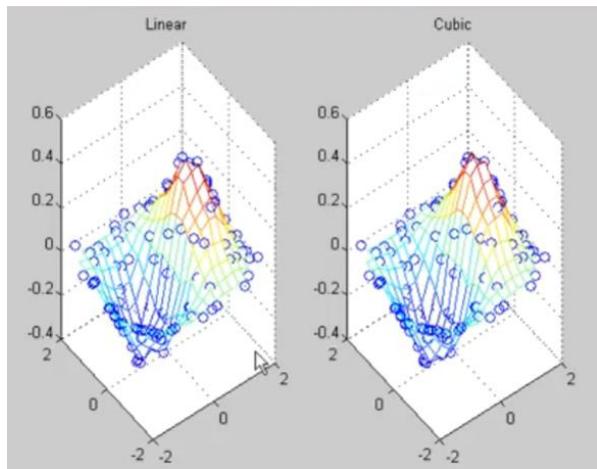
在MATLAB中，采用函数`mesh()`和`surf()`绘图时，要求数据时均匀分布的网格数据。如果数据不是网格数据，可以采用函数`meshgrid()`创建插值的网格数据。

`ZI=griddata(X, Y, Z, XI, YI)`: 该函数中采用数据X、Y和Z建立插值函数 $Z=F(X, Y)$ ，然后计算数据(XI, YI)处的插值结果ZI。

`ZI=griddata(X, Y, Z, XI, YI, method)`: 该函数中采用参数`method`设置插值的算法，可以为：‘linear’为线性插值算法，系统的默认值；‘cubic’为三次插值，‘nearest’为临近点插值等。

```
clear all;
x=rand(1, 100)*4-2;
y=rand(1, 100)*4-2;
z=x.*exp(-x.^2-y.^2);
ti=-2:0.25:2;
[xi, yi]=meshgrid(ti, ti);
figure;
subplot(121);
zi=griddata(x, y, z, xi, yi); %采用默认方法
mesh(xi, yi, zi);
hold on; I
plot3(x, y, z, 'o');
hold off;

title('Linear');
subplot(122);
zi=griddata(x, y, z, xi, yi, 'cubic'); %采
mesh(xi, yi, zi);
hold on;
plot3(x, y, z, 'o'); I
hold off;
title('Cubic');
```



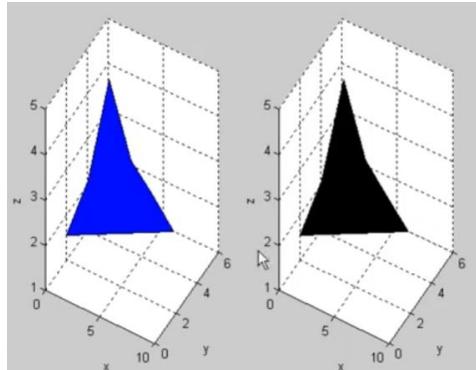
## 创建三维片块模型

在MATLAB中，采用函数patch( )绘制三维片块图，该函数的调用格式为：

**patch(X, Y, Z):** 该函数创建有参数X、Y和Z为顶点的空间多边形。

**patch(X, Y, Z, C):** 该函数中参数C用于设定颜色。

```
clear all;
x=1:6;
y=1:6;
z=[2 3 5 3 2 1];
figure;
subplot(121);
patch(x, y, z, 'b');
view(30, 30);
grid on;
xlabel('x');
ylabel('y');
zlabel('z');
subplot(122);
patch('XData', x, 'YData', y, 'ZData', z);
view(30, 30);
grid on;
xlabel('x');
ylabel('y');
zlabel('z');
```



## 三维隐函数绘图

在MATLAB中，有一些简易绘制三维图形的函数，可以绘制隐含数的三维图形，这些函数如表10.1所示。下面对这些函数分别进行介绍。

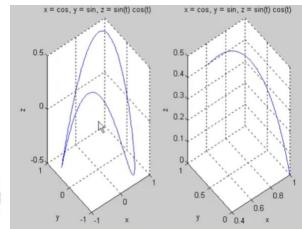
表 10.1 三维简易函数绘图

函数	说明
ezplot3()	绘制三维曲线
ezmesh()	绘制三维网格图
ezmeshc()	绘制带等高线的三维网格图
ezsurf()	绘制三维曲面图
ezsurfc()	绘制带等高线的三维曲面图

```

clear all;
figure;
subplot(121);
ezplot3(@cos, @sin, @(t)sin(t).*cos(t));
subplot(122);
ezplot3('cos', @sin, @(t)sin(t).*cos(t), [0 1])

```



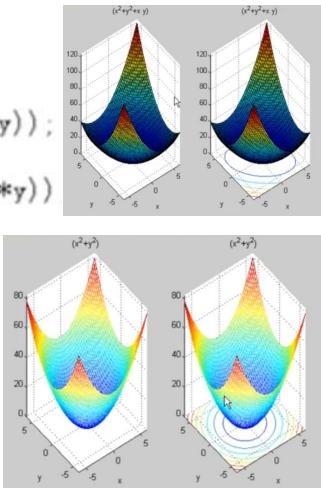
//隐函数与显函数？？？

```

clear all;
close all;
figure;
subplot(121);
ezsurf(@(x,y)(x.^2+y.^2+x.*y));
subplot(122);
ezsurfc(@(x,y)(x.^2+y.^2+x.*y))

clear all;
close all;
figure;
subplot(121);
ezmesh(@(x,y)(x.^2+y.^2));
subplot(122);
ezmeshc(@(x,y)(x.^2+y.^2));

```



## 三维图形的显示

在MATLAB中，三维图形绘制结束后，还需要对三维图形的显示进行控制，主要包括设置视角、颜色设置、光照设置和Camera设置等，这些都是为了方便用户更加清楚的观看三维图形。

### 设置视角

对于三维图形，从不同的视角查看会得到不同的侧面和结果。

在MATLAB中，采用函数view( )设置三维图形的视角，该函数的调用格式为：

**view(az, el)**或**view([az, el])**：该函数设置视角的位置，在方位角为az和仰角为el，它们都以度为单位。

**view([x, y, z])**：该函数设置视角为向量[x, y, z]所指的方向。

**view(2)**：该函数为默认的二维视角。

**view(3)**：该函数为默认的三维视角。

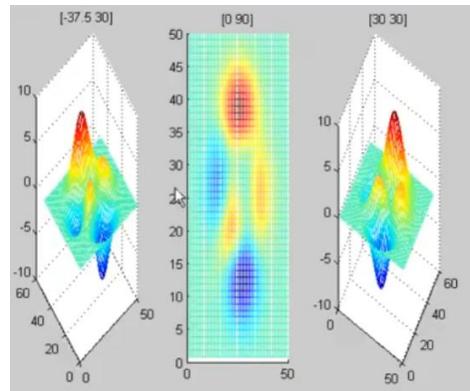
**[az, el]=view**：该函数返回当前的视角az和el。

```

close all;
figure;
subplot(131);
mesh(peaks(50));
[a, b]=view;
title(mat2str([a, b]));
subplot(132);
mesh(peaks(50));
view(2);
[a, b]=view;
title(mat2str([a, b]));
subplot(133);
mesh(peaks(50));
view(30, 30);

[a, b]=view;
title(mat2str([a, b]));

```



在MATLAB中，采用函数**rotate3d( )**设置图形的三维视角，该函数的调用格式为：

**rotate3d on**或**rotate3d('on')**: 该函数打开图形的三维视角，可以方便的用鼠标来控制视角的变化，并进行图形的实时更新。

**rotate3d off**或**rotate3d('off')**: 该函数关闭图形的三维视角功能。

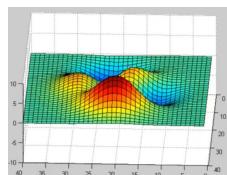
**rotate3d**: 该函数在**rotate3d on**和**rotate3d off**间进行切换。

在利用函数**mesh( )**绘制三维网格图时，默认情况下隐藏三维图形中被遮挡的部分。可以采用函数**hidden( )**显示图形的透视效果，

```

clear all;
close all;
figure;
surf(peaks(40));
rotate3d('on');

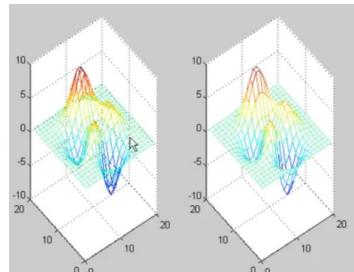
```



```

clear all;
close all;
figure;
subplot(121);
mesh(peaks(20));
hidden off;
subplot(122);
mesh(peaks(20));
hidden on;

```



## 色彩控制

在MATLAB绘图时，可以使用颜色信息让图形更加生动和直观。MATLAB中可以采用RGB真彩色和色图着色。

RGB真彩色是采用颜色映像来处理颜色的，即RGB色系。

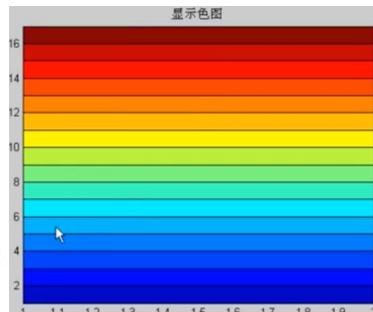
计算机中的各种颜色都是通过三原色（红、绿和蓝）按照不同的比例调制出来的。颜色映象中每一种颜色的值为一个的向量，其中R、G和B为0到1之间的数，即红、黄、蓝三种颜色的强度，形成一种特定的颜色。

除了真彩色，还可以采用色图着色。在MATLAB中内置了很多的色图函数。

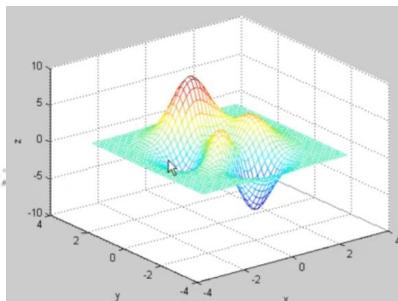
表 10.3 常用色图函数<sup>①</sup>

jet <sup>②</sup>	hsv的一种变形 <sup>③</sup>
hsv <sup>④</sup>	色彩饱和值 <sup>⑤</sup>
hot <sup>⑥</sup>	从黑到红到黄到白 <sup>⑦</sup>
cool <sup>⑧</sup>	青蓝和洋红的色度 <sup>⑨</sup>
spring <sup>⑩</sup>	粉色到黄色 <sup>⑪</sup>
summer <sup>⑫</sup>	从青色到黄色 <sup>⑬</sup>
autumn <sup>⑭</sup>	红色到黄色 <sup>⑮</sup>
winter <sup>⑯</sup>	蓝色到青色 <sup>⑰</sup>
gray <sup>⑱</sup>	线性灰度 <sup>⑲</sup>
bone <sup>⑳</sup>	带一点蓝色的灰度 <sup>⑳</sup>
copper <sup>㉑</sup>	线性铜色度 <sup>㉑</sup>
pink <sup>㉒</sup>	粉红的彩色度 <sup>㉒</sup>
lines <sup>㉓</sup>	线性色图 <sup>㉓</sup>

```
clear all;
close all;
X=hsv;
size(X)
n=16;
Y=jet(n)
colormap(Y)
pcolor([1:n+1;1:n+1]');
title('显示色图');
```



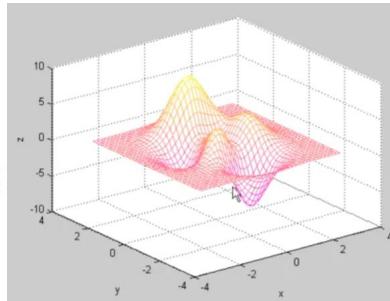
```
clear all;
close all;
[x, y, z]=peaks(40);
figure;
mesh(x, y, z);
colormap('default');
xlabel('x');
ylabel('y');
zlabel('z');
```



```

clear all;
close all;
[x, y, z]=peaks(40);
figure;
mesh(x, y, z);
colormap(spring(32));
xlabel('x');
ylabel('y');
zlabel('z');

```

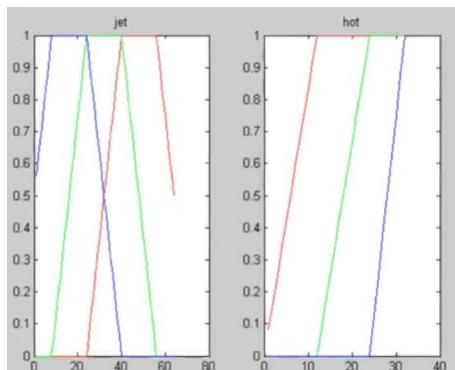


1. 在MATLAB中，可以采用函数rgbplot( )绘制色图。

```

clear all;
close all;
x=jet;
y=hot(32);
figure;
subplot(121);
rgbplot(x);
title('jet');
subplot(122);
rgbplot(y);
title('hot');

```

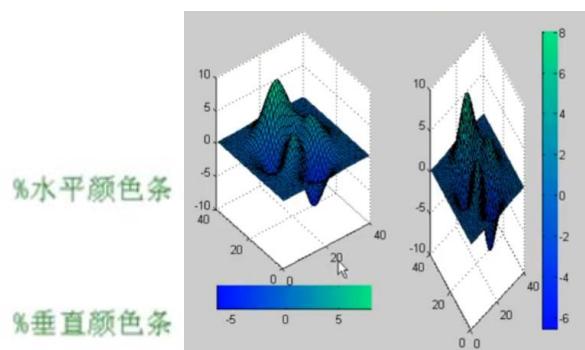


2. 采用函数colorbar( )给当前的图形窗口添加颜色条，显示当前坐标轴的色图。

```

clear all;
close all;
figure;
subplot(121);
lsurf(peaks(40));
colorbar('horiz')
subplot(122);
surf(peaks(40));
colormap(winter(32));
colorbar('vert');

```

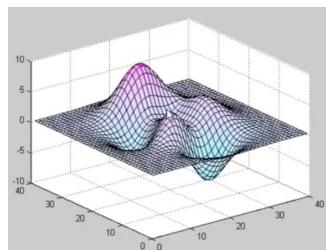


3. 采用函数brighten( )增亮或变暗图形的色图。

```

clear all;
close all;
figure;
lsurf(peaks(40));
colormap(cool);
brighten(0.8);
m1=brighten(0.8);

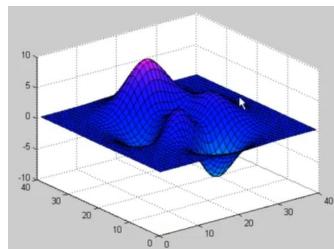
```



```

clear all;
close all;
figure;
surf(peaks(40));
colormap(cool);
brighten(-0.8);
m1=brighten(-0.8);

```

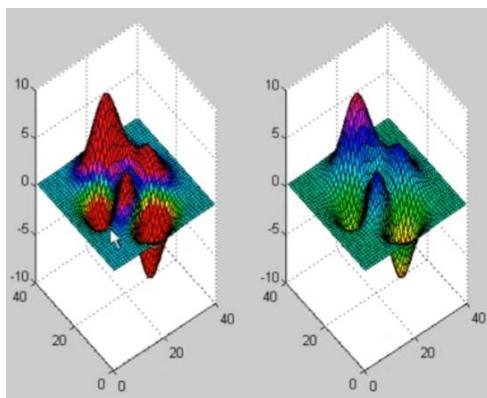


#### 4. 利用函数caxis( )控制颜色轴的刻度，数值和颜色的对应关系以及颜色的显示范围等。

```

clear all;
close all;
figure;
colormap(hsv);
subplot(121);
surf(peaks(40));
caxis([-1.5 1.5]);
v1=caxis
subplot(122);
surf(peaks(40));
caxis auto
v2=caxis

```



```

v1 =
-1.5000    1.5000

v2 =
-6.5497    8.0493

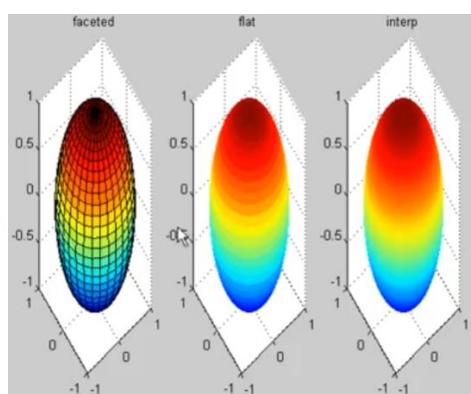
```

#### 5. 采用函数shading( )控制曲面图形的着色方式，该函数用于三维网格图、三维曲面图和三维填充图。

```

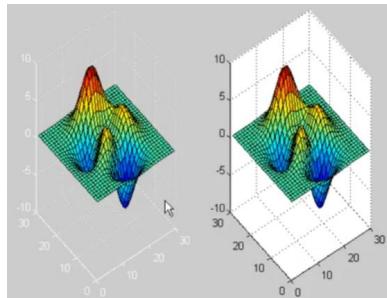
clear all;
close all;
figure;
subplot(131);
sphere(30);
shading faceted;
title('faceted');
subplot(132);
sphere(30);
shading flat;
title('flat');
subplot(133);
sphere(30);
shading('interp');
title('interp');

```



#### 6. 采用函数colordef( )设置图形的背景颜色。

```
clear all;
close all;
figure;
colordef none;
subplot(121);
surf(peaks(30));
subplot(122);
colordef white;
surf(peaks(30));
```



## 光照效果

### 1. 函数light( )

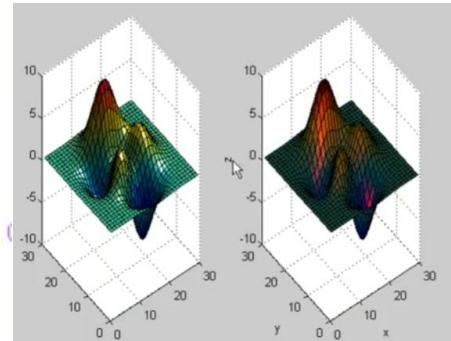
在MATLAB中，可以给图形添加光照效果，利用函数light( )建立光源，该函数的调用格式为：

**light:** 该函数给当前的图形添加光源，光源对象的属性值都是系统默认的。

**light(Parm1, Value1, ..., ParamN, ValueN):** 该函数给参数Parm1赋值为Value1，以此类推，设置光源对象的各个属性值。

**L=light(...):** 该函数返回光源对象的句柄。获取句柄后，可以通过函数get( )和set( )获取和设置属性值。

```
clear all;
close all;
figure;
subplot(121);
surf(peaks(30));
f1=light
get(f1)
subplot(122);
surf(peaks(30));
light('Style','local','Position',[0 0 0],'
 xlabel('x');
 ylabel('y');
 zlabel('z');
```



### 2. 函数lighting( )

在MATLAB中，利用函数lighting( )设置光源的模式，该函数的调用格式为：

**lighting flat或lighting('flat'):** 该函数设置光源模式为flat，即平面模式，以网格为光照的基本单元。该模式为系统默认的模式。

**lighting gouraud或lighting('gouraud'):** 该函数设置光源模式为gouraud，即点模式，以像素为光照的基本单元。

**lighting phong或lighting('phong'):** 该函数设置光源模式为phong，以像素为光照的基本单元，并考虑了各点的反射。

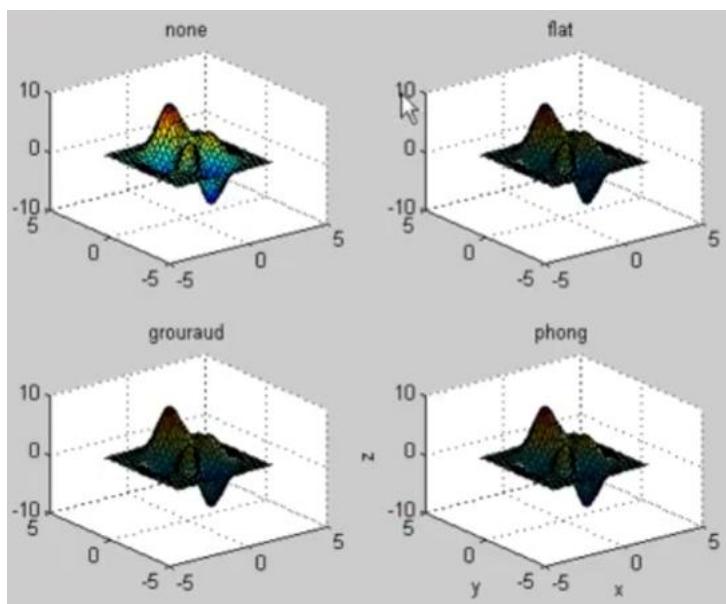
**lighting none或lighting('none'):** 关闭光照效果。

```

clear all;
close all;
[x, y, z]=peaks(30);
figure;
subplot(221);
surf(x, y, z);
lighting none;           I
title('none');
subplot(222);
surf(x, y, z);
light('Position',[1 0 1],'Style','infinite'
lighting flat;
title('flat');
subplot(223);
surf(x, y, z);
light('Position',[1 0 1],'Style','infinite'
lighting gouraud;
title('gouraud');
subplot(224);
surf(x, y, z);
light('Position',[1 0 1],'Style','infinite'
lighting phong;
title('phong');

xlabel('x');
ylabel('y');
zlabel('z');

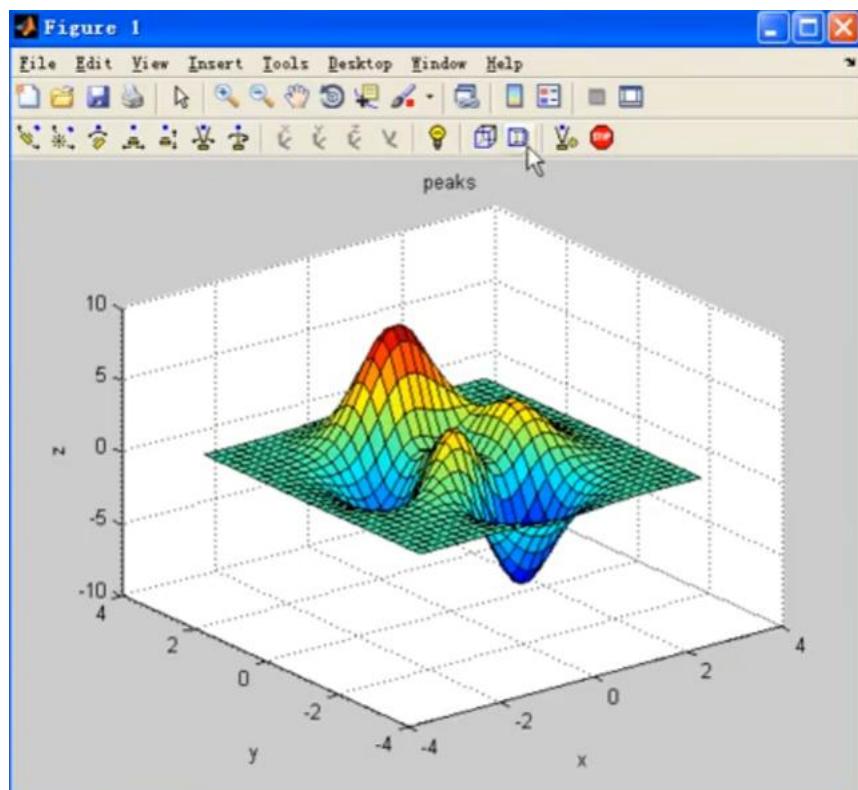
```



## Camera 控制

在默认的图形窗口下， Camera 控制条不显示。在 MATLAB 的命令行输入如下的程序，从而在图形窗口中显示三维曲面图：

```
clear all;           ↘  
close all;  
[x,y,z]=peaks(30);  
figure;  
surf(x,y,z);      %绘制三维曲面图  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('peaks');
```

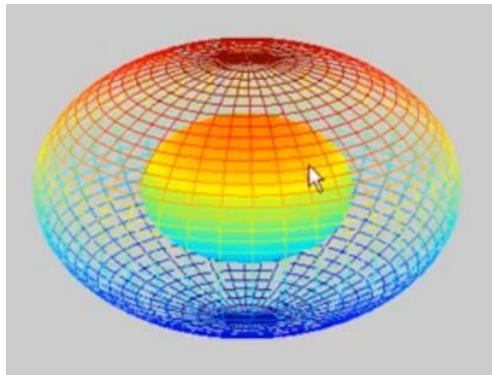


## 图形绘制实例

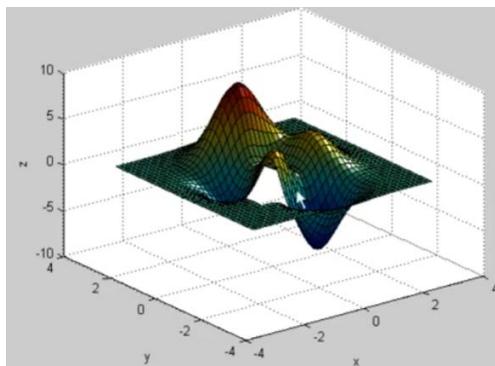
下面对几种典型的图形进行绘制，包括图形的透视效果、图形的镂空，以及用三维图形表现四维的数据等。

1. 图形的透视
2. 图形的镂空
3. 四维绘图
4. 图形动画

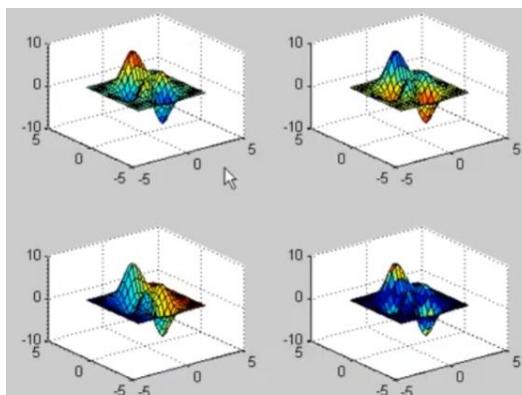
```
clear all;
close all;
[x, y, z]=sphere(40);
figure;
surf(x, y, z);
shading interp;
hold on;
mesh(2*x, 2*y, 2*z);
colormap(jet);
hold off;
hidden off;
axis off
```



```
clear all;
close all;
[x, y, z]=peaks(40);
z(10:15, 10:15)=NaN;
figure;
surf(x, y, z);
light;
lighting flat;
xlabel('x');
ylabel('y');
zlabel('z');
```



```
clear all;
close all;
[x, y, z]=peaks(20);
figure;
subplot(221);
surf(x, y, z, z);
subplot(222);
surf(x, y, z, -z);
subplot(223);
surf(x, y, z, x);
subplot(224);
surf(x, y, z, abs(z));
```

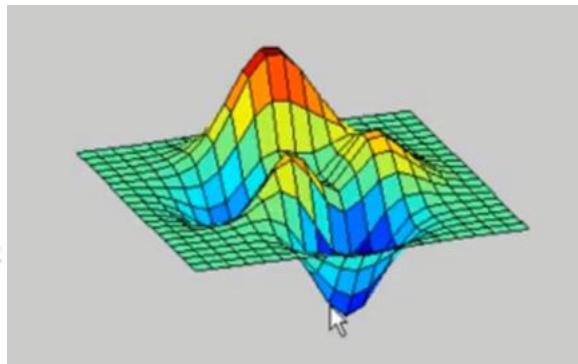


```

clear all;
close all;
[x, y, z]=peaks(20);
figure;
h=surf(x, y, z);
colormap(jet);
axis off;
n=20;
for i=1:n
    rotate(h, [0 0 1], 18);
    mv(:, i)=getframe;
end
movie(mv, 1, 10);

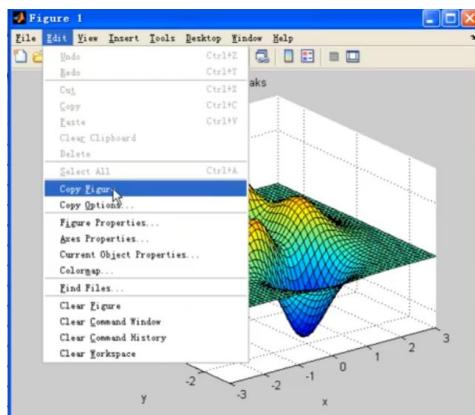
```

//动画哦!!!

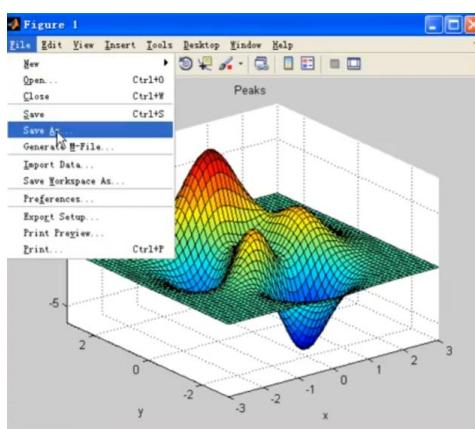


## 图形的输出

利用**MATLAB**绘制了图形后，经常需要以图片的形式保存起来，进行打印输出或利用其它的图片处理软件进行进一步的处理。用户可以采用3种方法将**MATLAB**绘制的图形保存为图片：



//剪切板



// jpg 等

//help print(打印)

## 本章小结

本章对三维图形的绘制和显示进行了详细的介绍。三维图形的绘制主要包括三维曲线图、三维网格图和特殊三维图形的绘制，以及三维隐函数的绘图。三维图形的显示控制，主要包括视角、颜色、光照等。最后介绍了图形的输出。

# MATLAB 基本编程

在使用**MATLAB**的过程中，可以直接在命令行窗口输入程序，采用交互式的方式进行编程，边输入程序边运行。

对于大型的复杂程序，在命令行窗口中直接输入程序会非常的麻烦。**MATLAB**和其它的高级语言一样，可以进行程序的设计，编写扩展名为.m的M文件，实现各种复杂的计算。本章将详细介绍**MATLAB**的编程，程序的流程控制，以及函数编程等。

## MATLAB 编程概述

**MATLAB**是一种解释性语言，本身不能完成任何事情，只是对用户的程序进行解释执行。**MATLAB**语言是采用C语言编写的，因此语法和C语言非常相似。对于熟悉C语言的用户，学习**MATLAB**编程会非常容易。**MATLAB**编写的程序文件有两种：脚本M文件和函数M文件。

### M 文件的创建

在**MATLAB**中，M文件有两种类型：函数（**Functions**）和脚本（**Scripts**），都是以.m作为文件的扩展名。M文件为普通的ASCII文本文件，可以采用文字编辑和处理软件来建立。脚本是一系列命令、语句的简单组合。

### 脚本 M 文件

脚本文件中的变量都为全局变量，程序运行后，这些变量保留在**MATLAB**的基本工作空间内，一般采用函数clear清除这些变量。为了避免因为变量名相同引起冲突，一般在脚本文件的开始，都采用函数clear all，清除所有基本空间中的变量。

```
% chap11_1.m  
% 脚本M文件
```

```
clear all; %  
format long;  
n=40  
s=0  
for i=1:n  
    s=s+1/2^i;  
end  
s  
format short;
```

```

>> help chap11_1.m
chap11_1.m
脚本M文件

>> which chap11_1.m
D:\Mark-ALL-MATLAB\chap11\chap11_1.m
>> whos
  Name      Size            Bytes  Class    Attributes
  i          1x1              8  double
  n          1x1              8  double
  s          1x1              8  double

```

## 函数 M 文件

函数M文件在执行的过程中，所产生的变量一般都是局部变量，存放在函数自身的函数工作空间中，不会和基本工作空间中的变量产生冲突。对用户来说，函数M文件就像一个黑匣子，只有输入和输出。采用函数M文件，非常易于实现程序的模块化，可以实现程序的分工合作、共同开发，适合大型程序开发。

在MATLAB的命令行窗口输入：edit std.m，利用MATLAB文本编辑器打开函数M文件，如图11.2所示。函数std( )用于求样本数据的标准差，程序非常简单，但是包含了函数M文件的全部内容。

```

function z=mymax11(x,y)
% 求最大值
% 如果为两个标量，则返回最大值;
% 如果为向量，则先分别计算两个向量的最大值，然后返回较大的

if nargin==0
    return;
end
t1=max(x);
t2=max(y);
z=max(t1,t2);

>> type mymax11.m

function z=mymax11(x,y)
% 求最大值
% 如果为两个标量，则返回最大值;
% 如果为向量，则先分别计算两个向量的最大值，然后返回较大的

if nargin==0
    return;
end
t1=max(x);
t2=max(y);
z=max(t1,t2);

>> help mymax11
求最大值
如果为两个标量，则返回最大值;
如果为向量，则先分别计算两个向量的最大值，然后返回较大的

```

## 函数的参数传递

在MATLAB中，函数的参数传递是值传递，将变量或常量的值传递给函数的形式参数指定的变量，函数的计算在函数空间中进行，计算结束后，函数空间的变量被清除，函数的返回值返回到MATLAB的基本工作空间中。

在MATLAB中，函数nargin的值为输入参数的个数，函数nargout的值为输出参数的个数。

```
function y=mymax12(x1, x2, x3)
% 计算标量或向量的最大值

if nargin==0
    y=NaN;
    disp('请输入参数！');
    return;
elseif nargin==1
    y=max(x1);
elseif nargin==2
    y=max(max(x1), max(x2));
else
    y=max([max(x1), max(x2), max(x3)]);
end

//nargin!!!(有点用)
```

```
function [y1, y2]=mymin(x1, x2)
% 计算标量或向量的最小值

if nargin==1
    y1=min(x1);
    y2=y1;
elseif nargin==2
    y1=min(x1);
    y2=min(x2);
end
if nargout==1
    y1=min(y1, y2);
end
```

```

function y=mymax(varargin)
% 任意数目的输入参数，求所有元素的最大值

if length(varargin)==0
    disp('请输入参数！');
    return;
else
    tmp=[length(varargin)];
end
for i=1:length(varargin)
    tmp(i)=max(varargin{i}(:));
end
y=max(tmp);

```

## 流程控制

MATLAB编程的基本结构包括：顺序结构、分支结构和循环结构，此外还有一些其它的控制语句。本节首先介绍MATLAB中的变量，然后将详细介绍三种基本的程序结构，最后介绍了交互式输入。

## 变量

在MATLAB中，变量的类型包括：局部变量、全局变量和永久变量。MATLAB为每个变量分配一块内存空间。变量和工作空间紧密联系。

### 1. 局部变量

### 2. 全局变量

全局变量采用**global**进行声明，格式为：**global 变量1 变量2...**  
2. 中间采用空格进行分割。

### 3. 永久变量

永久变量只能在函数M文件中进行定义，格式为：

**persistent 变量1 变量2...** 中间采用空格进行分割。

```

clear all;
global G;
G=1;
x=1;
myglobal(x)    function y=myglobal(x)
myglobal(x)        global G;
G=5;
myglobal(x)        G=G+1;
y=G+x;

```

```
>> clear all;
```

```
ans =
```

```
3
```

```
ans =
```

```
| 4
```

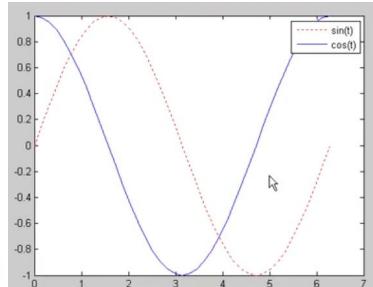
```
ans =
```

```
7
```

## 顺序结构

顺序结构是最简单的程序结构，用户在编写好程序后，按照从上到下、从左到右的顺序执行程序中的语句。由于没有控制语句，结构比较单一，但是这种程序比较容易编写。

```
clear all;
t=0:pi/20:2*pi;
x=sin(t),
y=cos(t);
figure;
plot(t, x, 'r');
hold on;    I
plot(t, y);
legend('sin(t)', 'cos(t)');
```



## 分支结构

分支结构需要进行判断，只有满足一定的条件才执行某些语句。在MATLAB中，分支结构有两类：**if**语句和**switch**语句。下面分别进行介绍。

1. **if...end**语句
2. **if...else...end**语句
3. **if... elseif... else...end**语句
4. **switch...case...end**语句

```
clear all; a =
a=6;
b=4;
if a>b
    a=b;
end
a, b
```

4

```

clear all;
a=12345;
if rem(a, 2)==0
    disp(' 偶数');
else
    disp(' 奇数'); 奇数
end                                >>

clear all;
x=87;
if x>=90
    disp(' 优秀');
elseif x>=80
    disp(' 良好');
elseif x>=70
    disp(' 中等');
elseif x>=60
    disp(' 及格');
else
    disp(' 不及格'); 不及格
end                                    |良好| >>

clear all;
x=5;
switch x
    case 1
        disp(' x=1');
    case 2
        disp(' x=2');
    case {3, 4, 5}
        disp(' x=3, 4, 5');
    otherwise
        disp(' x');
end|                                     |      //状态!!!

```

## 循环结构

循环结构能够重复执行某一段相同的语句。MATLAB中，提供了两种循环语句，分别是**for**循环和**while**循环。如果已知循环次数，通常用**for**循环语句；如果未知循环次数，但有循环条件，则用**while**循环语句。

1. **for**循环语句
2. **while**循环语句
3. **break**语句和**continue**语句

```

    clear all;
for i=1:1:10
    x(i)=i^10;
end
for i=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    y(i)=i^10;
end
i=1:10;
z=i.^10

z =
1.0e+010 *

Columns 1 through 6

    0.0000    0.0000    0.0000    0.0001    0.0010    0.0060

Columns 7 through 10

    0.0282    0.1074    0.3487    1.0000

clear all;
x=zeros(5, 5);
for i=1:5;
    for j=1:5
        x(i, j)=i.^j;
    end
end
x

x =
      1         1         1         1         1
      2         4         8        16        32
      3         9        27        81        243
      4        16        64       256       1024
      5        25       125       625       3125

clear all;
x=zeros(1, 10);
i=1;
while i<=length(x)
    x(i)=1/i;
    i=i+1;
end
x

```

```

clear all;
x=magic(6)
for i=1:size(x, 1)
    for j=1:size(x, 2)
        if x(i, j)==5
            break;
        else
            x(i, j)=0;
        end
    end
end
x

```

//break 退出一层循环

```

clear all;
x=magic(6)
for i=1:size(x, 1)
    if rem(i, 2)==0
        continue;
    end
    x(i, :)=0;
end
x

```

### try...catch 语句

在程序设计时，如果不能确定某段程序代码是否会出现错误，可以采用**try...catch**语句。在MATLAB中，使用**try...catch**语句能够捕获和处理错误，其调用格式为：

```

try
    程序代码1
catch
    程序代码2
end

```

```

clear all;          a =
a=[1, 2, 3; 2, 3, 4]      1   2   3
b=[3, 4, 5; 1, 6, 7]      2   3   4
try
    c=a*b;          b =
    disp('c=a*b');
catch
    disp(lasterr); Error using ==> mtimes
end                         Inner matrix dimensions must agree.
```

```

## 人机交互函数

在MATLAB中提供了一些人机交互的函数，用户可以从键盘输入数据或字符串，可以暂停程序的运行等。下面分别进行介绍。

### 1. 函数input( )

### 2. 函数keyboard

### 3. 函数pause

### 4. 函数echo

### 5. 函数return

```
clear all;
r1=input('Do you like apple? Y/N: ','s');
while isempty(r1)
    r1=input('Do you like apple? Y/N: ','s');
end
disp(r1);
disp('input matrix, for example:eye(5)');
r2=input('','s');
f=eval(r2)

clear all;
a=[1 2 3;2 3 4]
b=[4 5 3;7 8 9]
keyboard
c=a+b

a =
1     2     3
2     3     4

b =
4     5     3
7     8     9

K>> a

a =
1     2     3
2     3     4

K>> a

K>> return
//调试
```

```
clear all;
t=0:pi/20:2*pi;
x=sin(t);
figure;
plot(t, x)
xlabel('t');
ylabel('x');
hold on;
for i=1:7
    pause 1
    plot(t, sin(t+i/5));
    hold on;
end

clear all;
a=[1 2 3; 2 3 4];
b=[1 2 3; 7 8 9];
echo on;
c=a+b
d=a-b
echo off;
e=a.*b
```

## 函数类型

在MATLAB中，函数M文件可以分为：主函数、子函数、嵌套函数、私有函数、重载函数和匿名函数。下面分别进行介绍。

### 主函数

每个函数M文件中第一行定义的函数是主函数，一个函数M文件只有一个主函数。主函数的名字通常和函数的名字相同。除了主函数，在函数M文件中，还有嵌套函数或子函数。主函数对嵌套函数或子函数进行调用。

## 子函数

在主函数之后编写的函数为子函数。在函数M文件中，只有一个主函数，但是可以有多个子函数。所有的子函数也采用**function**进行定义。如果主函数中包含子函数，则每个采用**function**定义的函数必须采用**end**结束。各个子函数的先后顺序和调用的先后顺序无关。

在主函数进行函数调用时，首先查找该函数文件中的子函数，如果有同名的子函数，则调用该子函数。因此，可以编写同名的子函数实现函数重载。子函数只能被同一文件的主函数或其它子函数调用。

```
clear all;
[a, b]=mymain(3, 4)
[x, y]=mymain(5, 6)
help mymain
help mymain>mysub1
help mymain>mysub2

function [a, b]=mymain(x, y)
% 主函数
a=mysub1(x, y);
b=mysub2(x, y);
end

function z=mysub1(x, y)
% 子函数1
z=x-y;
end

function z=mysub2(x, y)
% 子函数2
z=x+y;
end
```

## 嵌套函数

在函数的内部，还可以定义一个或多个函数，称为函数的嵌套。在函数内部定义的函数称为嵌套函数。MATLAB中，可以进行多层嵌套，一个函数的内部可以嵌套多个函数，这些嵌套函数内部又可以嵌套其它函数。

```
function a=A(x,y)
    % B(x,y);
    % D(x,y);
    a=1
    function b=B(x,y)
        C(x);
        D(y);
        b=2
        function c=C(x)
            B(x)
            D(x);
            c=3
        end
    end
    function d=D(x,y)
        B(x,y);
        E(y);
        d=4
    end
    function e=E(y)
        B(x);
        D(x);
        e=5
    end
end
```

## 私有函数

私有函数要保存在该目录的**private**文件夹下，具有有限的访问权限。私有函数的编写和普通函数的编写没有什么区别，可以是一个主函数和多个子函数，以及嵌套函数等。

函数M文件可以直接调用私有函数。该函数M文件所在的目录下直接有一个**private**目录，私有函数就保存在该目录下。私有函数只能被其父目录内的脚本或函数调用。

脚本M文件不能直接调用私有函数，必须通过调用一个在私有函数的父目录中的函数来间接调用私有函数。

对于私有函数，也可以通过**help**获取该函数的帮助信息。例如要获取私有函数**myfun( )**的帮助信息，需要通过如下的命令：**help private/myfun**。

```

function z=myprivatefun(x,y)
% 这是私有函数 myprivatefun()

    disp('这是私有函数！');
    if nargin==2
        z=x+y;
    elseif nargin==1
        z=x;
    else
        warning('没有输入参数！');
        return;
    end

>> help private/myprivatefun
这是私有函数 myprivatefun()

```

## 重载函数

在MATLAB中很多函数都有重载函数，重载函数具有相同的函数名，但是参数类型或个数不同。当用户在调用函数时，MATLAB会根据实际的参数类型和个数来选择其中的一个。在MATLAB中，重载函数保存在不同的目录下，通常目录的名字以符号@开头，然后跟一个代表MATLAB数据类型的字符。例如@int8目录下存放的重载函数的输入参数类型为int8，即8位有符号整数。

Example: If  $A = [2 \ 3 \ 4 \ 7 \ 3 \ 9]$  then  $\max(A, 1)$  is  $[1 \ 3 \ 9]$ ,

$\max([], 2)$  is  $[8 \ 9]$  and  $\max(5)$  is  $[5 \ 5]$ .

See also [min](#), [median](#), [mean](#), [sort](#).

Overloaded methods:

- [int8/max](#)
- [timeseries/max](#)
- [codistributed/max](#)
- [fints/max](#)
- [localpoly/max](#)
- [localpspline/max](#)
- [ordinal/max](#)

Reference page in Help browser  
[doc max](#)

//多个重载函数

```

function m=max(x, y, z, w, varargin)
% 新建的重载函数 max()
% 该函数求三个标量的最大值
% author, copyright
if nargin~=4
    return;
end
temp=x;
if x<y
    temp=y;
end
if temp<z
    temp=z;
end
if temp<w
    temp=w;
end
m=temp;

clear all;
help int8/max.m
a=int8(1);
b=int8(2);
c=int8(4);
d=int8(8);
max(a, b, c, d)

```

## 匿名函数

匿名函数的定义格式如下：

**fhandle=@(arglist) expr**

其中**arglist**是匿名函数的输入参数列表，采用逗号分割。

**expr**为具体的函数表达式，执行该匿名函数要完成的功能。**@**是MATLAB的操作符，用来建立函数句柄。匿名函数的返回值**fhandle**为函数句柄，可以采用函数句柄来调用该函数。

```

clear all;
f=@(x) 2*x.^2+3*x+4
a1=f(2)
a2=f(3)
a3=f(4)

clear all;
f=@(x, a, b)a*x.^2+x.*b
f(1, 2, 3)
f(2, 2, 3)
f(1:3, 2, 3)

```

```

clear all;
f1=@()disp('This is an anonymous function')
f1()
functions(f1)|

f1 =
|   []
@()disp('This is an anonymous function')

This is an anonymous function

ans =

function: '@()disp('This is an anonymous function')'
  type: 'anonymous'
    file: ''
workspace: {[1x1 struct]}

clear all;
a=[1, 2, 3; 2, 3, 4]
f1=@(x) size(x)
f1(a)
[b, c]=f1(a)

clear all;
f1=@(a, b, c)@(x)a*x.^2+x.*b+c
f1_info=functions(f1)
f2=f1(1, 2, 3)
f2_info=functions(f2)
f3=f2(3)|      []

//多重匿名函数

clear all;
F1={@(x) 3*x+4, @(y) 8*y+9, @(a) 6*a+2}
F2={@(x) 3*x+4; @(a) 5*a+10}
F1{1}(2)
F2{2, 1}(3)

clear all;
f1=@(x)fzero(@(y) sin(x)*cos(y)+x*exp(y)-3*x.^2*y.^2, 1)
f1(3)
f1(5)          T

f1 =
|   []
@(x)fzero(@(y)sin(x)*cos(y)+x*exp(y)-3*x.^2*y.^2, 1)

ans =
|   []
0.4162

ans =      []
0.2746

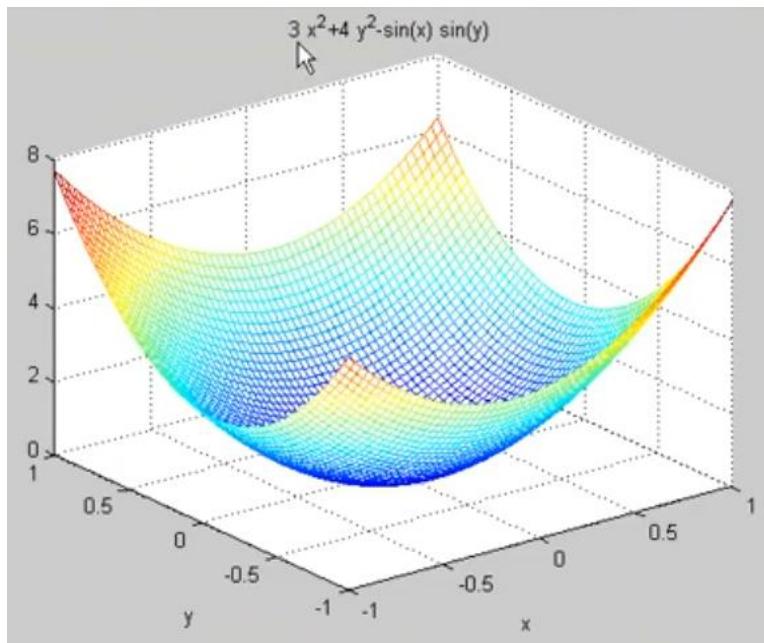
```

```

clear all;
f1=@(a) qu&#233d(@(x) 3*x.^2+a*x+5, 0, 1)
f1(1)
f1(3)

clear all;
close all;
figure;
ezmesh(@(x,y) 3*x^2+4*y^2-sin(x)*sin(y), [-1, 1], [-1, 1]);

```



## 函数句柄

创建函数句柄使用`@`或者`str2func`命令。可以使用函数句柄来调用其它函数，也可以将函数句柄存储在数据结构中，方便以后使用。当使用函数句柄调用该函数时，MATLAB会立即执行，不进行文件搜索。当多次调用一个函数文件时，可以节省大量的搜索时间，从而提高函数的执行效率。

```

clear all;
A={@sin, @cos, @(x)x.^2-2}
figure;
t=-pi:pi/20:pi;
for i=1:length(A)
    plot(t,A{i}(t));
    hold on;
end
xlabel('x');
ylabel('y');

```

```
clear all;
A.x=@sin;
A.y=@cos;
A.z=@(x)sin(x).*cos(x);
t=0:pi/10:4*pi
figure;
plot3(A.x(t), A.y(t), A.z(t));
grid on;
```

## P 码文件和 ASV 文件

在MATLAB中，可以将M文件转换为P码文件。P码文件可以直接执行，速度比M文件要快。此外，在MATLAB的编程过程中，还会产生临时保存的ASV文件。下面分别进行详细介绍。

### P 码文件

P码文件是对应M文件的源代码不可见，但是可以执行的版本。P码文件的后缀为.p，采用函数pcode( )生成脚本M文件或函数M文件的P码文件。第一次执行M文件时，MATLAB需要将其解析（parse）一次（第一次执行后的已解析内容会放入内存作第二次执行时使用，即第二次执行时无需再解析），这增加了程序的执行时间。

产生P码文件之后，如果再调用M文件时，便会直接执行其P码文件。MATLAB的解析速度非常快，一般不用产生P码文件。只有当一些程序要调用到非常多的M文件时，如GUI应用程序时，才会作预解析，以增加以后的调用速度

```
clear all;
X=[1 2 3;3 4 5]
Y=[4 5 6;7 8 9]
A=X-Y
```

## ASV 文件

在该目录内有很多的.asv文件。可以采用dir进行查询，也可以采用ls进行查询。可以用记事本打开看一下，为正在编写的MATLAB源程序自动保存。如果matlab正常关闭的话，删除这些.asv文件没有任何影响。

在M文本编辑器中，单击菜单File| Preferences...，会弹出一个窗口，选择Editor/debugger下的Autosave，可以修改asv文件的设置，可以选择是否自动保存。如果选择自动保存，几分钟后进行自动保存，保存的临时文件的后缀是什么等等。

```
>> dir *.asv
chap11_11.asv  chap11_27.asv  chap11_36.asv  mymax.asv
chap11_12.asv  chap11_28.asv  chap11_37.asv  mymax11.asv
chap11_16.asv  chap11_30.asv  chap11_8.asv   mymax12.asv
chap11_18.asv  chap11_32.asv  chap11_9.asv
chap11_20.asv  chap11_33.asv  max.asv
chap11_24.asv  chap11_35.asv  myglobal.asv

>> ls *.asv
chap11_11.asv  chap11_27.asv  chap11_36.asv  mymax.asv
chap11_12.asv  chap11_28.asv  chap11_37.asv  mymax11.asv
chap11_16.asv  chap11_30.asv  chap11_8.asv   mymax12.asv
chap11_18.asv  chap11_32.asv  chap11_9.asv
chap11_20.asv  chap11_33.asv  max.asv
chap11_24.asv  chap11_35.asv  myglobal.asv
```

## 本章小结

本章详细的介绍了MATLAB的编程，包括脚本M文件和函数M文件。详细的介绍了MATLAB编程时，程序的流程，包括顺序结构、分支结构和循环结构。然后重点介绍了MATLAB的函数类型和函数编程，以及函数句柄等。最后介绍了P码文件、ASV文件和MAT文件。

# 程序调试和编程技巧

在编写MATLAB程序时，难免会出现错误，这时就需要对程序进行调试。MATLAB中，M文件的调试主要有两种方法：直接调试法和工具调试法。

编写的程序在能够完成特定的功能的情况下，还要对程序的性能进行分析，例如运算速度和效率等。在MATLAB中，采用Profile工具对程序的性能进行分析。在本章的最后，介绍了一些常用的编程技巧。

## M 文件调试

对于编程者来说，程序出现错误在所难免，尤其是在编写多人参与的大型程序时，掌握常用的程序调试方法非常重要。下面首先介绍出错信息，然后介绍常用的直接调试法和工具调试法等。

### 出错信息

程序调试是一个很基础也很重要的问题。没有人能一遍写出完全正确没有错误的程序。错误的程序大致分为以下三类：

**拼写错误：**比如应该是sum( )，写成了smu( )。拼写错误非常容易发现，在程序运行时，系统会提示错误。

**语法错误：**比如6/0，输出结果变为无穷大Inf。程序在运行时不一定会报错，但是输出结果不正常。

**逻辑错误：**这样的错误非常隐蔽，通常是对程序的算法考虑不周全。程序可以正常执行，但是输出结果不符合预期值。

## 直接调试法

利用MATLAB编写脚本文件或函数，非常灵活，得到的程序也非常的简洁，比较容易阅读和理解。对于不是很复杂的程序，可以使用直接调试法进行程序的调试。

- (1) 如果程序出错或给出警告信息，仔细分析出错或警告信息，会得到出错的原因。
- (2) 在程序中，利用函数**disp( )**将函数M文件的中间结果显示出来。
- (3) 在单独调试一个函数M文件时，可以将该函数M文件修改为脚本M文件进行调试。
- (4) 将程序的某些行通过注释进行屏蔽，在程序行的前面输入注释符%。
- (5) 在程序中需要调试的地方添加**keyboard**函数。

## 工具调试法

在MATLAB中，有强大的程序调试工具，可以在命令行窗口采用调试命令进行调试，也可以采用**Debugger**工具在图形窗口中进行程序的调试。下面分别进行介绍。

### 1. 命令行为主的调试

在命令行窗口中输入：**help debug**可以查看调试命令。

**dbstop**设置断点。**dbclear**清除断点。**dbcont**从断点处恢复执行。**dbstack**调用堆栈。**dbstatus**列出所有断点。**dbstep**执行一行或多行程序。**dbtype**查看程序的代码。

### 2. 图形化程序调试

---

```
>> help debug
DEBUG List M-file debugging functions

dbstop      - Set breakpoint.
dbclear     - Remove breakpoint.
dbcont      - Resume execution.
dbdown      - Change local workspace context.
dbmex       - Enable MEX-file debugging.
dbstack     - List who called whom.
dbstatus    - List all breakpoints.
dbstep      - Execute one or more lines.
dbtype      - List M-file with line numbers.
dbup        - Change local workspace context.
dbquit      - Quit debug mode.
```

```
% chap12_1.m
```

```
clear all;
x=[1, 2, 3; 2, 3, 4];
y=[1:3;1:3];
a=x./y;
b=x+y;
```

> dbstop : + 1.0 + + 1.1 x |  
> dbstop  
> 1 % chap12\_1.m  
2  
3 ● clear all;  
4 ● x=[1, 2, 3; 2, 3, 4];  
5 - y=[1:3;1:3];  
6 - a=x./y;  
7 - b=x+y; I  
8 |

Breakpoints for chap12\_1 are on lines 3, 4.

```
>> chap12_1
3 clear all;
K>> dbstep
4 x=[1, 2, 3; 2, 3, 4];
K>>
```

Breakpoints for chap12\_1 are on lines 3, 4.

```
>> chap12_1
3 clear all;
K>> dbstep
4 x=[1, 2, 3; 2, 3, 4];
K>> x
??? Undefined function or variable 'x'.
```

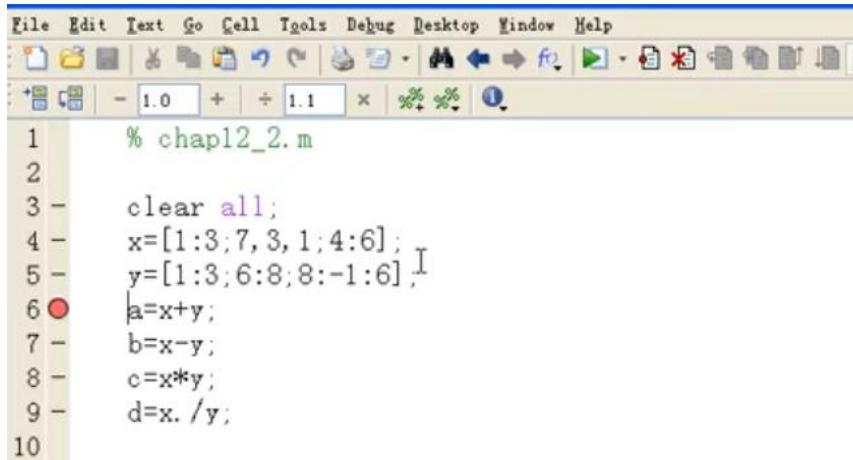
```
K>> dbstep
5 y=[1:3;1:3];
K>> x
x =
```

```
1 2 3
2 3 4
```

```
K>> dbcont
```

```
>> whos
Name      Size            Bytes  Class       Attributes
a         2x3              48  double
b         2x3              48  [double
x         2x3              48  double
y         2x3              48  double
```

```
% chap12_2.m  
  
clear all;  
x=[1:3;7,3,1;4:6];  
y=[1:3;6:8;8:-1:6];  
a=x+y;  
b=x-y;  
c=x*y;  
d=x./y;
```



## 错误处理

在程序运行过程中，如果遇到警告或错误，需要进行处理。

在MATLAB中，可以采用try...catch语句来捕获错误。

可以利用函数error( )发出错误信息，利用函数warning( )发出警告信息。

1. 使用try...catch语句
  2. 使用函数error( )
  3. 使用函数warning( )

% chap12\_3.m

```

clear all;
a=[1 2 3;2 3 4]
b=[1:4;2:5]
try
    c=a/b;
    d=a+b;
catch
    disp('系统出错了');
end

clear all;
while 1
    x=input('请输入正数: ');
    if x<=0
        error('你输入的数是: %d, 不是正数!', x);
    end

```

请输入正数: 3  
 请输入正数: 5  
 请输入正数: -3  
 ??? Error using ==> chap12\_4 at 7  
 你输入的数是: -3, 不是正数!

```

clear all;
a=[1 2 3;2 3 4]
b=[1:4;2:5]
try
    c=a+b;
catch
    err=lasterror
    err.message
    err.stack
end
lasterror('reset');
err=lasterror

clear all;
a=5;
if a>3
    warning('该数%d大于3', a);
    b=a
    warning('BiggerThanThree', '该数%d大于3', a);
end

s1=warning('query','last')
s2=warning('query','all')
warning('on','all');

clear all;
s1=warning('query','last')
s2=warning('query','all')
s2(1)
s2(2)
warning(s2);

```

## M 文件性能分析

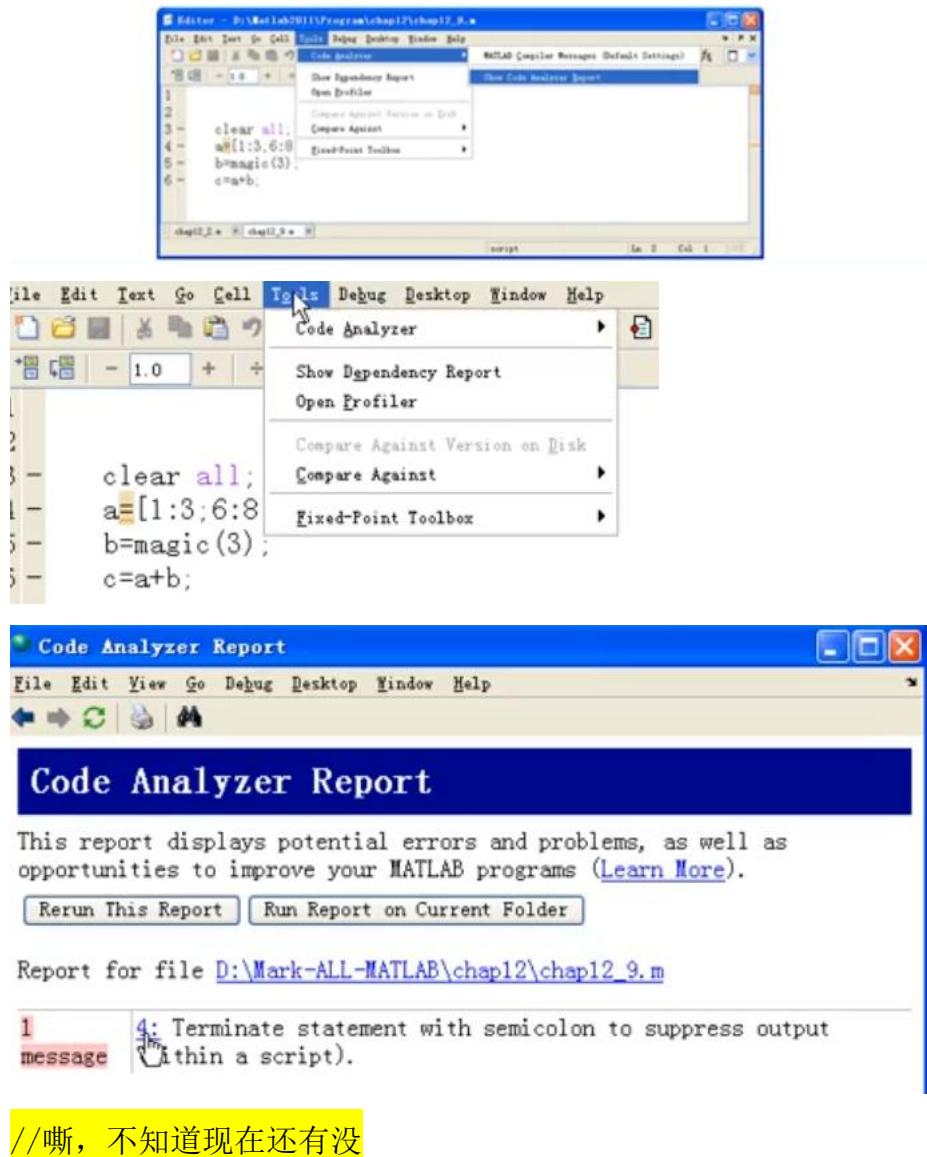
在**MATLAB**中，通过程序的调试，可以改正程序中的语法错误和逻辑错误等。程序调试结束以后，用户编写的**MATLAB**程序可以正常运行了，但可能并不是最优的，运行效率也不是最好的。这就需要对**MATLAB**程序的性能进行分析，并进行优化。

**MATLAB**提供的M文件分析工具包括：**Code Analyzer**工具和**Profiler**工具，它们都有图形操作界面，使用起来非常的方便，下面进行详细的介绍。

### Code Analyzer 工具

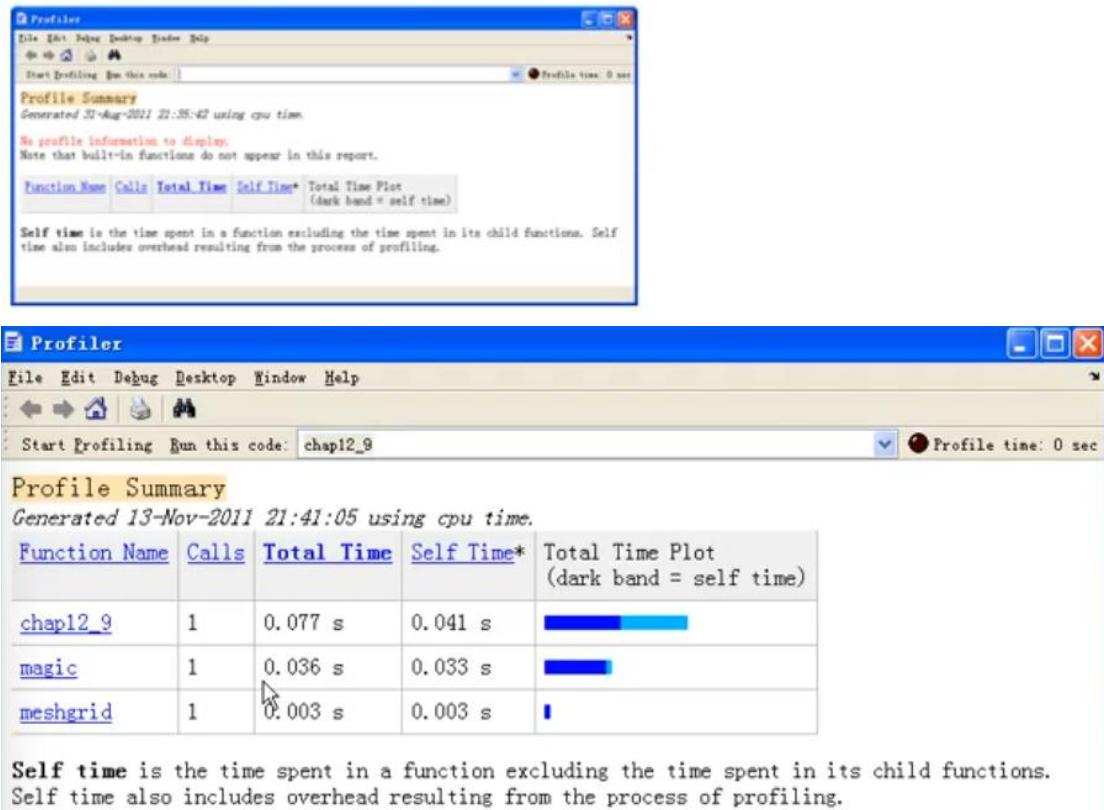
在**MATLAB**中，可以利用**Code Analyzer**工具来分析程序中的错误和性能等。在以前的**MATLAB**版本中叫**M-lint**，基本功能类似，只是名称不一样。

用户首先采用M文件编辑器打开要分析的M文件，然后选择M文件编辑器中Tool菜单下的Code Analyzer选项，如图12.8所示。其中菜单MATLAB Compiler Message Default Setting将Code Analyzer工具设置为默认值，单击菜单Show Code Analyzer Report，会在浏览器窗口中显示分析报告。



### Profiler 分析工具

在MATLAB中，采用Profiler工具对MATLAB代码进行分析。用户可以在MATLAB的主界面中单击菜单Desktop|Profiler，显示Profiler工具。在Run this code后面输入程序文件名或程序语句后，通过单击按钮Start Profiling进行程序的性能分析。



## 编程技巧

由于 MATLAB 语言是一种解释性语言，有时 MATLAB 程序的执行速度不是很理想。下面给出了一些编程的技巧，例如：尽量采用向量运算来代替循环，对于大型的矩阵预先给出维数等。

## 程序执行时间

在 MATLAB 中，可以对程序进行计时，从而获得程序的执行时间。在本书第 2 章的 2.3.4 节，给出了 3 种计时方法，分别是 1) 采用函数 **tic** 和 **toc** 进行计时；2) 利用函数 **etime** 进行计时；3) 利用 **cputime** 进行计时。

如果程序不小心进入死循环，或者计算时间太长，用户可以在命令窗口中使用 **Ctrl+c** 来中断当前程序的执行。

## 编程技巧

- (1) 在利用 MATLAB 编程时，应尽量避免使用循环。
- (2) 如果必须使用多重循环，但是两个循环执行的次数不同，则建议在外循环执行循环次数少的，内循环执行循环次数多的。这样也可以显著提高速度。

(3) 对于大型矩阵，如果预先定维，会减少程序的执行时间，提高程序的运算效率。

在程序中，给大型矩阵动态的确定维数，花费的时间非常多。建议在定义大矩阵时，首先用 MATLAB 的内在函数，如函数zeros( )或ones( )对之先进行定维，然后再进行赋值处理，这样会显著减少所需的时间的。

(4) 在编写程序时，优先考虑MATLAB的内在函数。

(5) 如果用户想继续提高程序的效率，可以考虑采用更加有效的算法。

(6) 此外，还可以尝试采用Mex 技术。

```
clear all;
x=rand(3, 4, 5);
tic;
m=max(x(:))
t=toc

clear all;
tic;
i=1:10000;
m=sum(1./i.^2);
m
t=toc

clear all;
tic;
H=zeros(5, 20000);
for i=1:5
    for j=1:20000
        H(i, j)=1/(i+j-1);
    end
end
toc
```

## 小技巧

下面给出在利用**MATLAB**编程时，可能会用到的一些小技巧。在有些时候这些小技巧，也能起到事半功倍的效果。

- (1) 对于M文件的命名，尽量不要用简单的英文单词，最好是由大小写英文字母、数字和下划线等组成。
- (2) 在调试程序时，经常要屏蔽掉多行程序，可以在选定程序后，按快捷键**Ctrl+r**将这些程序行注释掉。
- (3) 快捷键**Ctrl+c**可以中断正在执行的操作。
- (4) 使用**Tab**键补全函数名或文件名。
- (5) 在命令行窗口中，使用上下光标键↑和↓，将历史记录中的命令复制到输入位置。
- (6) 使用文本编辑器的**cell**模式，能使程序更加清晰。

## 本章小结

本章给出了在**MATLAB**编程时，程序调试的方法，包括命令行下的直接调试法和工具调试法。介绍了**Code Analyzer**工具和**Profiler**分析工具，这些工具能够帮助用户对程序进行分析，实现程序的特定功能，并提高程序的运算效率。

最后，给出了一些常用的编程小技巧，这些小技巧的灵活应用，用户在编写程序时，能够起到事半功倍的效果，需要灵活掌握。

# Simulink 基本知识

**Simulink**是MATLAB软件的重要组成部分，能够进行动态系统的建模、仿真和综合分析，在学术和工业领域的应用越来越广泛。本章详细的介绍了**Simulink**建模的基本概念，然后介绍了**Simulink**的基本操作，例如**Simulink**的启动、模块的连接和模块参数的设置等。接着详细的介绍了**Simulink**常用的基本模块库，这些基本模块是建模的基本元素，了解各个模块的作用是熟练掌握**Simulink**建模的基础。最后介绍了**Simulink**中的子系统和封装，以及**Simulink**模型的工作空间。

## Simulink 概述

**Simulink**是MATLAB软件最重要的组件之一，能够进行动态系统建模、仿真和综合分析，提供了交互式图形化环境。在**Simulink**中，把现实中的每个系统都看成是由输入、输出和状态这3个基本元素组成，并随时间变化的数学函数关系。本节对**Simulink**的基本概念进行详细的介绍。

## Simulink 的概念

**Simulink**是MATLAB中的一种可视化仿真工具，是实现动态系统建模、仿真和分析的一个软件包，被广泛应用于线性系统、非线性系统、数字控制及数字信号处理的建模和仿真中。**Simulink**可以用连续采样时间、离散采样时间或两种混合的采样时间进行建模，它也支持多速率系统，即系统中的不同部分具有不同的采样率。为了创建动态系统模型，**Simulink**提供了一个建立模型方块图的图形用户接口（GUI），这个创建过程只需单击和拖动鼠标操作就能完成，它提供了一种更快捷、直接明了的方式，而且用户可以立即看到系统的仿真结果。

## Simulink 的应用和特点

**Simulink**提供一个动态系统建模、仿真和综合分析的集成环境。在该环境中，无需大量书写程序，而只需要通过简单直观的鼠标操作，就可构造出复杂的系统。

**Simulink**具有适应面广、结构和流程清晰及仿真精细、贴近实际、效率高、灵活等优点，并基于以上优点

**Simulink**已被广泛应用于控制理论和数字信号处理的复杂仿真和设计。同时有大量的第三方软件和硬件可应用于**Simulink**。

## Simulink 的基本操作

利用**Simulink**建模的过程，可以简单地理解为从模块库中选择需要的模块，然后将这些模块进行连接和属性设置，最后进行系统的运行，并查看仿真结果。下面对**Simulink**的基本操作进行详细的介绍，主要包括启动**Simulink**、模块的操作和属性设置，以及仿真器设置等。

### 启动 Simulink

在进行MATLAB安装时，选择**Simulink**模块，进行**Simulink**的安装。在MATLAB 2010a中，**Simulink**的版本为7.5。首先启动MATLAB软件，然后再启动**Simulink**。用户可以单击MATLAB主界面工具栏的按钮，或在MATLAB的命令行窗口输入：`simulink`，来启动**Simulink**。

用户可以在MATLAB的命令行输入：`help simulink`，来查看**Simulink**命令行下的函数。

用户在命令行输入：`demo simulink`，将会打开MATLAB的帮助系统，并显示**Simulink**的例子程序。

## 选择模块

首先打开Simulink的库浏览器窗口。然后单击Simulink库浏览器窗口工具栏上的New按钮，新建模型文件，并保存为chap13\_1.mdl。

在Simulink库的Sources子库中，找到正弦信号模块Sine Wave。可以用鼠标直接将该模块拖入到文件 chap13\_1.mdl中，也可以在该模块上单击右键并选择 Add to chap13\_1，或选中该模块后按快捷键Ctrl +I。在 Sources子库中找到常量模块Constant并添加。在Sinks 子库中找到示波器Scope，并添加2个示波器。在数学库 Math Operations中，找到加法Add模块和减法模块 Subtract，并添加到模型文件中。

## 模块的连接

将鼠标放在模块的连线处，鼠标箭头会变成一个十字叉，然后单击并按下鼠标左键，拖动鼠标到其它模块的输入口，鼠标箭头变为双十字叉后，放开鼠标左键后。将模型文件chap13\_1连线后，如图13.6所示。



图13.5 选择和添加模块

图13.6 模块的连接

模块间的连线默认为黑色的实线，如果两个模块间的连线为红色的虚线，表示两个模块并没有连接起来。用鼠标单击该线，然后按键盘的Delete键或单击鼠标右键并选择 Delete菜单进行删除。

## 模块的基本操作

### 1、选中模块

在模型文件中，用鼠标单某个模块来选中该模块，被选中模块的四角处会出现小黑块编辑框。如果想选定多个对象，可以按下Shift键，然后再单击所需选定的模块。也可以用鼠标拉出矩形虚线框，将所有待选模块框在其中，则矩形框中所有的对象均被选中。

## 2、复制模块

在同一个模型文件中，可以采样如下的方法进行模块的复制：

选定该模块，按下鼠标右键，拖动模块到合适的地方，释放鼠标。

选定该模块，按住**Ctrl**键，用鼠标拖动到合适的地方，释放鼠标。

选中该模块，然后使用菜单或工具栏中的**Copy**和**Paste**按钮。

## 3、模块的移动

在同一个模型文件中，选中需要移动的一个或多个模块，然

后用鼠标将模块拖到合适的地方。还可以在不同模型文  
件移动模块。用鼠标选中要移动的模块，然后之间拖入  
到其它模型文件中。如果在移动的同时按下**Shift**键，则  
删除原来模型文件中的模块。

## 4、改变模块大小

选定需要改变大小的模块，出现小黑块编辑框后，用鼠标拖  
动编辑框，可以实现放大或缩小。

## 5、模块的删除

对于不需要的模块，需要进行删除。选中需要删除的模块，然  
后按键盘上的**Delete**键进行删除。或选中模块后，单击菜  
单**Edit**下的**Delete**或**Cut**选项。也可以在选中模块后，单击  
工具栏中的**Cut**按钮进行删除。

## 6、模块的翻转

首先选中模块，选择模型文件中的菜单**Format**下的**Flip Block**  
选项，可以将模块旋转180度。菜单**Format**下的选项  
**Rotate Block**可以将模块旋转90度。

此外，利用**Format**菜单下的选项，还可以修改模块名，对模  
块名的字体进行设置，隐藏模块名，模块名的翻转等。

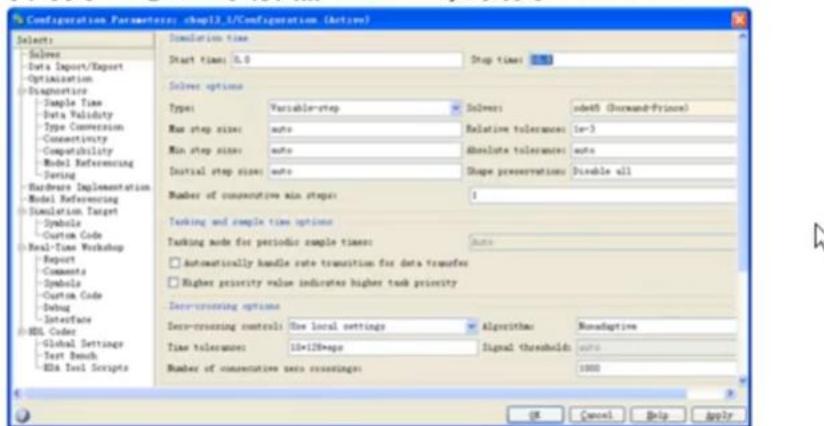
## 模块参数的设置

对于每个模块，都可以用鼠标双击该模块，打开参数设置窗口，对模块进行参数设置。对于正弦信号（Sine Wave），用鼠标双击该模块后，会出现如图13.7的参数设置对话框。在图13.7的上部为参数说明，仔细阅读可以帮助用户设置参数。正弦信号的参数中，Amplitude为正弦信号的幅值，设置为1。Bias为幅值偏移值，设置为0。Frequency为正弦频率，设置为5。Phase为正弦的初相，设置为0。Sample time为采样时间，设置为0.01。用鼠标双击常数模块Constant，会出现图13.8所示的参数设置窗口。设置常数为3，如图13.8所示。



## 仿真器设置

在模型文件窗口中，单击菜单Simulation下的Configuration Parameters选项，会出现Simulink仿真器设置窗口，如图13.9所示。仿真的起始时间为0，终止时间为10秒，求解器Solver默认为ode45。



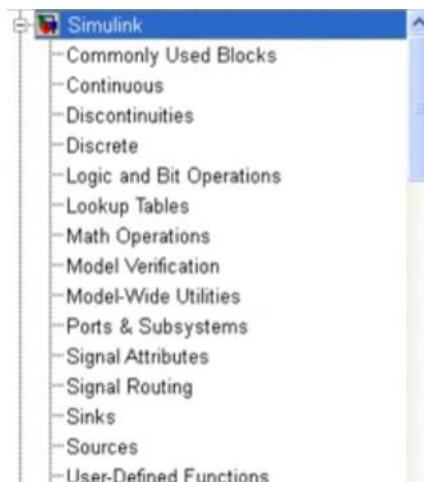
## 运行仿真

首先建立仿真模型，然后对各个模块的参数进行设置，以及Simulink仿真器进行设置。最后进行系统的仿真。用户可以单击模型文件chap13\_1.mdl窗口中的菜单Simulation下的Start，或按快捷键Ctrl+T，进行系统的仿真。仿真结束后，单击示波器，可以查看进行加法和减法后的正弦波形，分别为图13.10和图13.11。在图13.10中，幅值范围为[2, 4]，在图13.11中，幅值范围为[-4, -2]。用户可以单击示波器工具栏中的按钮，来调制纵轴的范围。此外，还可以单击工具栏中的按钮，对示波器的参数进行设置。



## 常用的模块库

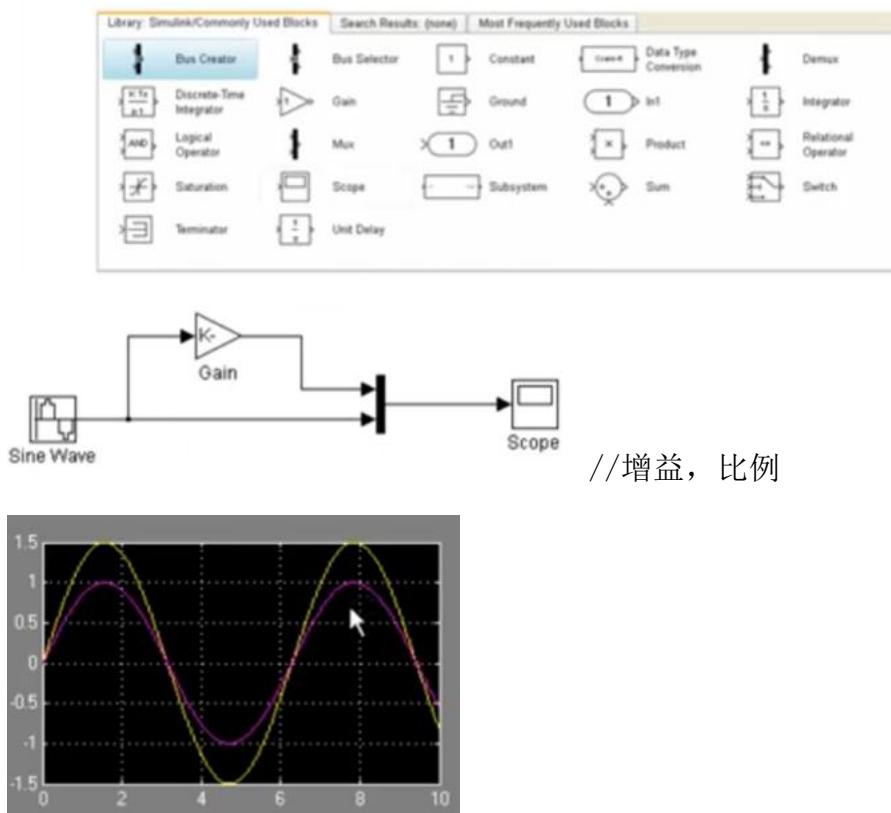
在Simulink模块库浏览器中，Simulink模块包括两类，一类是Simulink基本模块库，是进行系统建模的基本单元。另一类是各个工具箱的模块库，和各个具体应用领域相关。Simulink基本模块库包含16个子集。下面介绍Simulink的基本模块库。



## Simulink 常用模块子集

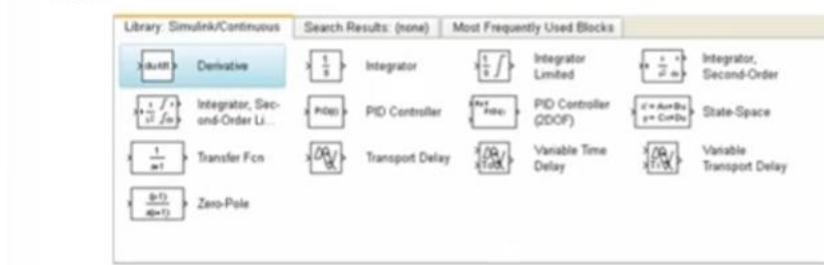
Simulink常用模块子集包括22个基本模块，如图13.12所示。

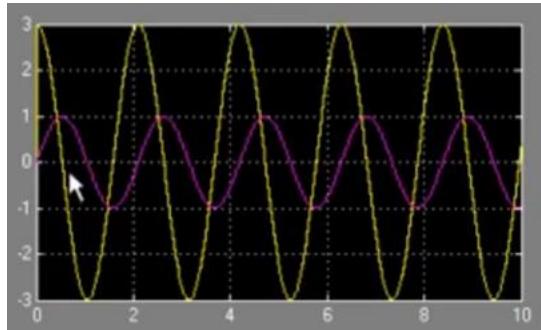
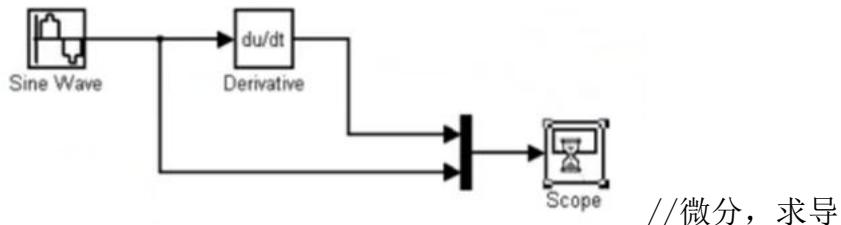
下面介绍其中的**Mux**模块和**Gain**模块。**Mux**模块将多个单一输入信号合成一个复合输出。**Gain**模块进行比例运算。



## 连续时间模块子集

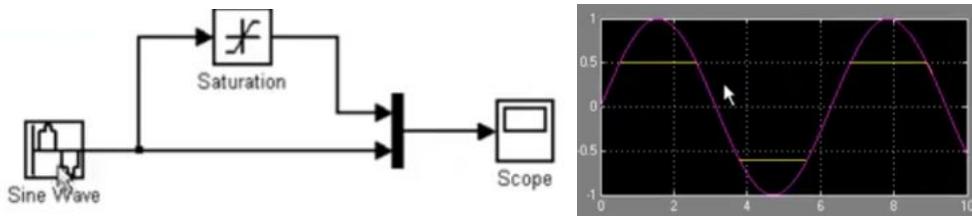
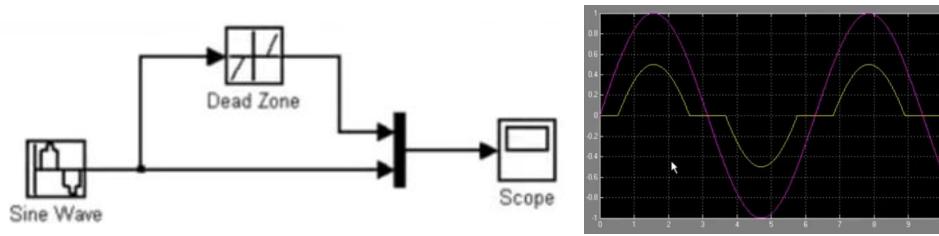
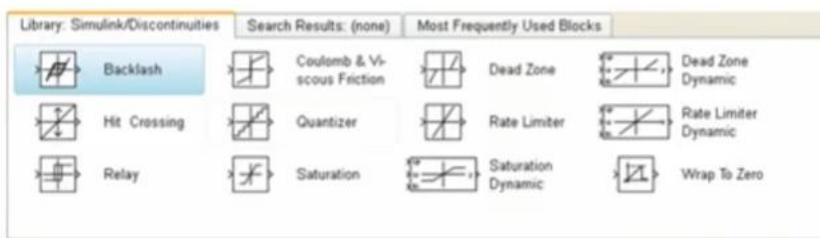
连续时间模块子集包括13个模块，如图13.17所示。下面介绍其中的**Derivative**模块（微分器）。微分器计算输入信号的导数 $du/dt$ ，其中 $du$ 是输入信号的变化量， $dt$ 是时间的变化量。





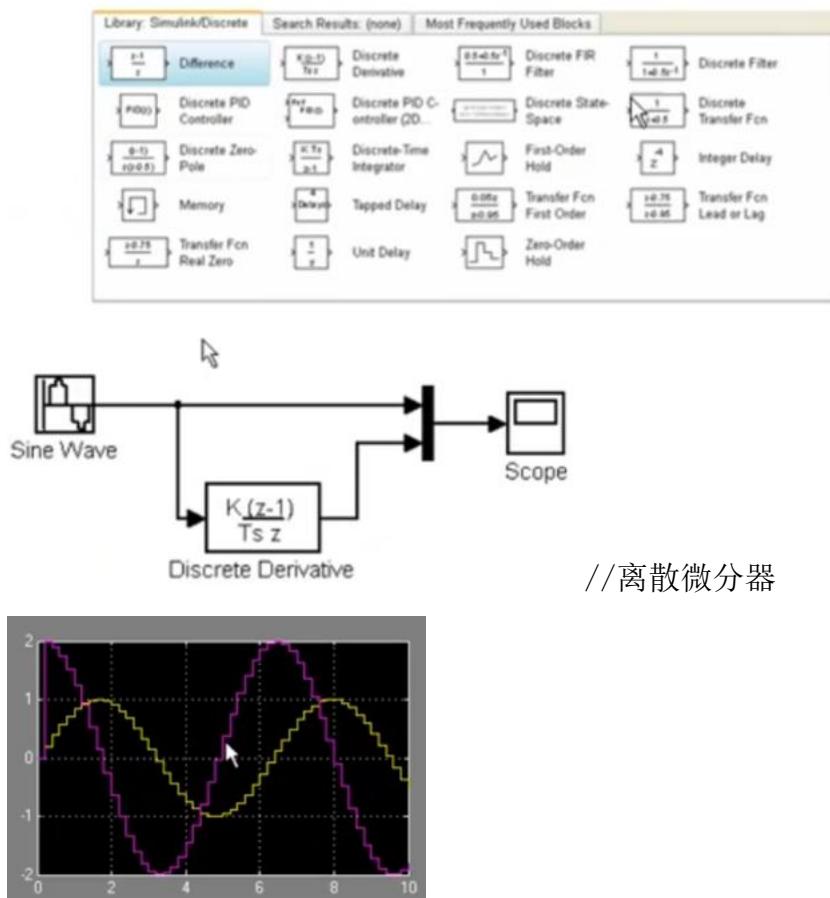
## 非连续时间模块子集

非连续时间模块子集共有12个模块，如图13.20所示。下面介绍其中的**Dead Zone**模块和**Saturation**模块。**Dead Zone**模块产生死区，输入在某一范围取值时，输出为0。**Saturation**模块，可以对输入信号的范围进行限制。当输入信号超出规定的上限值时，输出为该上限值；当输入信号小于规定的下限值时，输出为该下限值。



## 离散时间模块子集

离散时间模块子集，如图13.25所示，共包括19个模块。下面介绍Discrete Derivative模块，即离散微分器。

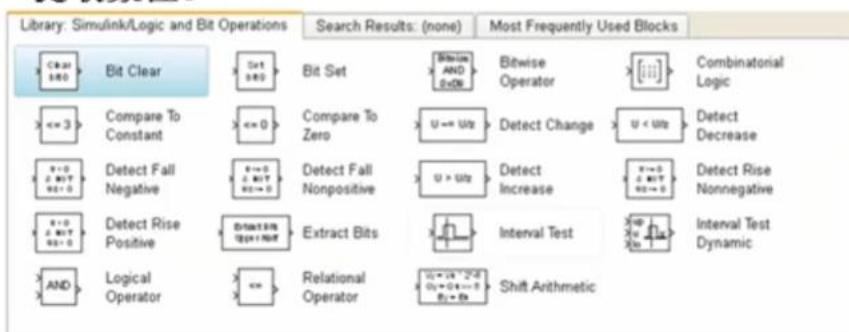


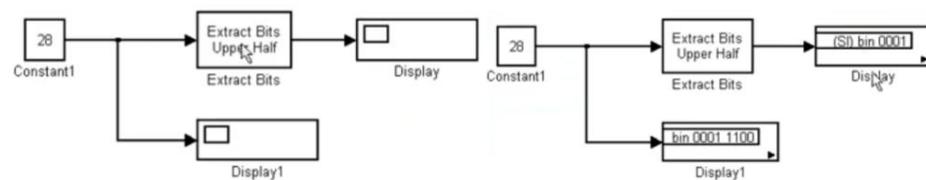
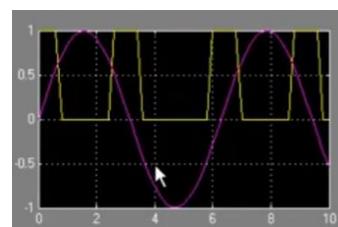
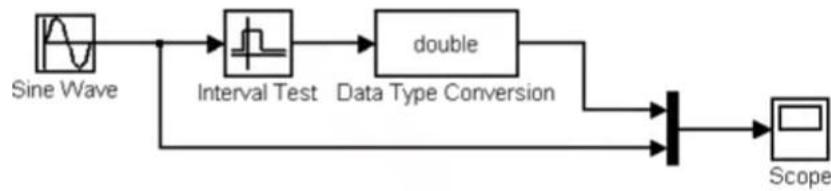
## 逻辑和位操作子集

逻辑和位操作模块共包括19个基本模块，如图13.28所示。

下面介绍其中的Interval Test模块和Extract Bits模块。

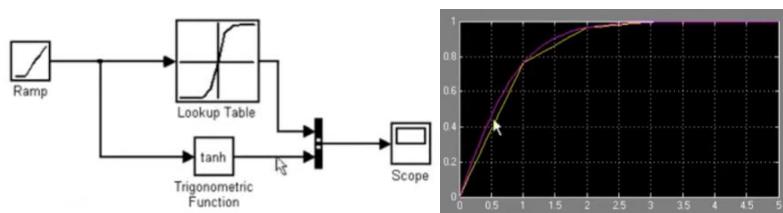
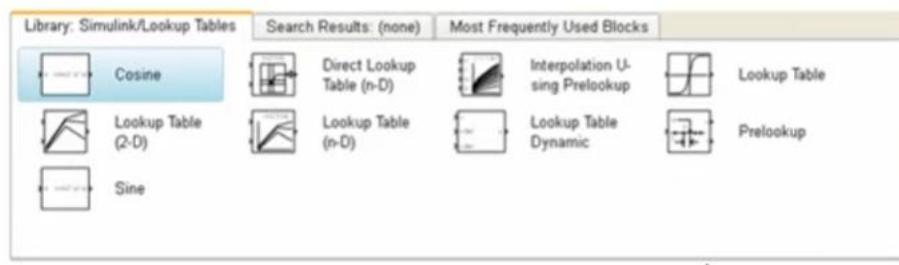
**Interval Test**模块用于间隔测试，**Extract Bits**模块用于提取数位。





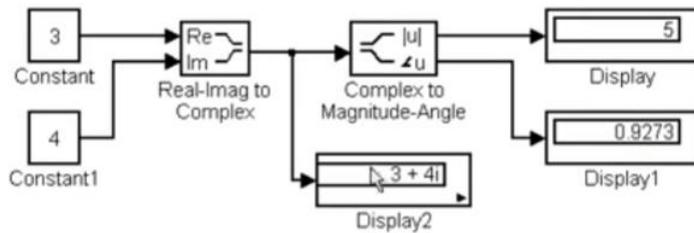
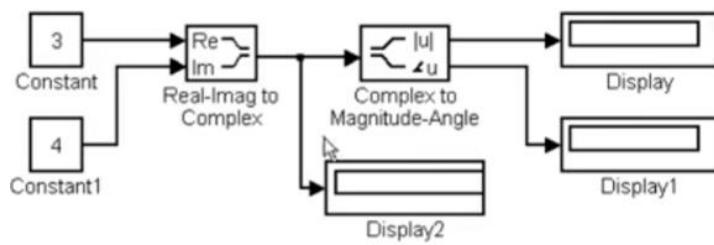
## 查表模块子集

在查表模块子集中包括9个模块，如图13.33所示。查表模块子集提供了常用查表方法的模块，包括一维、二维和多维查表模块等。下面详细的介绍**Lookup Table**模块，即查表模块。



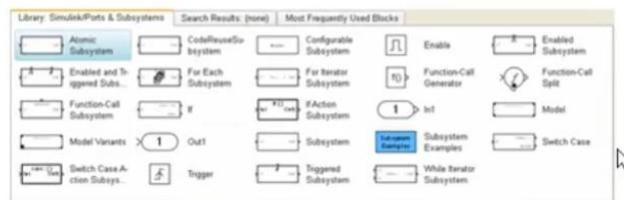
## 数学运算模块子集

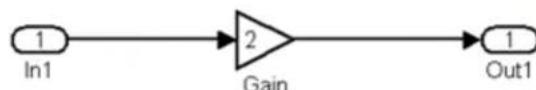
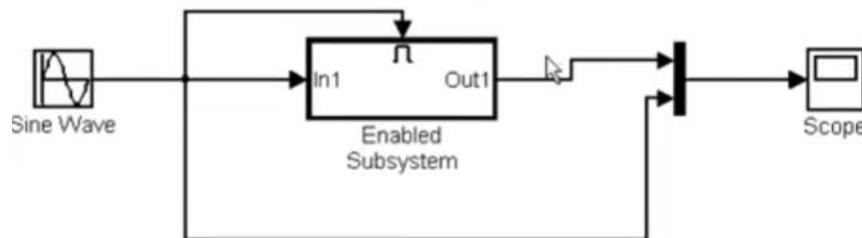
Simulink的数学运算模块共包含37个基本模块，这些模块用来进行常用的数学运算，如图13.38所示。下面介绍**Real-Imag to Complex**模块和**Complex to Magnitude-Angle**模块。**Real-Imag to Complex**模块通过实部和虚部组成复数，**Complex to Magnitude-Angle**获取复数的幅度和角度。



## 端口和子系统模块子集

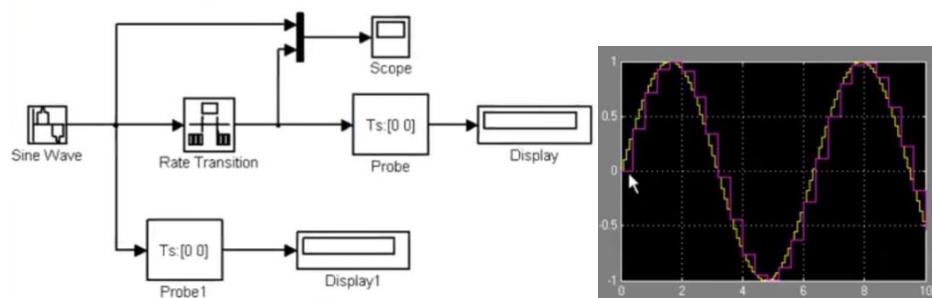
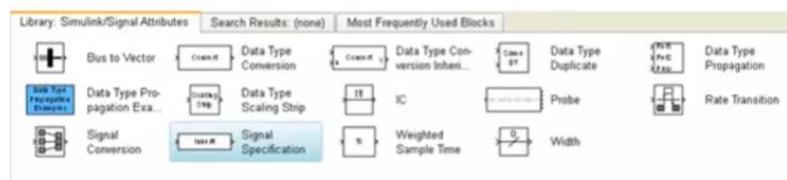
端口和子系统模块子集共包括24个基本模块，如图13.4.1所示。下面介绍其中的**Enabled Subsystem**模块，即使能子系统模块。将**Enable Subsystem**模块加入系统后，就变为了一个使能子系统。系统在仿真时，首先判断使能信号是否大于0，只有在使能端大于0时，子系统才运行，否则子系统不运行。





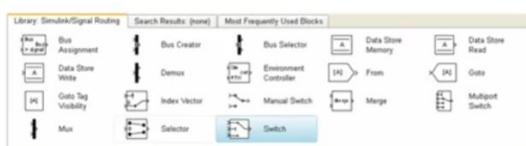
## 信号特征模块子集

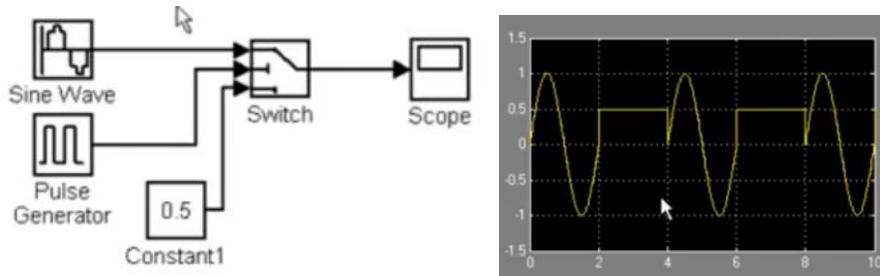
Simulink的信号特征模块子集包括14个基本模块，如图13.46所示。这些模块可以对信号的数据类型、采样率、宽度等特征进行检测和操作。下面介绍其中的**Rate Transition**模块和**Probe**模块。**Rate Transition**模块用于改变信号的采样率。**Probe**模块用于获取信号的宽度、维数、采样时间等信息。



## 信号路径模块子集

在Simulink的信号路径模块子集，共包含18个基本模块，如图13.51所示，其中的一些模块也是常用模块子集的一部分。下面介绍其中的**Switch**模块，根据条件对输入信号进行选择。该模块有3个输入端，从上到下依次为u1、u2和u3，只有一个输出端。





## Sinks 模块子集

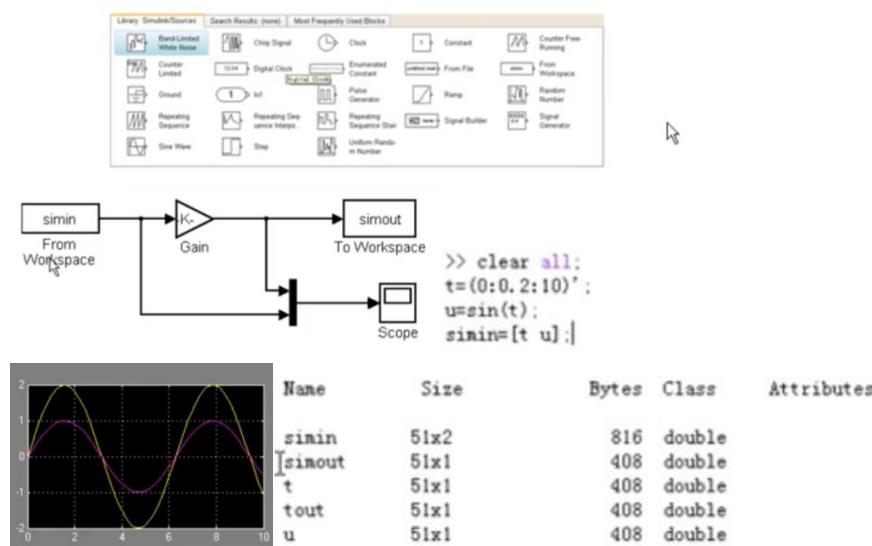
Simulink的Sinks模块子集共包括9个模块，如图13.56所示。

Sinks模块子集的很多模块在前面的例子中已经用过。To File模块将Simulink中某个变量保存到文件中，To Workspace将某个变量保存到MATLAB的基本工作空间中。



## 信号源模块子集

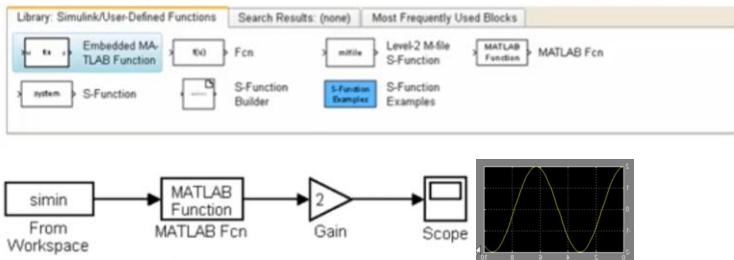
信号源模块子集共包含23个基本模块，如图13.57所示，用来产生仿真模型的信号源和时间等。From File模块将保存在文件中的数据读入到Simulink模型中。From Workspace模块将MATLAB基本工作空间中的变量读入到Simulink模型中，可以实现MATLAB和Simulink之间的数据交换。



## 用户定义模块子集

用户定义模块子集包括6个基本模块和S-Function

Examples模块，如图13.62所示。S-function模块，在第14章会详细的介绍。这里对MATLAB Fcn模块进行介绍。MATLAB Fcn模块对输入进行规定的数学运算，输入和输出都必须是双精度的浮点型，但可以是实数也可以是复数。

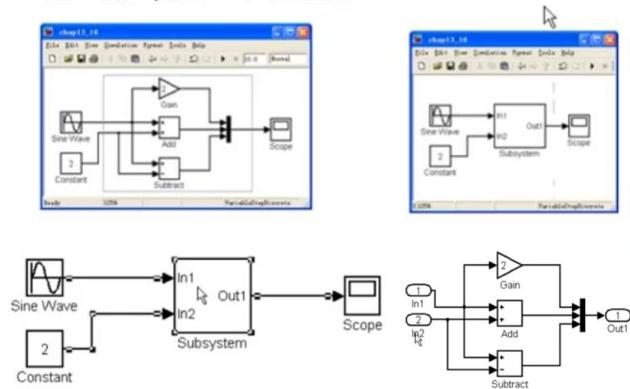


## 子系统及其封装

当模型变得越来越大、越来越复杂时，使用的模块非常多，用户很难轻易读懂所建立的模型。因此，可以将大的模型分成一些小的子系统，每个子系统非常简单、可读性好，能够完成某个特定的功能。通过子系统，可以采用模块化设计方法，层次非常清晰。有些常用的模块集成在一起，还可以实现复用。

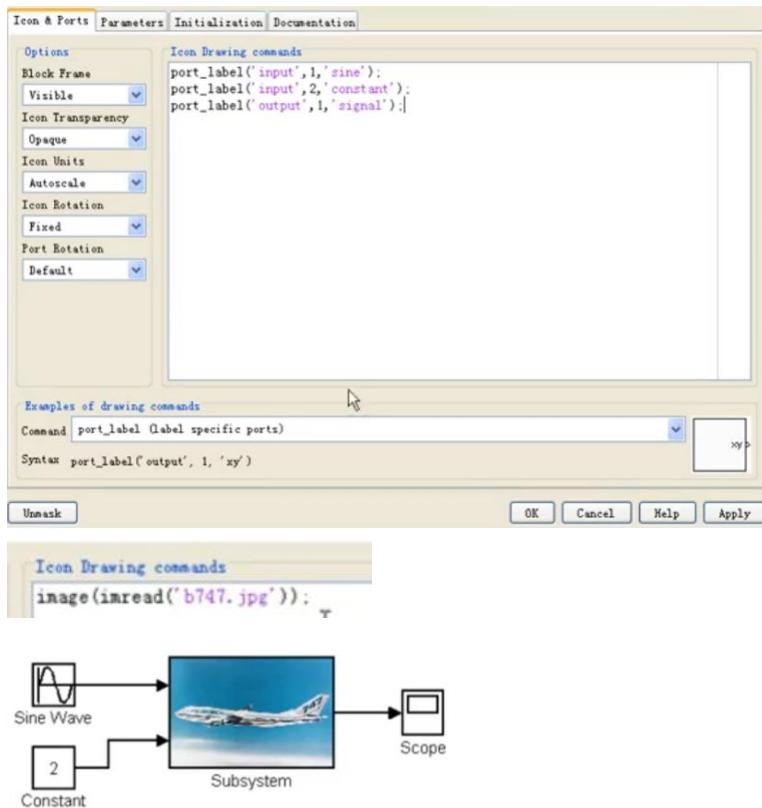
## 子系统

建立子系统的方法有2种。第一种方法是采用Simulink模块库的端口和子系统模块子集。Subsystem模块可以建立子系统，Triggered Subsystem模块用来建立触发子系统，Enabled Subsystem模块建立使能子系统。在本章的13.3.8节已经进行了介绍，给出了一个使能子系统的例子。另一种方法是把系统模型中已有的一部分模块组合在一起，建立一个子系统。



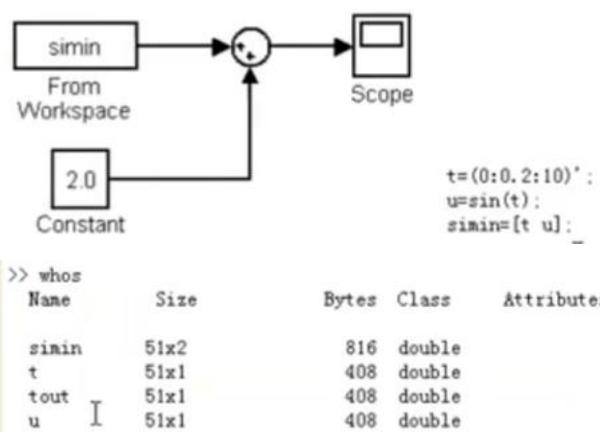
## 子系统的封装

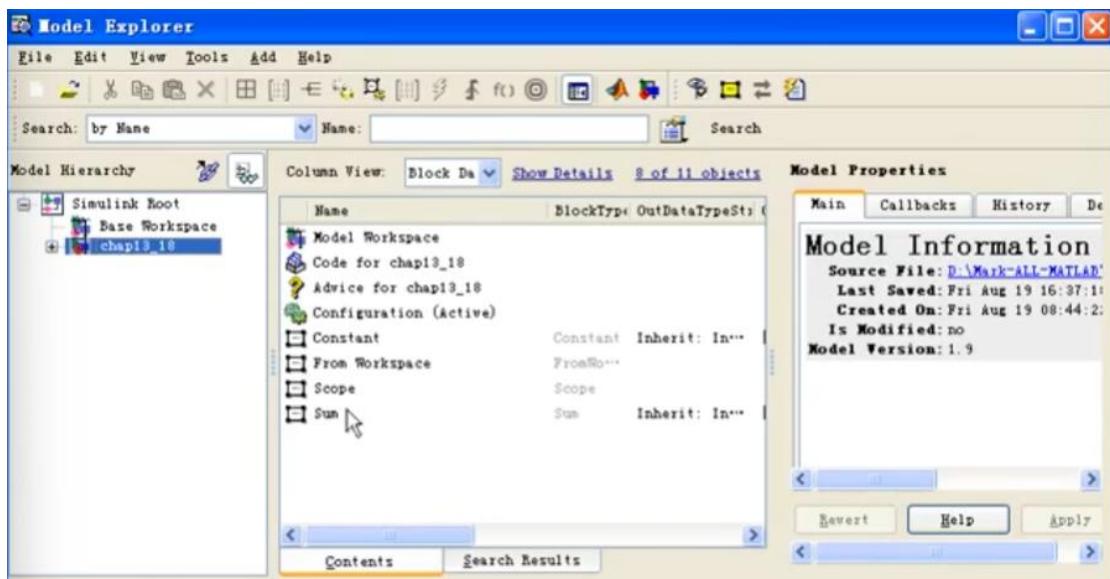
子系统可以使设计的模型非常简洁易懂，但是设置各个模块的参数仍然非常繁琐。通过对子系统进行封装，使用户创建的子系统和Simulink的其它模块一样有自己的图标，用鼠标双击该子系统后会出现用户自定义的属性设置窗口，对子系统中各个模块的参数进行设置。



## Simulink 模型工作空间

每个Simulink模型有两个工作空间，一个是基本的MATLAB工作空间，另一个是模型的工作空间。对于模型工作空间中的变量只有该模型能够看到。通常模型工作空间中的变量不能改变。





## 本章小结

本章详细介绍了利用**Simulink**建模的基本知识，主要包括**Simulink**建模的基本概念、基本操作、常用的模块库，以及子系统和封装技术等。对于建模时常用的模块，以及这些模块的属性设置，需要用户熟练掌握。

对于仿真的结果，可以采用示波器等进行查看。用户通过对仿真结果进行分析，从而对模型或模块的参数进行修改。用户在利用**Simulink**建模时，可以首先尝试建立比较简单的模型，然后逐步建立稍微复杂一些的模型。

# Simulink 建模和 S-函数

Simulink的功能越来越强大，应用领域越来越多。本章将更加深入的介绍利用Simulink进行建模和仿真。首先详细的介绍Simulink建模中的回调函数，利用回调函数在模块或模型发生变化时执行某些特定的操作，非常的方便。接着介绍了Simulink仿真器的设置以及在命令行下进行模型的仿真，命令行下模型的仿真，可以非常方便的修改模块或模型的参数，在查看在不同参数下模型的结果非常的方便。然后介绍了Simulink中的调试器，如果建立的模型没有实现预期的功能，可以利用调试器来进行模型的调试。最后介绍了S-函数的基本概念、工作原理，并详细的介绍了如何利用M文件编写S-函数。利用S-函数可以扩展Simulink的功能，例如采用MATLAB、C/C++或Fortan等语言编写模块。

## 回调函数

在Simulink中，可以通过回调函数实现对模型或模块的操作。回调函数是当用户建立的模型或模块发生某些特殊行为时所要执行的MATLAB程序。下面分别介绍模型的回调函数和模块的回调函数。

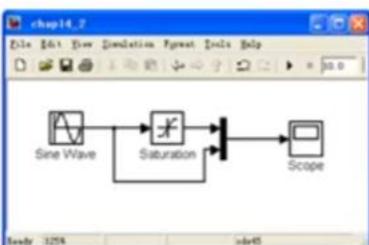
### 模型回调函数

模型文件的回调函数（Callback）可以在模型文件的属性窗口中进行设置。建立模型（文件chap14\_1.mdl），如图14.1所示。增益模块Gain的参数为2.0，其他模块采用默认参数。对于该模型文件，单击File菜单下的Model Properties子菜单，得到模型属性窗口，然后单击Callbacks选项，如图14.2所示。在Model Callbacks窗口中，选中InitFcn，然后在右侧的文本框中输入程序，在MATLAB的基本工作空间中建立变量simin。变量simin作为From Workspace模块的输入变量。模型文件的回调函数改变后，函数名后面会添加一个星号（\*）。



## 模块回调函数

与模型文件的回调函数类似，模块回调函数在模块的模块属性（**Block Properties**）窗口中的**Callbacks**内输入。建立模型文件（文件名为chap14\_2.mdl），如图14.5所示，**Sine Wave**模块和**Saturation**模块都采样默认的属性值。用鼠标单击**Sine Wave**模块，然后选择**Edit**菜单下的**Block Properties**子菜单，会出现**Sine Wave**模块的模块属性窗口如图14.6所示。



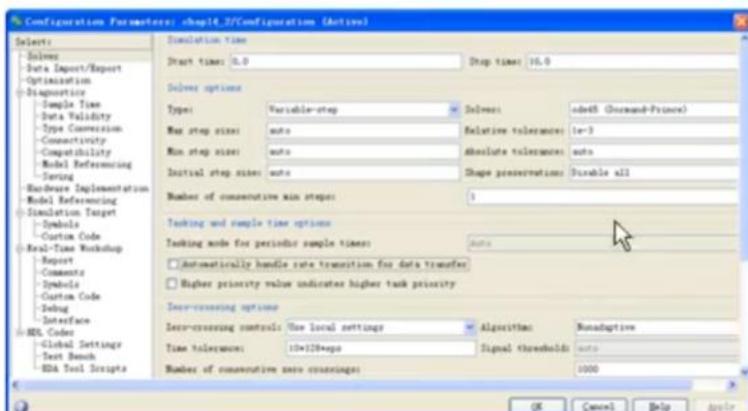
//回调函数注意理解!!!

## 运行仿真

模型建立后，就可以运行该模型和对仿真结果进行分析。用户可以单击菜单**Simulation**下的**Start**子菜单运行该模型。下面介绍如何设置仿真参数，以及仿真出错时的信息。最后介绍如何在MATLAB的命令行窗口进行模型的仿真。

## 仿真参数的设置

用户可以选择模型文件中的菜单**Simulation**下的**Configuration Parameters**子菜单，得到仿真参数设置窗口，如图14.9所示。下面只介绍其中的**Solver**和**Data Import/Export**，对于其他的参数设置，用户可以查阅MATLAB的帮助系统。



## 仿真的出错信息

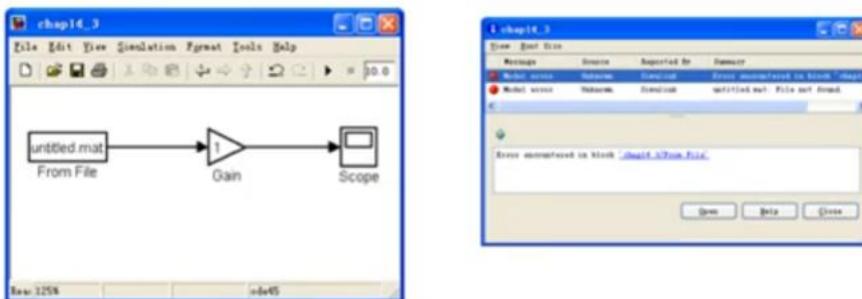
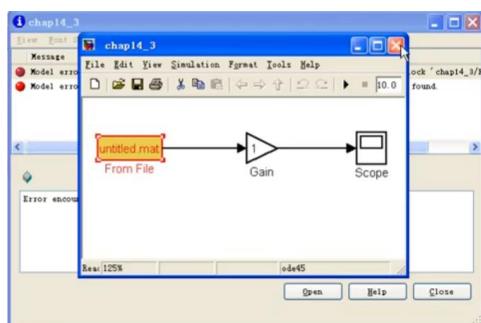


图14.14 模型的出错例子 图14.15 Simulink出错信息

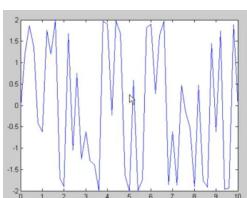
如果在模型的运行时，出现错误，用户需要认真阅读出错信息，查找出错的模块，然后进行模型的修改。对于本模型，需要在模型文件所在的文件夹中建立包含仿真数据的untitled.mat文件。



## 使用 MATLAB 命令行运行仿真

在利用模型窗口直接进行仿真时，每次运行时都是针对固定的模块参数。在运行过程中很难改变这些参数。在 MATLAB 中，对于建立的 Simulink 模型允许通过函数 sim( ) 在命令行进行仿真。这样在进行仿真时，可以很容易的改变模块的参数。

```
clear;
simOut=sim('chap14_4','AbsTol', '1e-5',...
    'SaveOutput', 'on', 'OutputSaveName', 'youtNew');
simOutVars=simOut.who;
tout=simOut.get('tout');
yout=simOut.get('youtNew');
figure;
plot(tout,yout);
```



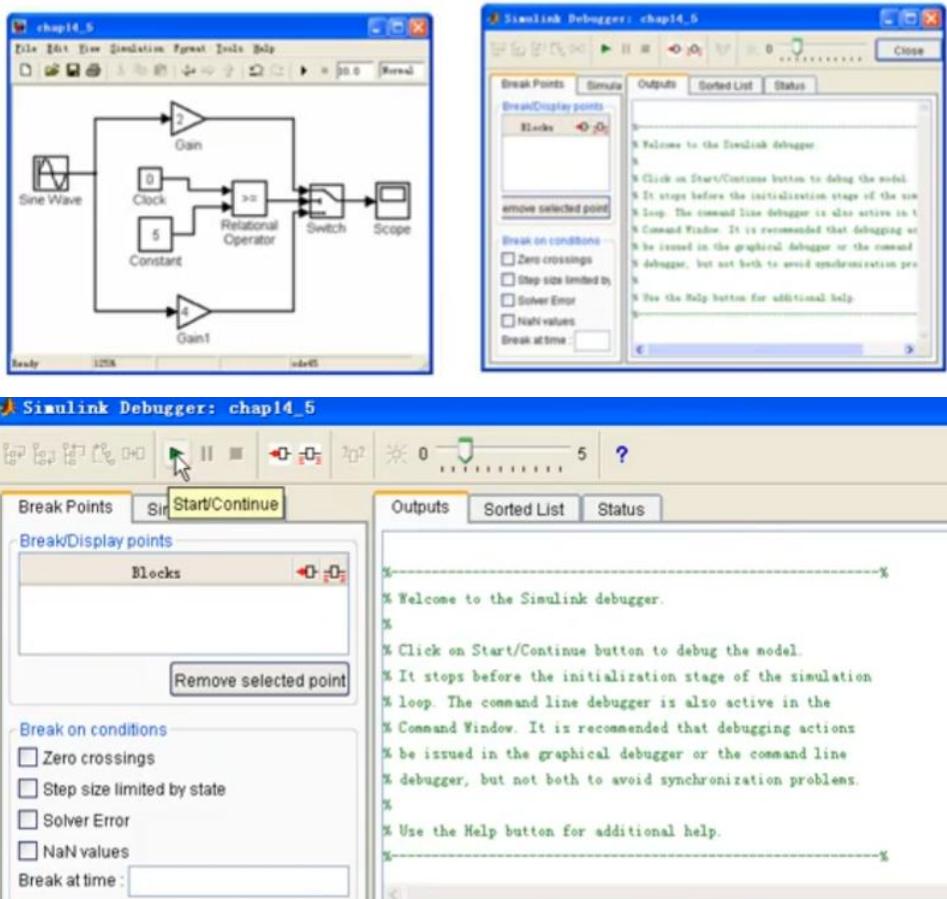
## 模型的调试

在Simulink中，可以通过调试器对模型进行调试，允许用户通过单步执行，来观察模块的状态和输入输出等。用户既可以采用图形化的调试界面进行模型的调试，也可以通过命令进行调试。下面介绍Simulink中模型的调试。

### Simulink 调试器

建立模型文件（文件名为chap14\_5.mdl），如图14.19所示

。模块Sine Wave的角频率为5，其他参数为默认值。然后对其他模块的参数进行了设置，采样示波器Scope显示输出结果。单击Tools菜单下的Simulink Debugger子菜单，打开Simulink的调试器，如图14.20所示。



## 命令行调试

在MATLAB中，还可以在命令行窗口中通过命令对Simulink模型进行调试。可以通过函数simget( )获取模型文件的设置信息，通过函数simset( )对模型的参数进行设置。这两个函数的详细调用格式，用户可以查阅MATLAB的帮助系统。有两种方法启动调试器，对模型进行调试：一种是在利用函数sim( )对模型进行仿真时，通过函数simset( )将debug属性设置为on。

```
//quit 退出
```

## S-函数建模

**S-函数（System-函数）**是扩展Simulink功能的非常好的工具，在很多情况下非常有用。用户可以利用MATLAB语言、C语言、C++语言以及Fortan等语言创建自定义的Simulink模块。在Simulink基本模块库中的用户定义模块子集中有多个和S-函数有关的模块。

### S-函数介绍

S-函数是对一个动态系统的计算机程序语言描述。利用C/C++、Fortan等语言写的S-函数需要用mex命令编译成MEX文件，然后就可以动态的连接到MATLAB了。本文只介绍采样MATLAB语言编写S-函数。

S-函数使用一种特殊的调用规则使用户可以与Simulink的内部求解器进行交互，和内部模块的调用非常相似。S-函数可以用于连续系统、离散系统和混合系统，几乎所有的模块都可以描述为S-函数。

通过Simulink基本模块库中的用户定义模块子集中的S-Function模块，可以将S-函数加入到Simulink模型中，并指定S-函数的名字。

### S-函数工作原理

通过S-函数可以创建一个自定义的Simulink模块，在模型中可以多次使用这个模块，使用时只需要改变它的参数即可。

S-函数是由一系列S-函数回调方法组成，在每次仿真循环中的每一个仿真步骤由Simulink调用相应的S-函数回调方法来执行相应的任务。

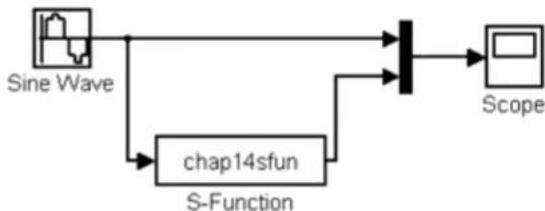
## M文件的S-函数

Simulink为编写S-函数提供了各种模版文件，其中定义了S-函数完整的框架结构，用户可以根据自己的需要在模版上进行修改。利用M文件编写S-函数时，推荐采用S-函数的模版文件sfuntmpl.m。用户可以在MATLAB的命令行窗口输入：edit sfuntmpl.m打开该文件。这个模版文件包含1个主函数和6个子函数，在主函数内通过标志变量flag，通过switch...case语句根据flag执行对应的子函数。flag变量作为主函数的参数由系统调用时给出。

```
switch flag,  
    %%%%%%  
    % Initialization %  
    %%%%%%  
    case 0,  
        [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;  
  
    %%%%%%  
    % Derivatives %  
    %%%%%%  
    case 1,  
        sys=mdlDerivatives(t,x,u);  
  
    %%%%%%  
    % Update %  
    %%%%%%  
    case 2,  
        sys=mdlUpdate(t,x,u);  
  
    %%%%%%  
    % Outputs %  
    %%%%%%  
    case 3,  
        sys=mdlOutputs(t,x,u);  
    end //模板函数
```

## S-函数实例分析

在MATLAB中，提供了S-函数的例子程序，用户可以在命令行窗口输入：sfundemos，来查看例子程序，如图14.23所示。S-函数的例子程序包括M文件例子、C文件例子、C++文件例子和Fortan例子。用鼠标双击M-files，得到M文件编写的S-函数例子，如图14.24所示。其中Level-1 M-files用来创建的S-函数用于兼容以前版本的S-函数，Level-2 M-files用于扩展M文件的S-函数。



```

function [sys,x0,str,ts,simStateCompliance] = chap14sfun(t,x,u,flag)
switch flag,
    case 0,
        [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 2,
        sys=mdlUpdate(t,x,u);           %
    case 3,
        sys=mdlOutputs(t,x,u);
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys=mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end

function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;          %一个输出端口
sizes.NumInputs = 1;           %一个输入端口
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];

str = [];
ts = [-1 0];                  %采样率
simStateCompliance = 'UnknownSimState';

function sys=mdlDerivatives(t,x,u)
sys = [];

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u)      %原信号乘以2，然后再加1
sys = 2 * u + 1;

function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;

function sys=mdlTerminate(t,x,u)
sys = [];

```

## 本章小结

本章首先介绍了**Simulink**中的回调函数，包括模块和模型的回调函数。然后详细的介绍了**Simulink**仿真器的设置、命令行下模型的仿真和**Simulink**调试器等。最后介绍了**S-函数**的概念和基本原理，并重点讲解了如何利用**M**文件建立**S-函数**，给出了详细的实例。

# GUI 编程开发

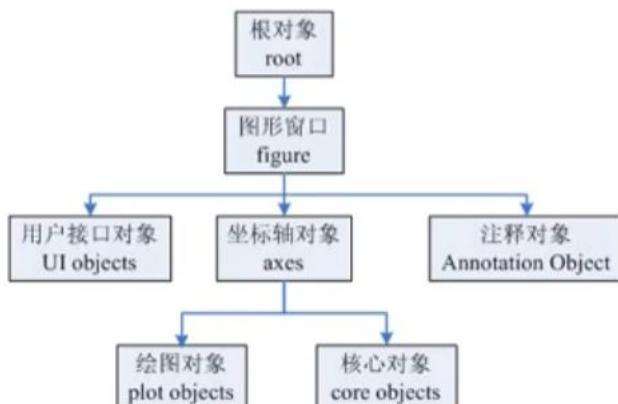
在本章首先详细的介绍了MATLAB的图形句柄，通过对图形对象的属性进行设置，可以是实现图形的底层控制和设置。然后介绍了用户接口对象，用于建立各种按钮、菜单和工具条等。最后介绍了常用的对话框，这些对话框是GUI编程常用的基本元件。

## 图形句柄

句柄图形（Handle Graphics）是一种面向对象的绘图系统。这些细节一般隐藏在图形M文件的内部。用户通过句柄图形可以定制图形的许多特性，这是使用高级绘图函数无法实现的。利用句柄图形可以创建图形用户界面。

## MATLAB 图形系统

在MATLAB中，各种句柄图形对象是有层次的，按照父对象（Parent）和子对象（Children）的形式进行管理。当子对象创建时，继承了父对象的许多属性。



## 图形句柄

MATLAB在创建每一个图形对象时，都为该对象分配唯一的一个值，称其为图形对象句柄(Handle)。句柄是图形对象的唯一标识符，不同对象的句柄不可能重复和混淆。

计算机屏幕作为根对象（root）由系统自动建立，其句柄值为0，而图形窗口对象（figure）的句柄值为一正整数，并显示在该窗口的标题栏。其他图形对象的句柄为浮点数。

MATLAB在创建各种对象时，会产生该对象的句柄，通过句柄可是实现对该对象的各种控制和设置。MATLAB提供了若干个函数用于获取已有图形对象的句柄。

## 图形对象的属性

图形对象的属性是一些特殊值，可以对图形对象进行控制和设置。每个属性都有一个属性名和属性值。属性名通常采用大小写字母组成的字符串，第一个字母大写。

用户可以在创建图形对象时，可以对属性值进行设置。如果用户没有对属性值进行设置，则所有的属性都会自动初始化为系统的默认值。例如：

`figure('Toolbar','none','Menubar','none')`，将创建没有工具栏和菜单的图形窗口。

图形对象的属性非常多，通常在创建图形对象后，通过句柄对属性值进行修改。利用函数`get()`获取图形对象的属性值，通过函数`set()`设置图形对象的属性值。图形对象的属性设置，将会在下一节详细介绍。



## 图形对象

### 创建图形对象

除根对象外，每一种图形对象都有一个相对应的创建函数。

在MATLAB中，图形对象的创建函数，如表15.1所示。

例如，采用函数`figure()`创建`figure`对象，函数`axes()`创建坐标轴对象。

表 15.1 图形对象的创建函数

| 函数                   | 说明  | 函数                     | 说明   |
|----------------------|-----|------------------------|------|
| <code>root</code>    | 根对象 | <code>figure</code>    | 图形窗口 |
| <code>axes</code>    | 坐标轴 | <code>image</code>     | 图像   |
| <code>light</code>   | 灯光  | <code>line</code>      | 线条   |
| <code>patch</code>   | 块   | <code>rectangle</code> | 矩形   |
| <code>surface</code> | 曲面  | <code>text</code>      | 文本   |

## 获取对象的属性

在MATLAB中，利用函数get( )来获取对象的属性值。该函数的调用格式为：

v=get(h): 该函数获取句柄h的所有属性值。

v=get(h, 'PropertyName'): 该函数获取句柄h的指定的属性值。

v=get(0, 'Factory'): 该函数获取根对象 (root) 的所有出厂属性值。

v=get(h, 'default'): 该函数获取句柄h的所有默认属性值。

## 设置对象的属性

在MATLAB中，利用函数set( )设置对象的属性。该函数的调用格式为：

set(h, 'PropertyName', PropertyValue): 该函数设置对象h的属性PropertyName的值为PropertyValue。

set(h, 'PropertyName1', PropertyValue1, 'PropertyName2', PropertyValue2, ...): 该函数同时对h的多个属性值进行设置。

A=set(h, 'PropertyName')或set(h, 'PropertyName'): 该函数获取对象h的属性PropertyName的可设置值。

A=set(h)或set(h): 该函数获取对象h的所有可设置的属性值。

## 对象的基本操作

在MATLAB中，有一些特殊的函数可以获取当前的图形对象、坐标轴对象的句柄值，这些函数如表15.2所示。利用这些函数可以非常方便的获取句柄值。

表 15.2 获取当前图形对象的函数

| 函数   | 说明                      |
|------|-------------------------|
| gcf  | 获取当前图形窗口的句柄             |
| gca  | 获取当前图形窗口中的坐标轴的句柄        |
| gco  | 获取当前图形窗口中当前对象的句柄        |
| gcbf | 获取正在执行的回调程序对应的对象所在窗口的句柄 |
| gcbo | 获取正在执行的回调程序对应的对象句柄      |

在MATLAB中，除了可以采用函数get( )和set( )获取和设置图形对象的属性，还可以对图形对象进行一切其他的操作，例如利用函数findobj( )进行对象的查找等。对象的基本操作，如表15.3所示。下面分别进行介绍。

表 15.3 对象的操作常用作用

| 函数       | 说明             |
|----------|----------------|
| reset    | 对象的复位，恢复为默认值   |
| findobj  | 对象的查找          |
| findAll  | 查找所有对象，包括隐藏的对象 |
| copyobj  | 对象的复制          |
| delete   | 对象的删除          |
| allchild | 查找所有的子对象       |
| ancestor | 查找对象的父对象       |

```
>> h=figure(1);
>> get(h, 'resize')

ans =

on      []

>> set(h, 'resize', 'off')
>> reset(gcf)
>> get(h, 'resize')

ans =

on
```

## root 根对象

在MATLAB中，根对象（root）相当于计算机的屏幕。只有一个根对象，是figure对象的父对象。根对象的句柄值为0，没有父对象。根对象不需要创建，在MATLAB软件启动后自动创建。根对象也不能删除，在退出MATLAB软件后，自动删除。用户可以通过get(0)获取根对象的所有属性。

## figure 对象

图形窗口也称为**figure**对象。图形窗口的句柄值为正整数。

在MATLAB中，采用函数**figure( )**来创建图形窗口。

在MATLAB中，利用函数**figure(h)**创建句柄为h的图形窗口

后，可以采用**get(h)**获取该图形句柄的所有属性值。采用  
函数**clf**清空当前图形窗口。该函数的调用格式为：

**clf**: 该函数清除当前**figure**中的所有不隐藏的对象

**clf('reset')**: 该函数清除当前**figure**的所有对象，并对  
**figure**的属性复位为系统的默认值。

**clf(fig)**: 该函数清除窗口**fig**中所有不隐藏的对象。

**clf(fig, 'reset')**: 该函数清除窗口**fig**中所有不隐藏的对象，  
并复位为系统的默认值。

```
>> close all;
>> figure; plot(1:20);
>> clf
>> get(1)
```

## axes 坐标轴对象

在MATLAB中，采用函数**axes( )**创建坐标轴对象，该函数  
的调用格式为：

**axes**: 该函数在当前的**figure**内创建坐标轴对象，采用系统  
的默认值。 

**axes('position', RECT)**: 该函数设置坐标轴的位置为  
RECT，RECT为4维向量，指定坐标轴的尺寸和位置，  
格式为：[left, bottom, width, height]，即[左，底，宽  
，高]，以屏幕的左下角为原点。

**axes(h)**: 当句柄为h的坐标轴对象存在时，将其设置为当前  
对象。如果句柄为h的坐标轴对象不存在，则创建句柄为  
h的坐标轴对象，并设置为当前的对象。

**h=axes(...)**: 该函数返回坐标轴对象的句柄h。

```
>> clear all;
>> figure;
>> h=axes;
>> set(h, 'Color', [1 0 0]);
>> get(h, 'Color')
```

ans =

1 0 0

## 核心图形对象

在MATLAB中，核心图形包括：图像（image）、线条（line）、文本（text）、光照（light）、块（patch）、矩形（rectangle）和曲面（surface）。下面对常用的image、line和text进行介绍。对于light、patch、rectangle和surface用户可以通过MATLAB的帮助进行查看。

在MATLAB中，利用函数image( )创建图像对象，该函数的低级调用格式为：`h=image('P1Name', P1Value, 'P2Name', P2Value, ...)`。image对象的主要属性为CData和Visible。CData属性保存了image的图像数据，Visible属性设置图像是否显示。

```
clear all;
close all;
A=imread('football.tif','tif');           %读取图像文件
A=im2double(A);                            %图像数据转换为double类型
h=image(A);                                %显示图像
set(gca,'xlim',[0 320],'ylim',[0 256]);    %设置坐标轴的属性
get(h,'type')                             %获取图形对象的属性值
```

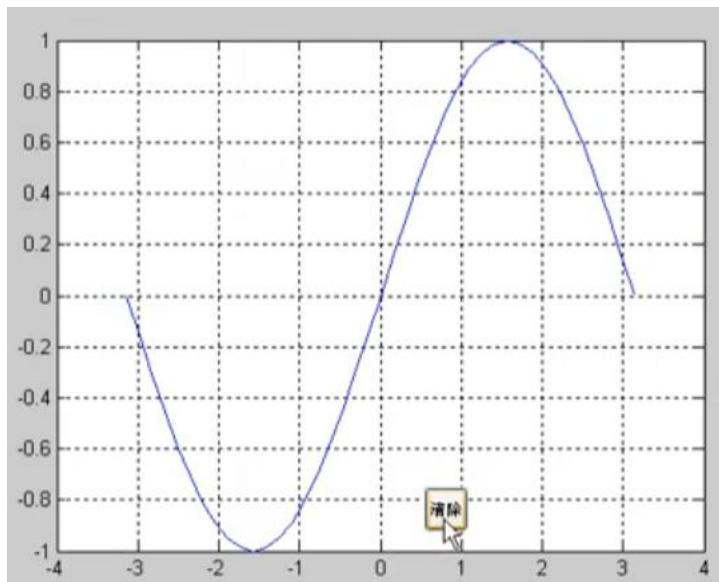
## 用户接口对象

在MATLAB中，用户接口对象（User Interface Objects）主要包括uicontrol对象、uimenu对象、uicontextmenu对象、uitoolbar对象和uitable对象等。下面分别进行介绍。

### uicontrol 对象

利用函数uicontrol( )建立按钮，单击该按钮后，清除figure中的图形。

```
h=figure;
t=-pi:pi/20:pi;
plot(t,sin(t));                           %绘制正弦曲线
grid on;
h=uicontrol('Style','pushbutton','String','清除',...
'Position',[320 60 30 30],'Callback','cla');
```



//要的就是这效果!!!

## uimenu 对象

在MATLAB中，利用函数uimenu( )创建自定义菜单和菜单选项，该函数的调用格式为：

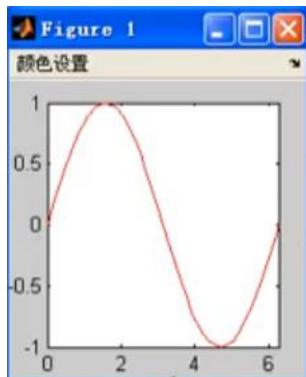
**uimenu('P1Name', P1Value, ...):** 该函数设置菜单的属性 P1Name 的值为 P1Value。

**uimenu(h, ...):** 该函数以 h 为父对象创建菜单。

**h=uimenu(...):** 该函数返回菜单的句柄 h。

利用函数uimenu( )创建菜单，并设置快捷键，代码如下：

```
clear all; close all;
t=0:pi/20:2*pi;
figure('menubar','none'); % 创建图形窗口
h1=plot(t,sin(t));
set(gca,'xlim',[0 2*pi]);
h2=uimenu('Label','颜色设置'); % 创建菜单
uimenu(h2,'Label','红色','Accelerator','R',...
    'callback','set(h1,"Color","red")');
uimenu(h2,'Label','绿色','Accelerator','G',...
    'callback','set(h1,"Color","green")');
uimenu(h2,'Label','蓝色','Accelerator','B',...
    'callback','set(h1,"Color","blue")');
set(gcf,'position',[100,100,200,200]);
```



## uicontextmenu 对象

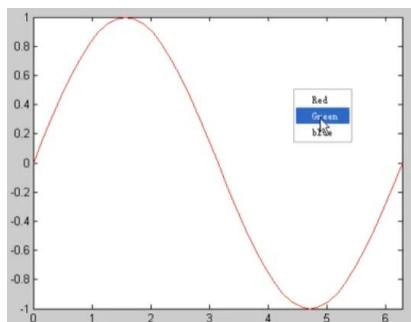
在MATLAB中，利用函数uicontextmenu( )创建右键快捷菜单，该函数的调用格式为：

**uicontextmenu('P1Name', P1Value, ...):** 该函数对右键快捷菜单的属性进行设置。

**h=uicontextmenu(...):** 该函数返回右键快捷菜单的句柄h。

创建快捷菜单，对正弦曲线的颜色进行设置，代码如下：

```
clear all; close all;
t=0:pi/20:2*pi;
figure('menubar','none');      %建立图形窗口
h1=plot(t,sin(t));
set(gca,'xlim',[0 2*pi]);
h2=uicontextmenu;            %创建右键快捷菜单
c1='set(h1,"color","red")';
c2='set(h1,"color","green")';
c3='set(h1,"color","blue")';
uimenu(h2,'Label','Red',...
    'callback',c1);
uimenu(h2,'Label','Green',...
    'callback',c2);
uimenu(h2,'Label','blue',...
    'callback',c3);
set(gca,'uicontextmenu',h2);
```



//右键快捷键

## uitoolbar 对象

在MATLAB中，利用函数`uitoolbar( )`为图形对象创建工具栏，

利用函数`uipushtool( )`和`uitoggletool( )`为工具栏创建按钮。

`uitoolbar`对象是`uipushtool`对象和`uitoggletool`对象的父对象

。函数`uitoolbar( )`的调用格式为：

`uitoolbar('P1Name', P1Value, ...)`：该函数设置工具栏的属性。

`uitoolbar(h, ...)`：该函数为句柄为`h`的图形对象设置工具栏。

取消图形窗口的标准工具栏，创建新的工具栏：

```
clear all; close all;
```

```
h1=figure('toolbar','none'); % 创建图形窗口
```

```
h2=uitoolbar; % 创建工具栏
```

## uibuttongroup 对象

在MATLAB中，利用函数`uibuttongroup( )`创建按钮组，该

函数的调用格式为：

`uibuttongroup('P1Name', P1Value, ...)`：该函数对按钮组的属性进行设置。

`h=uibuttongroup(...)`：该函数返回按钮组的句柄`h`。

如果`h`为按钮组的句柄，通过函数`get(h)`可以获取按钮组的

所有属性值。利用函数`uibuttongroup( )`创建了按钮组后，可以在按钮组中添加各种类型的按钮，并设置按钮的父对象为按钮组。

## uipanel 对象

在MATLAB中，可以采用函数`uipanel( )`创建面板，该函数的调用格式为：

`uipanel('P1Name', P1Value, ...)`：该函数对面板的属性进行设置。

`h=uipanel(...)`：该函数返回面板的句柄为`h`。

如果`h`为面板的句柄，则通过函数`get(h)`可以获取面板的所

有属性值。利用函数`uipanel( )`创建面板后，用户可以在面板中添加按钮或文本等。

## uitable 对象

在MATLAB中，利用函数uitable( )创建表格对象，该函数的调用格式为：

uitable('P1Name', P1Value, ...): 利用函数uitable创建表格对象，并对属性进行设置。

h=uitable(...): 该函数返回表格对象的句柄为h。

创建表格对象，并对数据进行显示：

```
clear all; close all;  
X=rand(10,3);  
figure;  
h=uitable('Data',X); %创建uitable对象  
set(h,'Units','normalized'); %设置属性  
set(h,'Position',[0.1 0.1 0.8 0.8]); %设置属性
```

|    | 1      | 2      | 3      |
|----|--------|--------|--------|
| 1  | 0.7060 | 0.4387 | 0.2760 |
| 2  | 0.0318 | 0.3816 | 0.6797 |
| 3  | 0.2769 | 0.7655 | 0.6551 |
| 4  | 0.0462 | 0.7952 | 0.1626 |
| 5  | 0.0971 | 0.1869 | 0.1190 |
| 6  | 0.8235 | 0.4898 | 0.4984 |
| 7  | 0.6948 | 0.4456 | 0.9597 |
| 8  | 0.3171 | 0.6463 | 0.3404 |
| 9  | 0.9502 | 0.7094 | 0.5853 |
| 10 | 0.0344 | 0.7547 | 0.2238 |

## 常用的对话框

在MATLAB中，有很多的预定义对话框，进行用户和计算机之间的交互。在进行GUI编程时，经常会用到这些预定义对话框，使用也非常的简单和方便。MATLAB中的预定义对话框如表15.5所示。下面分别进行介绍。

表 15.5 常用的预定义对话框<sup>①</sup>

| 函数 <sup>②</sup>           | 说 明 <sup>③</sup>     | 函数 <sup>②</sup>         | 说 明 <sup>③</sup>           |
|---------------------------|----------------------|-------------------------|----------------------------|
| dialog <sup>④</sup>       | 创建对话框 <sup>⑤</sup>   | msgbox <sup>④</sup>     | 信息显示对话框 <sup>⑤</sup>       |
| errordlg <sup>④</sup>     | 错误对话框 <sup>⑤</sup>   | uoload <sup>④</sup>     | 加载数据文件的标准图形界面 <sup>⑤</sup> |
| warndlg <sup>④</sup>      | 警告对话框 <sup>⑤</sup>   | uisave <sup>④</sup>     | 保存数据文件的标准图形界面 <sup>⑤</sup> |
| helpdlg <sup>④</sup>      | 帮助对话框 <sup>⑤</sup>   | uigetdir <sup>④</sup>   | 选取目录的标准图形界面 <sup>⑤</sup>   |
| inputdlg <sup>④</sup>     | 输入对话框 <sup>⑤</sup>   | uigetfile <sup>④</sup>  | 选取文件的标准图形界面 <sup>⑤</sup>   |
| listdlg <sup>④</sup>      | 列表对话框 <sup>⑤</sup>   | uiputfile <sup>④</sup>  | 保存文件的标准图形界面 <sup>⑤</sup>   |
| printdlg <sup>④</sup>     | 打印对话框 <sup>⑤</sup>   | uisetcolor <sup>④</sup> | 设置色彩的标准图形界面 <sup>⑤</sup>   |
| pagesetupdlg <sup>④</sup> | 页面设置对话框 <sup>⑤</sup> | uisetfont <sup>④</sup>  | 设置字体的标准图形界面 <sup>⑤</sup>   |
| questdlg <sup>④</sup>     | 询问对话框 <sup>⑤</sup>   | waitbar <sup>④</sup>    | 显示包含进度条的对话框 <sup>⑤</sup>   |

## 消息对话框

在MATLAB中，利用函数**msgbox( )**建立消息对话框。它不接受用户的任何输入，在用户单击OK按钮后，对话框自动关闭，然后返回程序中继续执行。该函数的调用格式为：

**msgbox('Message')**：该函数建立消息对话框，消息的内容为Message。

**msgbox('Message', 'Title')**：该函数建立消息对话框的标题为Title。

**msgbox('Message', 'Title', 'Icon')**：该函数建立消息对话框的Icon，Icon可以为**none**、**error**、**help**、**warn**或**custom**等，系统默认为**none**。

**msgbox('Message', 'Title', 'custom', IconData, IconMap)**：该函数中利用**IconData**和**IconMap**建立Icon。

**h=msgbox(...)**：该函数返回消息对话框的句柄。

```
>> h=msgbox('大家好！','标题');
>> h=msgbox('大家好！','标题','error');
>> h=msgbox('大家好！','标题','help');
>> h
h =
0.0151
```

## 错误对话框

在MATLAB中，利用函数**errordlg( )**建立错误对话框。它不接受用户的任何输入，在用户单击OK按钮后，对话框自动关闭，然后返回程序中继续执行。该函数的调用格式为：

**errordlg**：建立默认的错误对话框。

**errordlg('Errorstring')**：该函数建立错误对话框的内容为Errorstring。

**errordlg('Errorstring', 'Dialogname')**：该函数建立错误对话框的名称为Dialogname。

**h=errordlg(...)**：该函数返回错误对话框的句柄。

## 警告对话框

在MATLAB中，采用函数**warndlg( )**建立警告对话框，该函数的调用格式和**errordlg( )**基本相同。用户可以在MATLAB的命令行窗口输入：**help warndlg**进行查询。

```
warndlg;  
warndlg('警告信息! ','警告');
```

## 帮助对话框

在MATLAB中，采用函数**helpdlg( )**建立帮助对话框。该函数的调用格式和**errordlg( )**基本相同。用户可以在MATLAB的命令行窗口输入：**help helpdlg**进行查询。

## 输入对话框

在MATLAB中，利用函数**inputdlg( )**建立输入对话框。该函数接受用户的输入，然后将输入值进行返回。

```
answer=inputdlg({'UserName:','Password:'},'log in');  
带有默认值的输入对话框:  
clear all; close all;  
prompt={'输入矩阵的行: ','输入矩阵的列: '};  
name='矩阵大小';  
numlines=1;  
defaultanswer={'5','5'};  
answer=inputdlg(prompt,name,numlines,defaultanswer);
```

```
>> answer=inputdlg({'UserName:','Password:'},'log in');  
>> answer  
  
answer =  
  
    'robinson'  
    '123456'  
  
>> clear all; close all;  
prompt={' 输入矩阵的行: ',' 输入矩阵的列: '};  
name=' 矩阵大小';  
numlines=1;  
defaultanswer={' 5',' 5'};  
answer=inputdlg(prompt, name, numlines, defaultanswer);  
>> answer  
  
answer =  
  
    '7'
```

## 列表对话框

在MATLAB中，利用函数listdlg( )建立列表对话框，该函数的调用格式为：

[Selection, ok]=listdlg('P1Name', 'P1Value'): 该函数设置列表框的属性，用户单击按钮Ok后，选择的序号保存在Selection，参数ok为1。

利用函数listdlg( )创建列表对话框：

```
[Selection,ok]=listdlg('ListString',{'Blue','Green','Yello','Black'},... %创建列表框  
'name','Choose Color',...  
'OKString','确定',...  
'CancelString','取消',...  
'InitialValue',3,...  
'SelectionMode','Single',....  
'ListSize',[200 100]);
```



## 问题对话框

在MATLAB中，利用函数questdlg( )建立问题对话框，然后等待用户回答。默认情况下，回答按钮有3个，分别是Yes、No和Cancel。

```
choice=questdlg('你确定要关闭吗? ',... %问题对话框  
'确认',...  
'No');  
switch choice,  
case 'Yes'  
    disp('You choose Yes.');  
case 'No'  
    disp('You choose No.');//  
case 'Cancel'  
    disp('You choose Cancel');//  
end
```



## 进度条设置对话框

在MATLAB中，利用函数**waitbar( )**创建进度条对话框，反映程序运行的完成情况。该函数的调用格式为：

**waitbar(x)**: 创建进度条，x介于[0 1]之间，如果x=0表示进度条开始，x=1时进度完成。

**h=waitbar(x, 'message')**: 创建标题为**message**的进度条，返回值h为进度条的句柄。

**h=waitbar(x, 'message', 'P1Name', P1Value, ...)**: 对进度条的属性进行设置。

**waitbar(x, h)**: 更新句柄为h的进度条的进度x。

**waitbar(x, h, 'updated message')**: 更新句柄为h的进度条的进度和标题。

创建进度条，显示完成情况，代码如下：

```
h=waitbar(0,'Please wait...','Name','进度条');  
    创建进度条
```

```
steps=1000;
```

```
for i=1:steps
```

```
    waitbar(i/steps,h,['Have finished...'   
    num2str(100*i/steps) '%']); %更新进度  
    pause(0.01);
```

```
end
```

```
close(h);
```

## 路径选择对话框

在MATLAB中，采用函数`uigetdir()`建立路径选择对话框，返回目录名称字符串。

`folder_name=uigetdir`: 该函数创建目录选择对话框。如果用户单击OK按钮，返回路径字符串，如果用户单击Cancel按钮，返回值为0。

`folder_name=uigetdir('start_path')`: 该函数中起始目录为`start_path`。

`folder_name=uigetdir('start_path', 'dialog_title')`: 该函数中起始目录为`start_path`，对话框的标题为`dialog_title`。

建立默认的路径选择对话框，代码如下：

```
dirname=uigetdir;
```

## 文件打开对话框

在MATLAB中，采用函数`uigetfile()`建立文件打开对话框，用户通过该对话框来选择要打开的文件。该函数返回用户选择的路径和文件名，然后可以进行数据的读取和写入等操作。

`Filename=uigetfile`: 该函数建立默认的文件打开对话框，默認為当前的目录，文件类型为所有的MATLAB文件，返回所选择的文件名。

`[Filename, Pathname]=uigetfile(FilterSpec)`: 该函数通过`FilterSpec`指定扩展名，返回选择的文件名和路径。  
`FilterSpec`为字符串或字符串单元数组，用于文件名的过滤。如果`FilterSpec`为`*.m`，则在对话框中只列出后缀为`.m`的文件。

`[Filename, Pathname]=uigetfile(FilterSpec, DialogTitle)`：该函数设置对话框的标题为`DialogTitle`。

```
>> [filename, pathname]=uigetfile  
filename =  
chap15_1.m  
  
pathname =  
D:\Mark-ALL-MATLAB\chap15\
```

## 文件保存对话框

在MATLAB中，采用函数uiputfile( )建立文件保存对话框，  
返回保存的文件名和路径名。

**Filename=uiputfile:** 该函数建立默认的文件保存对话框，  
默认为当前的目录，文件类型为所有的MATLAB文件，  
返回保存的文件名。

**[Filename, Pathname]=uiputfile:** 该函数建立默认的文件  
保存对话框，返回保存的文件名和路径。

**[Filename, Pathname]=uiputfile(FilterSpec):** 该函数通  
过FilterSpec指定文件的扩展名。FilterSpec为字符串或  
字符串单元数组，用于文件名的过滤。

**[Filename, Pathname]=uiputfile(FilterSpec, DialogTitle)**  
：该函数设置对话框的标题为DialogTitle。

## 页面设置对话框

在MATLAB中，采用函数pagesetupdlg( )建立页面设置对  
话框，该函数的调用格式为： dlg=pagesetupdlg(fig)，  
其中fig为图形窗口句柄。该函数只支持单一图形窗口的  
页面设置，参数fig也必须是单一的图形句柄，不能是句  
柄向量。

利用函数pagesetupdlg( )建立页面设置对话框，代码  
如下：

```
h=figure;  
dlg=pagesetupdlg(h); %创建页面设置对话框
```

## 打印预览对话框

在MATLAB中，采用函数printpreview( )建立打印预览对话框  
，该函数的调用格式为：

**printpreview:** 该函数建立当前窗口的打印预览对话框。如果没  
有图形窗口，则采用figure新建一个图形窗口。

**printpreview(f):** 该函数建立图形句柄为f的打印预览窗口。

```
t=0:pi/20:4*pi;  
h=figure;  
plot(t,sin(t),'r');  
set(gca,'xlim',[0 4*pi]);  
printpreview(h); %创建打印预览对话框
```

## 打印对话框

在MATLAB中，可以采用函数printdlg( )函数建立打印对话框，用户可以对打印机和各种打印参数进行设置。

利用函数printdlg( )建立Window系统的打印对话框：

```
h=figure;  
printdlg(h); %打印对话框
```



建立MATLAB的打印对话框：

```
h=figure;  
printdlg('-crossplatform',h); %打印对话框
```

## 颜色设置对话框

在MATLAB中，采用函数uisetcolor( )建立颜色设置对话框，返回用户选择的颜色数据。该函数的调用格式为：

c=uisetcolor：该函数建立默认的颜色设置对话框。如果用户单击OK按钮，返回值c为用户选择的颜色。如果用户单击Cancel按钮，返回值为0。

c=uisetcolor([r g b])：该函数设置默认的颜色为[r g b]。

c=uisetcolor(..., 'DialogTitle')：该函数设置对话框的标题为DialogTitle。

利用函数uisetcolor( )建立颜色设置对话框：↓

```
c=uisetcolor([0 0 1],'Choose Color'); %颜色设置对话框
```

## 字体设置对话框

在MATALB中，采用函数uisetfont( )创建字体设置对话框，用户可以设置或修改字体。该函数的调用格式为：

uisetfont：该函数建立字体设置对话框。

uisetfont(h)：该函数对对象h的字体属性进行设置。

uisetfont(s)：该函数创建字体设置对话框，初始字体为结构体s。

uisetfont(..., 'DialogTitle')：该函数设置字体设置对话框的标题为DialogTitle。

s=uisetfont(...): 该函数将用户设置的字体保存在结构体s中。

利用函数uisetfont( )建立字体设置对话框，并对字体进行设置：

```
s=uisetfont;%字体设置对话框
```

## 本章小结

本章对MATLAB的图形句柄进行了详细的介绍，通过函数 `get( )` 来获取句柄的属性值，通过函数 `set( )` 来设置属性值。然后详细介绍了根对象（`root`）、图形窗口对象（`figure`）、坐标轴对象（`axes`）和核心图形对象等。最后详细的介绍了MATLAB的图形接口对象和常用的对话框，这些都是建立GUI界面必须的元件，读者需要熟练掌握、灵活应用。

# GUIDE 工具建立 GUI 界面

MATLAB可视化界面的设计，一般有两种方法，一是直接通过编辑M脚本文件产生GUI，这个方法在上一章已经进行详细的介绍。二是通过MATLAB图形用户界面开发环境GUIDE（Graphical User Interface Development Environment）来建立GUI界面。GUIDE是MATLAB为编写GUI界面提供的集成开发工具。本章将对GUIDE进行详细的介绍，包括常用的控件、控件的属性、自动产生的M文件等。然后通过一些例子程序，讲解如何使用这些控件。最后给出了两个综合一些的例子，介绍了如何建立更加实用的GUI界面。

## 采用 GUIDE 建立 GUI

GUI是通常包含图形窗口、坐标轴、菜单、按钮和文本等的用户界面。GUIDE提供了很多工具用于建立GUI界面。这些工具大大的简化了设计和建立GUI的过程。

### 图形用户界面概述

MATLAB为图形用户界面开发提供了一个方便高效的集成开发环境，GUIDE。GUIDE是一个界面设计工具集，提供了界面的外观、属性和回调函数等。GUIDE将用户保存好的GUI界面保存在一个FIG资源文件中，同时还能够产生包含GUI初始化和组件界面布局控制代码的M文件。这个M文件为实现回调函数提供了一个参考框架。

### GUIDE 启动

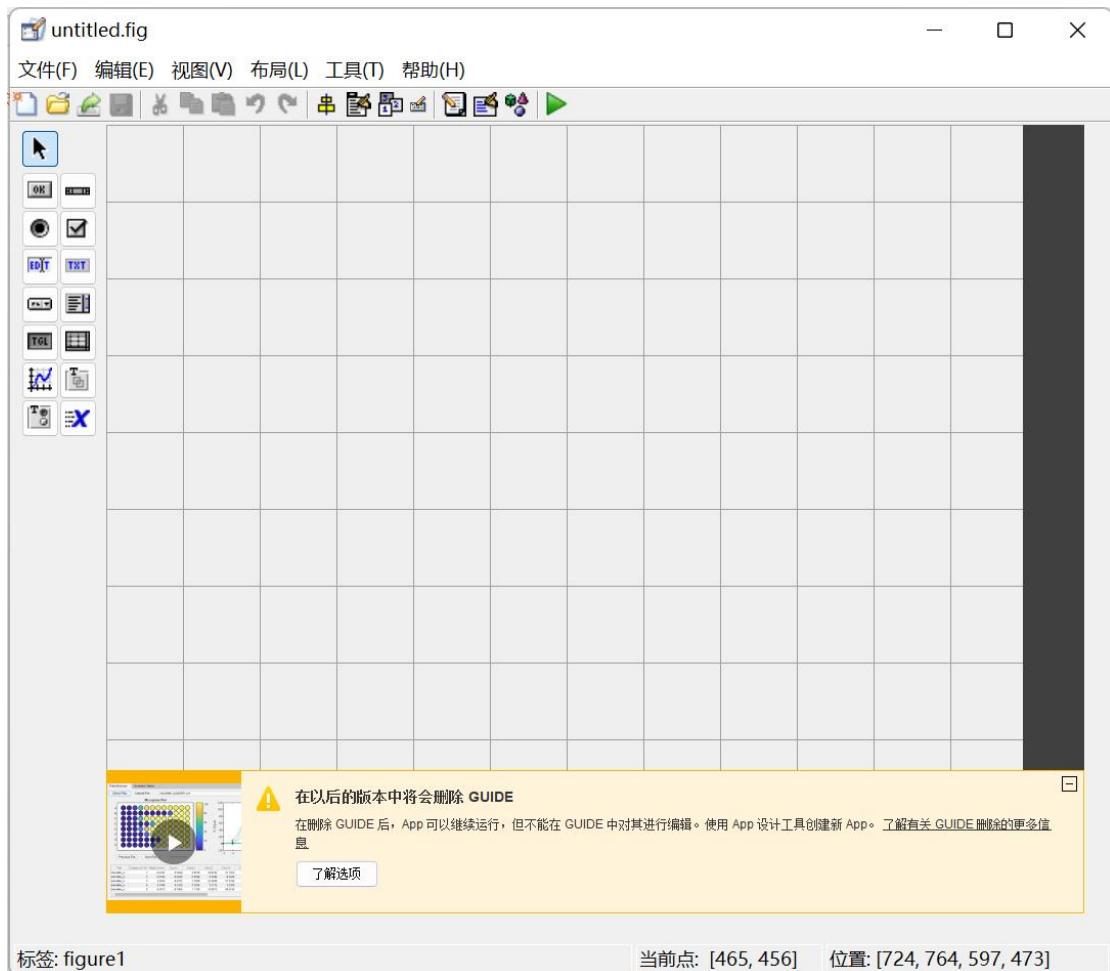
用户可以在MATLAB的命令行窗口输入：guide来启动GUIDE。产生GUIDE的快速启动窗口，如图16.1所示。在图16.1中，创建新的GUI时，有4个模版可以选择：

**Blank GUI (Default):** 系统默认为空白模版。

**GUI with Uicontrols:** 带有uicontrol对象的模版。

**GUI with Axes and Menu:** 带有坐标轴和菜单的模版。

**Modal Question Dialog:** 带有问答式对话框的模版。



>> guide

警告: 以后的版本中将会删除 GUIDE。请改用 APPDESIGNER。

//啊哦 (APP 了)

## GUI 对象的常用属性

利用属性查看器可以查看对象的属性值，也可以对属性值进行修改。用鼠标选中某个对象，然后单击工具栏的 **Property Inspector** 按钮或用鼠标双击该对象，则可以调用该对象的属性查看器。下面介绍一些常用的属性。

### 1. 控件风格和外观

**BackgroundColor:** 设置控件背景颜色，使用[R G B]或颜色定义。

**CData:** 在控件上显示的真彩色图像，使用矩阵表示。

**ForegroundColor:** 用于设置控件上显示的文本的颜色。

**String** 属性：设置控件上的文本，以及列表框和弹出菜单的选项等。

**Visible:** 设置控件是否可见，可以取值为on或off，默认值为on，即控件可见。当Visible属性设置为off时，控件不可见，但是控件仍然存在。

### 2. 控件的常规信息

**Enable** 属性：表示控件的使能状态，取值可以为on、inactive和off，默认值为on。设置为on时，该控件可用。设置为inactive时，控件不可用，但是外表看起来和为on时一样。设置为off时，表示控件不可用。

**Style:** 设置控件对象的类型，可以取值为pushbutton、togglebutton、radiobutton、checkbox、edit、text、slider、frame、listbox、popupmenu，系统默认值为pushbutton。

**Tag:** 该属性为控件名称，利用该名称来命名控件的回调函数的名称。例如，对于pushbutton类型的按钮，GUIDE默认的Tag属性为pushbutton1，则回调函数为pushbutton1\_Callback()。

**TooltipString** 属性：该属性为字符串，用来设置提示信息。当鼠标指针位于此控件上时，显示该提示信息。

**UserData:** 设置用户指定的数据。利用该属性可以实现数据的传递。

**Position:** 该属性设置控件对象的位置和大小，向量为[x y width height]。

**Units:** 设置控件的位置及大小等的单位，可以取值为pixels、normalized、inches、points、centimeters 和 characters。

### 3. 控件的回调函数

**BusyAction:** 处理回调函数的中断。有两种选项：即  
**Cancel:** 取消中断事件，**queue:** 排队（默认设置）。

**ButtonDownFcn**属性：按钮按下时的处理函数。

**CallBack**属性：是连接程序界面整个程序系统的实质性功能的纽带。该属性值应该为一个可以直接求值的字符串，在该对象被选中和改变时，系统将自动地对字符串进行求值。

**CreateFcn:** 在对象产生过程中执行的回调函数。

**DeleteFcn:** 删除对象过程中执行的回调函数。

**Interruptible**属性：指定当前的回调函数在执行时是否允许中断，去执行其他的函数。

---

### 4. 控件当前状态信息

**ListboxTop:** 在列表框中显示的最顶层的字符串的索引。

**Max:** 最大值。

**Min:** 最小值。

**Value:** 控件的当前值。对于单选按钮如果被选中时，该属性的值为**Max**属性值，如果没被选中，则为**Min**属性值。

## GUI 的 M 文件

利用**GUIDE**可以生成M文件框架，通过M文件决定GUI用户操作的响应。通过**GUIDE**可以自动生成M文件，然后在M文件中编写GUI控件的回调函数。

M文件由多个子函数构成，包括主函数、**Opening**函数、**Output**函数和回调函数。其中，主函数不能修改，和文件名相同。

## 数据传递

在GUI的各个控件之间进行数据传递，可以采用以下两种方法：利用结构体**handles**和控件的**UserData**属性。

结构体**handles**保存了GUI的数据。当在运行GUI时，M文件会自动产生结构体**handles**，其中保存了所有GUI对象的数据，可以看作是一个数据的容器。用户可以在**handles**中找到所有控件的信息。**handles**在各个控件的**callback**之间传来传去。

在结构体handles中新建字段a，并保存变量a的值。

```
handles.a=a;  
guidata(hObject, handles);
```

## 简单实例分析

利用GUIDE可以非常方便的建立GUI界面，同时建立了M文件框架。通过在M文件的回调函数中添加程序，可以非常快速的建立GUI界面。下面通过几个例子，介绍如何使用GUIDE提供的控件。

### 简单的按钮

**【例16-2】**简单的按钮例子。打开GUIDE，在GUI布局区放置Push Button按钮。调制控件大小和窗口大小。双击该按钮设置属性值，String设置为：关闭。

```
choice=questdlg('你确定要关闭吗？','关闭对话框  
','Yes','No','No');  
switch choice,  
    case 'Yes'  
        delete(handles.figure1);  
        return;  
    case 'No'  
        return;  
end
```

### 简单的滑动条

**【例16-3】**简单的滑动条例子。打开GUIDE，在GUI布局区放置静态文本控件Static Text和滑动条控件Slider。然后用鼠标双击静态文本控件，设置String属性的值为：“简单滑动条例子！”。设置滑动条的属性，最大值为1.0，最小值为0，最小步长SliderStep为[0.01 0.1]。

```
function slider1_Callback(hObject, eventdata, handles)  
% hObject    handle to slider1 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
  
% Hints: get(hObject,'Value') returns position of slider  
%        get(hObject,'Min') and get(hObject,'Max') to determine range  
val=get(hObject,'value');  
set(handles.text1,'BackgroundColor',[val val val]);
```

## 弹出式菜单

**【例16-4】**弹出式菜单例子程序。打开GUIDE，在GUI布局区放置静态文本控件**Static Text**、可编辑文本**Edit Text**，还有弹出式菜单**Pop-up Menu**。设置静态文本控件的**String**属性为：你选择的颜色。可编辑文本的**String**属性为空。设置弹出式菜单的**String**属性。修改文字的大小为14。用鼠标拖动来修改控件的大小和窗口的大小。

## 列表框

**【例16-5】**列表框例子程序。打开GUIDE，在GUI布局区放置可编辑文本**Edit Text**和列表框**Listbox**。设置静态文本控件的**String**属性为空。设置列表框的**String**属性。修改控件的文字大小为14。用鼠标拖动来修改控件的大小和窗口的大小。

## 加法运算

**【例16-6】**加法运算例子程序。打开GUIDE，在GUI布局区放置2个可编辑文本**Edit Text**，**Tag**属性分别为**edit1**和**edit2**。添加3个静态文本**Static Text**，**Tag**属性分别为**text1**、**text2**和**text3**。然后添加**pushbutton**按钮，**String**属性为“计算”。修改控件的文字大小为14。用鼠标拖动来修改控件和窗口的大小和位置。

```
a=str2num(get(hObject,'String'));
if isempty(a)
    set(hObject,'String','0');
end
guidata(hObject,handles);

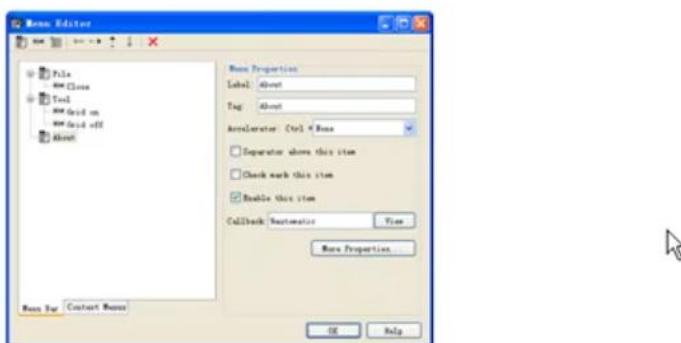
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=get(handles.edit1,'String');
b=get(handles.edit2,'String');
c=str2num(a)+str2num(b);
set(handles.text3,'String',num2str(c));
guidata(hObject,handles);
```

## 坐标轴

**【例16-7】**简单坐标轴例子程序。打开GUIDE，在GUI布局区放置坐标轴Axes，3个可编辑文本Edit Text，3个静态文本Static Text，两个按钮pushbutton。3个静态文本的String属性分别为：频率、初相和幅值。3个可编辑文本的Tag属性分别为edit1、edit2和edit3，分别表示正弦信号的频率、初相和幅值，默认值分别为10、0和1。两个按钮的String属性分别为绘图和清除。修改控件的文字大小为14。用鼠标拖动修改控件和窗口的大小和位置。

## 菜单

**【例16-8】**菜单例子程序。打开GUIDE，在GUI布局区放置坐标轴Axes，单击工具栏的添加菜单按钮，编辑菜单。修改菜单的属性Label和Tag。用鼠标拖动来修改控件和窗口的大小和位置。



## 快捷菜单

**【例16-9】**快捷菜单例子程序。打开GUIDE，在GUI布局区放置坐标轴Axes，单击工具栏的添加菜单按钮，编辑快捷菜单。修改菜单的属性Label和Tag。双击坐标轴，显示坐标轴的属性设置窗口，修改坐标轴的属性UIContextMenu属性为AxesMenu。用鼠标拖动来修改坐标轴和窗口的大小和位置。

## 综合实例分析

利用GUIDE设计GUI界面非常的方便、简单。在设计复杂的GUI界面时，需要很多的控件，例如菜单、按钮、坐标轴等。下面介绍几个综合一些的例子程序。

## 声音的显示和播放

**【例16-10】** 声音显示和播放例子。打开GUIDE，在GUI布局区放置坐标轴Axes，添加弹出式菜单Pop-up Menu，添加2个按钮pushbutton。双击弹出式菜单，修改属性值String，分别为：选择声音、chirp、handel和laughter。修连个按钮的String属性，分别为：播放声音和程序退出，修改Tag属性分别是playbutton和closebutton。用鼠标拖动来修改坐标轴和窗口的大小和位置。

## 语音实时采集和显示

**【例16-11】** 语音实时采集和显示例子。打开GUIDE，在GUI布局区放置2个坐标轴Axes，添加3个按钮pushbutton。修改3个按钮的属性String，分别为：开始、暂停和退出，Tag属性分别为startbutton、stopbutton和exitbutton。修改控件的文字大小为14。用鼠标拖动来修改坐标轴和窗口的大小和位置。

## 本章小结

本章详细的介绍了用于GUI界面开发的GUIDE工具。GUIDE提供了很多的控件，利用这些控件可以非常方便的设计GUI界面。读者对控件的常用属性需要有一个大概的了解，通过属性的设置可以实现复杂的功能。对于常用控件的使用，本章给出了简单的例子程序。最后，通过两个综合一些的例子，讲解如何设计复杂的GUI界面。

# 文件夹管理和文件 I/O 操作

本章将会详细的介绍MATLAB的文件夹管理和常用的文件I/O操作。文件I/O操作是本章的重点，首先介绍了低级和高级文件I/O操作，然后介绍了利用MATLAB读取音频文件、图像文件和视频文件，最后介绍了利用MATLAB读取Excel表格文件。

## 文件夹管理

文件夹管理主要包括：获取当前文件夹、目录的创建和删除等。MATLAB提供了很多文件夹操作函数，可以非常方便的建立和删除文件夹、获取当前文件夹下的文件等。

### 当前文件夹

在MATLAB中，用户在编写脚本M文件或函数M文件时，需要将这些文件放到当前文件夹下，或者放到特定的文件夹中。

在MATLAB的主界面的工具条中，就能显示和设置当前的文件夹。如图17.1所示，显示当前的文件夹是：

C:\Documents and Settings\Administrator\My Documents\MATLAB。这是MATLAB启动后，系统默认的文件夹位置。当前文件夹下没有任何文件。

## 文件夹管理

在MATLAB中，提供了很多文件夹操作的命令，可以在MATLAB的命令行窗口列出当前的文件夹、显示文件和文件夹、以及新建文件夹和删除文件夹等。

表 17.1 常用的文件夹操作命令<sup>①</sup>

| 命令 <sup>②</sup>             | 说明 <sup>③</sup>                |
|-----------------------------|--------------------------------|
| pwd <sup>④</sup>            | 返回当前的文件夹 <sup>⑤</sup>          |
| matlabroot <sup>⑥</sup>     | 返回MATLAB的安装文件夹 <sup>⑦</sup>    |
| dir或ls <sup>⑧</sup>         | 显示当前文件夹中的文件和子文件夹 <sup>⑨</sup>  |
| cd yourdir <sup>⑩</sup>     | 更改文件夹 <sup>⑪</sup>             |
| cd .. <sup>⑫</sup>          | 进入上一层文件夹 <sup>⑬</sup>          |
| what <sup>⑭</sup>           | 显示当前文件夹下的MATLAB文件 <sup>⑮</sup> |
| which filename <sup>⑯</sup> | 返回文件filename的文件夹 <sup>⑰</sup>  |
| mkdir newdir <sup>⑱</sup>   | 创建名字为newdir的文件夹 <sup>⑲</sup>   |
| rmdir newdir <sup>⑳</sup>   | 删除名字为newdir的文件夹 <sup>⑳</sup>   |
| isdir(var) <sup>㉑</sup>     | 判断变量var是否为文件夹 <sup>㉒</sup>     |
| copyfile <sup>㉓</sup>       | 拷贝文件或文件夹 <sup>㉔</sup>          |
| movefile <sup>㉕</sup>       | 移动文件或文件夹 <sup>㉖</sup>          |
| tempdir <sup>㉗</sup>        | 系统的临时存储目录 <sup>㉘</sup>         |
| tempname <sup>㉙</sup>       | 系统的临时文件名 <sup>㉚</sup>          |



```

clear all;
d1=dir
d2=ls
dir *.m
dir *2.m
dir *_1.m

clear all;
if mkdir('temp')
    disp('成功创建文件夹temp');
    cd temp
    dir    I
    cd ..
else
    warning('不能创建文件夹temp');
end                                成功创建文件夹temp
if rmdir('temp')
    disp('成功删除文件夹temp');| . . . I
else
    disp('没有删除文件夹temp');|成功删除文件夹temp
end                                >>

clear all;
if mkdir('temp')
    disp('创建文件夹temp成功');
    if copyfile('chap17_2.m','temp')
        disp('文件拷贝成功');
    else
        warning('拷贝文件失败');
    end
else
    warning('创建文件夹temp失败');| 创建文件夹temp成功
end                                文件拷贝成功
if rmdir('temp')
    disp('删除文件夹temp成功');|删除文件夹temp失败
else
    disp('删除文件夹temp失败');|>>
end

```

## 低级文件 I/O

在MATLAB中，有许多低级文件I/O操作函数。这些函数如表17.2所示。首先利用函数fopen( )打开文件，然后对文件的内容进行读写等操作，最后采用函数fclose( )关闭该文件。

表 17.2 低级文件I/O函数<sup>②</sup>

| 函数 <sup>③</sup>      | 说明 <sup>④</sup>       |
|----------------------|-----------------------|
| fopen <sup>⑤</sup>   | 文件打开 <sup>⑥</sup>     |
| fclose <sup>⑦</sup>  | 文件关闭 <sup>⑧</sup>     |
| fread <sup>⑨</sup>   | 二进制数据的读取 <sup>⑩</sup> |
| fwrite <sup>⑪</sup>  | 二进制数据的写入 <sup>⑫</sup> |
| fget <sup>⑬</sup>    | 逐行读取 <sup>⑭</sup>     |
| fgets <sup>⑮</sup>   | 逐行读取 <sup>⑯</sup>     |
| fscanf <sup>⑰</sup>  | 格式化读取 <sup>⑱</sup>    |
| fprintf <sup>⑲</sup> | 格式化写入 <sup>⑳</sup>    |

## 打开和关闭文件

在MATLAB中，读写文件之前，必须先用函数fopen( )打开或创建文件，并指定对该文件进行的操作方式。

在MATLAB中，文件在进行完读、写等操作后，应及时关闭文件，以免数据丢失。关闭文件采用函数fclose( )。

```
clear all;
(fid,message)=fopen('file1.txt','r');
if fid==-1
    disp(message);
end
fclose(fid);
```

## 读取和写入二进制文件

在MATLAB中，采用函数fopen( )打开文件后，可以采用函数fread( )从文件中读取二进制数据。

在MATLAB中，采用函数fwrite( )向已经打开的文件中写入二进制格式的数据。

```
clear all;
(fid,message)=fopen('chap17_7.txt','w+');
if fid==-1
    disp(message);
else
    fwrite(fid,'Good morning.','uint8');
end
fclose(fid);
(fid,message)=fopen('chap17_7.txt','r');
if fid==-1
    disp(message);
else
    a1=fread(fid);
    a1
    disp(char(a1))
end
fclose(fid);
```

```

clear all;
^
(fid,message)=fopen('chap17_8.txt','w+');
if fid==-1
    disp(message);
else
    magic(5)
    fwrite(fid,magic(5),'int32');
end
fclose(fid);
(fid,message)=fopen('chap17_8.txt','r');
if fid==-1
    disp(message);
else
    a1=fread(fid,[3,3],'int32');
    a1
end
fclose(fid);

```

## 逐行读取文本文件

**MATLAB**中，采用函数`fgetl()`和`fgets()`可以从格式化的文本文件中逐行读取数据，并以字符串的形式保存。这两个函数的功能基本相同，不同之处是函数`fgets()`将新的  
一行字符复制到字符串中，而函数`fgetl()`不是。

```

clear all;
(fid,message)=fopen('chap17_5.m');
if fid==-1
    disp(message);
else
    while 1
        tline=fgetl(fid);
        if ~ischar(tline)
            break;
        end
        disp(tline);
    end
    fclose(fid);
end

```

## 格式化读取和写入文件

在**MATLAB**中，采用函数`fscanf()`格式化读取文本文件，并按指定格式存入到矩阵中。

在**MATLAB**中，采用函数`fprintf()`将数据按指定格式写入到文本文件中。

```

clear all;
a='Good morning, Robinson.'
[fid,message]=fopen('chap17_10.txt','wt')
if fid==-1
    disp(message);
else
    fprintf(fid, '%s', a);
    fclose(fid);
end
[fid,message]=fopen('chap17_10.txt','r');
if fid==-1
    disp(message);
else

    a1=fopen(fid, ' %c', 4)
    frewind(fid);
    a2=fscanf(fid, ' %c', [3, 4])
    frewind(fid)
    a3=fscanf(fid, ' %c')
    frewind(fid)
    a4=fscanf(fid, ' %s', 4)
    frewind(fid)
    a5=fscanf(fid, ' %s')
    fclose(fid);
end

clear all;
t=0:pi/20:2*pi;
x=[t;sin(t)];
delete chap17_11.txt
[fid,message]=fopen('chap17_11.txt','wt');
if fid==-1
    disp(message);
else
    fprintf(fid, '%3.6f %3.6f\n', x);
    fclose(fid);
end
[fid,message]=fopen('chap17_11.txt','r');
if fid==-1

    disp(message);
else
    [A, count]=fscanf(fid, ' %f %f', [2, 21]);
    A=A'
    count
    fclose(fid);
end
close all;
figure;

plot(A(:, 1), A(:, 2));
xlabel('t');
ylabel('sin(t)');

```

## 文件内部的位置控制

在打开文件时，会产生一个文件标识，表示了文件中的当前位置。通过控制文件标识，来进行文件的读取和写入操作。获取文件位置的函数，如表17.4所示。

表 17.4 控制文件位置的函数

| 函数      | 说明          |
|---------|-------------|
| feof    | 测试是否到文件尾    |
| fseek   | 将文件标识移到特定位置 |
| frewind | 将文件标识移到文件头  |
| ftell   | 取得文件标识的位置   |

```
clear all;
a=1:6
[fid,message]=fopen('chap17_12.bin','wt');
if fid==-1
    disp(message);
else
    fwrite(fid,a,'int32');
end
fclose(fid);
[fid,message]=fopen('chap17_12.bin','r');
if fid==-1
    disp(message);
else

    fwrite(fid,a,'int32');
end
fclose(fid);
[fid,message]=fopen('chap17_12.bin','r');
if fid==-1
    disp(message);
else
    while ~feof(fid)
        a1=fread(fid,inf,'int32');
        a1=a1'
    end
    fseek(fid,4,'bof');

    f1=ftell(fid)
    a2=fread(fid,1,'int32')
    fseek(fid,12,'bof');
    f2=ftell(fid)
    a3=fread(fid,1,'int32')
end
fclose(fid); I
```

## 高级文件 I/O

在MATLAB中，使用向导将外部的数据文件导入到MATLAB的工作空间中，然后就可以进行分析和处理了。如果数据文件中，有文本内容，需要进行特别的处理。下面进行详细的介绍。

### 使用向导来导入数据

在MATLAB中，可以使用向导将外部数据文件导入到MATLAB的工作空间中。如果MATLAB的工作空间窗口没有显示，则通过单击MATLAB主界面的Desktop| Workspace显示当前的工作空间窗口。工作空间窗口可以附着在MATLAB的主界面，也可以变为独立的窗口。

利用向导来导入数据文件student.txt。

### 带有分隔符的数据文件

对于带有分隔符的数据文件，可以采用函数`dlmread()`导入到MATLAB的工作空间中，该函数的调用格式为：

`results=dlmread('filename')`: 该函数从数据文件filename导入到MATLAB的工作空间中。

`results=dlmread('filename', 'delimiter')`: 该函数设置分隔符delimiter。

在MATLAB中，可以利用函数`dlmwrite()`将MATLAB工作空间中的矩阵，写入到文本文件中。

如果数据文件的分割符为逗号(,)，也可以采用函数`csvread()`读取该数据文件，利用函数`csvwrite()`向数据文件中写入数据。

```
clear all;
data=csvread('chap17_14_1.txt')
csvwrite('chap17_14_2.dat', data);

//type 命令
```

### 带有文本内容的数据文件

在MATLAB中，采用函数`textread()`读取带有文本的数据文件。该函数的详细调用格式，可以在MATLAB的命令行窗口输入：`help textread`进行查看。

## 利用函数 importdata() 带入数据

在MATLAB中，可以采用函数importdata( )将文件中的数据导入到MATLAB的工作空间。该函数的详细调用格式，可以查询MATLAB的帮助系统。

```
clear all;
s=importdata('NiHao.wav')

s =
    [
        data: [39249x2 double]
        fs: 44100

clear all;
M=importdata('chap17_19.txt',' ',1)
M.data
M.textdata

>> type chap17_19.txt

Day1 Day2 Day3 Day4 Day5
95.01 76.21 61.54 40.57 5.79
23.11 45.65 79.19 93.55 35.29
60.68 1.85 92.18 91.69 81.32
```

## 利用函数 diary() 导出数据

在MATLAB中，可以采用函数diary( )记录用户和MATLAB的交互过程。该函数的调用格式如下：

**diary('filename')**或**diary filename**：保存的文件名为filename。如果没有指定文件名，则文件名默认为diary。

**diary off**：关闭。

**diary on**：打开。

**diary**：在**diary on**和**diary off**之间进行切换。

```
clear all;
diary mydiary.txt;
A=[1, 2, 3; 2, 3, 4]
B(2,1)=100
diary off;
```

## 其他函数

此外，在MATLAB中，还可以采用mat文件来保存数据，采用函数load( )进行数据的导入，利用函数save( )进行数据的导出。

MATLAB还非常方便的导入科学标准格式数据，例如CDF（Common Data Format）、FITS（Flexible Image Transport System）、HDF（Hierarchical Data Format v.4）和H5（Hierarchical Data Format v.5），用户可以查询MATLAB的帮助系统，查询这些数据的导入和导出函数。

## 音频文件

利用MATLAB可以进行语音信号的分析和处理。在MATLAB中，可以读取、播放和写入wav格式的语音和AU格式的语音文件。下面分别进行介绍。常用的音频处理函数，如表17.5所示。

表 17.5 常用的音频处理函数<sup>②</sup>

| 函数 <sup>③</sup>        | 说明 <sup>④</sup>                |
|------------------------|--------------------------------|
| wavread <sup>⑤</sup>   | 读取wav声音文件 <sup>⑥</sup>         |
| auread <sup>⑤</sup>    | 读取AU声音文件 <sup>⑥</sup>          |
| sound <sup>⑤</sup>     | 播放声音 <sup>⑥</sup>              |
| wavplay <sup>⑤</sup>   | 利用Window的播放器来播放声音 <sup>⑥</sup> |
| soundsc <sup>⑤</sup>   | 播放归一化后的声音 <sup>⑥</sup>         |
| wavrecord <sup>⑤</sup> | 录音 <sup>⑥</sup>                |
| wavinfo <sup>⑤</sup>   | 获取wav声音的信息 <sup>⑥</sup>        |
| auinfo <sup>⑤</sup>    | 获取AU声音的信息 <sup>⑥</sup>         |
| wavwrite <sup>⑤</sup>  | 写入wav声音文件 <sup>⑥</sup>         |
| auwrite <sup>⑤</sup>   | 写入AU声音文件 <sup>⑥</sup>          |

### 读取音频文件

在MATLAB中，采用函数wavread( )读取Microsoft的wav格式音频文件到工作空间中。用户可以在MATLAB的命令行窗口输入：help wavread来获取该函数的详细信息。

在MATLAB中，利用函数auread( )将AU格式的语音文件读入到MATLAB的工作空间中。

```
clear all;
[y, fs, nbits, opts]=wavread('DaJiaHao.wav');
size(y)
fs
nbts
opts
t=size(y, 1)/fs
close all;
figure;
subplot(121);
plot(1/fs:1/fs:t, y(:, 1));
xlabel('Time(s)');
```

```

title('左声道');
subplot(122);
plot(1/fs:1/fs:t, y(:, 2));
xlabel('Time(s)');
title('右声道');

clear all;
[y, fs, nbits]=auread('NiHao.au');
size(y)
fs
nbts
t=size(y, 1)/fs
close all;
figure;
subplot(121);
plot(1/fs:1/fs:t, y(:, 1));
xlabel('Time(s)');
title('左声道');
subplot(122);

plot(1/fs:1/fs:t, y(:, 2));
xlabel('Time(s)');
title('右声道');

```

## 播放和录音

在MATLAB中，可以采用3个函数进行语音的播放，分别是  
**函数sound( )、函数wavplay( )和函数soundsc( )。**可以  
采用**函数wavrecord( )**进行录音。下面分别进行介绍。

```

clear all;
[y1, fs1, nbts1]=wavread('DaJiaHao.wav');
sound(y1, fs1, nbts1);
[y2, fs2, nbts2]=auread('NiHao.au');
sound(y2, fs2, nbts2);

clear all;
[y1, fs1]=wavread('DaJiaHao.wav');
wavplay(y1, fs1);
[y2, fs2]=auread('NiHao.au');
wavplay(y2, fs2);
pause(3);
wavplay(y1, fs1, 'async');
wavplay(y2, fs2);    I

clear all;
a=rand(10000, 2);
fs=8000;
sound(a, fs);
pause(3);
soundsc(a, fs);

```

```
clear all;
fs=11025;
y=wavrecord(3*fs, fs, 'int16');
wavplay(y, fs);
//录音
```

## 写入音频文件

在MATLAB中，采用函数**wavinfo( )**来获取**wav**格式的语音文件的信息，采用函数**auinfo( )**来获取**AU**格式的语音文件的信息。

在MATLAB中，利用函数**wavwrite( )**将数据保存到**wav**语音文件中。

在MATLAB中，采用函数**auwrite( )**写入**AU**格式的语音文件

```
clear all;
[m1, d1]=auinfo('NiHao.au')
[m2, d2]=wavinfo('DaJiaHao.wav')

clear all;
[y, fs, nbts]=wavread('DaJiaHao.wav');
sound(y, fs);
y=y.*2;
sound(y, fs);
wavwrite(y, fs, nbts, 'DaJiaHao2.wav');

clear all;
[y, fs, nbts]=wavread('DaJiaHao.wav');
sound(y, fs, nbts);
auwrite(y, fs, nbts, 'linear', 'DaJiaHao.au');
[y1, fs1, nbts1]=auread('DaJiaHao.au');
sound(y1, fs1, nbts1);
```

## 图像文件

MATLAB能够读取、处理和显示多种标准图像文件。

MATLAB的图像处理工具箱在安装目录下的**matlabroot\toolbox\images**文件夹中。在文件夹**matlabroot\toolbox\images\imdemos**中有一些常用的图像文件。在MATLAB中，常用的图像读取、写入和显示函数，如表17.6示。下面进行详细的介绍。

表 17.6 常用的图像处理函数<sup>①</sup>

| 函数 <sup>②</sup>        | 说明 <sup>③</sup>              |
|------------------------|------------------------------|
| imread <sup>④</sup>    | 读取图像文件 <sup>⑤</sup>          |
| imwrite <sup>④</sup>   | 写入图像文件 <sup>⑤</sup>          |
| imfinfo <sup>④</sup>   | 获取图像文件的信息 <sup>⑤</sup>       |
| imshow <sup>④</sup>    | 显示图像 <sup>⑤</sup>            |
| imformats <sup>④</sup> | 获取MATLAB支持的图像格式 <sup>⑤</sup> |
| image <sup>④</sup>     | 显示矩阵数据为图像 <sup>⑤</sup>       |

## 读取和显示图像

在MATLAB中，利用函数imformats( )获取MATLAB支持的图像类型，在命令行窗口输入imformats。MATLAB能够处理常用的16种图像类型。

在MATLAB中，采用函数imread( )读取各种图像文件，利用函数imshow( )来显示各种图象。

```
clear all;
x1=imread('football.jpg');
[x2, map2]=imread('trees.tif');
x3=imread('cell.tif');

clear all;
close all;
[x, map]=imread('trees.tif');
figure;
imshow(x, map);
```

## 获取文件信息

在MATLAB中，利用函数imfinfo( )获取图像文件的详细信息，该函数的调用格式为：info=imfinfo('filename', 'fmt')。其中filename为图像文件名，fmt为图像的类型，该函数的返回值为一个结构体info，包含了该图象的各方面信息，其主要数据包括：文件名（路径）、图像格式、图像格式版本号、文件的修改时间、文件的大小、文件的长度、文件的宽度、每个像素的位数、图象的类型等。

```
clear all;
info=imfinfo('football.jpg', 'jpg')
```

```
info =  
    |     Filename: 'football.jpg'  
    |     FileModDate: '01-三月-2001 09:52:38'  
    |     FileSize: 27130  
    |     Format: 'jpg'  
    |     FormatVersion: ''  
    |         |     Width: 320  
    |         |     Height: 256  
    |         |     BitDepth: 24  
    |         |     ColorType: 'truecolor'  
    |     FormatSignature:  
    |     NumberOfSamples: 3  
    |     CodingMethod: 'Huffman'  
    |     CodingProcess: 'Sequential'  
    |     Comment: {}
```

## 写入图片文件

在MATLAB中，采用函数imwrite( )进行图像的写入，该函数的调用格式为：imwrite(A, 'filename', 'fmt')。其中A为图像数据的矩阵，按照fmt指定的格式将图象写入到文件filename中。

```
clear all;  
close all;  
A=rand(300, 300);  
imshow(A);  
imwrite(A, 'mygray1.jpg', 'jpg');  
  
clear all;  
close all;  
A=imread('football.jpg');  
imwrite(A, 'football.tif', 'tif');  
figure;  
imshow('football.tif');
```

## 通过函数 image 显示图像

在MATLAB中，还可以通过函数image( )将矩阵中的数据显示为图像。该函数的详细调用格式可以通过在MATLAB的命令行窗口输入：help image进行查询。此外，还有函数imagesc( )，其调用格式和函数image( )完全相同，只是对数据进行了归一化。

```

clear all;
close all;
a=magic(10);
b=imread('football.jpg','jpg');
figure;
subplot(121);
image(a);
subplot(122);
image(b);
```

I

## 视频文件

在MATLAB中，视频对象称为**MATLAB movie**。MATLAB可以读入AVI视频文件，并进行视频的播放和保存等。  
MATLAB中，也可以将图像保存为帧，从而创建AVI视频文件。MATLAB中，常用的视频处理函数如表17.7所示。

表 17.7 常用的视频处理函数<sup>1)</sup>

| 函数 <sup>2)</sup>         | 说明 <sup>3)</sup>                      |
|--------------------------|---------------------------------------|
| aviread <sup>4)</sup>    | 读取AVI视频文件 <sup>5)</sup>               |
| aviinfo <sup>4)</sup>    | AVI视频文件的信息 <sup>5)</sup>              |
| movie <sup>4)</sup>      | 播放MATLAB movie <sup>5)</sup>          |
| avifile <sup>4)</sup>    | 创建AVI视频文件 <sup>5)</sup>               |
| getframe <sup>4)</sup>   | 获取视频帧 <sup>5)</sup>                   |
| addframe <sup>4)</sup>   | 添加视频帧 <sup>5)</sup>                   |
| movie2avi <sup>4)</sup>  | 将MATLAB movie转换为AVI视频文件 <sup>5)</sup> |
| im2frame <sup>4)</sup>   | 将图像转换为视频帧 <sup>5)</sup>               |
| frame2im <sup>4)</sup>   | 将视频帧转换为图像 <sup>5)</sup>               |
| mmfileinfo <sup>4)</sup> | 获取多媒体文件的信息 <sup>5)</sup>              |
| close <sup>4)</sup>      | 关闭AVI视频文件 <sup>5)</sup>               |

## 读取视频文件

在MATLAB中，采用函数**aviread( )**读取AVI格式的视频文件，该函数的调用格式为：

**mov=aviread('filename')**。其中**filename**为AVI视频文件的文件名，函数的返回值**mov**为结构体变量，包含两个成员变量**cdata**和**colormap**。如果AVI视频中每一帧都是真彩色图像，则**colormap**为空。

**mov=aviread('filename', index)**：该函数只读取AVI视频中的图像帧，**index**可以为标量，读取某一帧；**index**也可以为矢量，读取某些帧。

在MATLAB中，可以采用函数**movie( )**播放视频文件。

```
clear all;
close all;
mov=aviread('example.avi')
m=aviread('example.avi',30)
figure,
imshow(m.cdata);
```

## 获取文件信息

在MATLAB中，利用函数`aviinfo()`来获取AVI视频文件的信息，该函数的调用格式为：`info=aviinfo('filename')`。其中，`filename`为AVI视频文件的文件名，`info`为结构体，包含AVI文件的信息，包括：文件名、大小、创建的日期、帧数、长和宽等。

```
clear all;
info=aviinfo('example.avi')

info =
| Filename: 'example.avi'
| FileSize: 248320
| FileModDate: '03-七月-2011 16:47:32'
| NumFrames: 41
| FramesPerSecond: 15
| Width: 560
| Height: 420
| ImageType: 'truecolor'
| VideoCompression: 'Indeo5'
| Quality: 0
| NumColormapEntries: 0
```

## 写入视频文件

在MATLAB中，利用函数`movie2avi()`将MATLAB中的视频保存为AVI视频，该函数的调用格式为：  
`movie2avi(mov, 'filename')`。其中`mov`为MATLAB中的视频，保存后的AVI视频文件名为`filename`。

在MATLAB中，利用函数`avifile()`创建视频文件，该函数的调用格式为：`aviobj=avifile('filename')`。该函数创建文件名为`filename`的视频文件`aviobj`。通常通过函数`addframe()`添加帧到AVI视频中，最后通过函数`close()`关闭该视频文件。

```

clear all;
close all;
mov1=aviread('example.avi');
movie2avi(mov1,'myexample.avi');
mov2=aviread('myexample.avi');
h1=figure(1);
movie(h1,mov2);

clear all;
close all;
mov=avifile('example.avi');
t=-pi:pi/20:pi;
figure;
for i=1:length(t)|           I
    plot(t(1:i),sin(t(1:i)),'-r');
    xlabel('t');
    ylabel('sin(t)');
    hold on;
    axis([-pi pi -1 1]);
    F=getframe(gcf);
    mov=addframe(mov,F);
end

mov=close(mov);

clear all;
t=-pi:pi/20:pi;
h1=figure(1);
for i=1:length(t)
    plot(t(1:i),sin(t(1:i)),'-r');
    xlabel('t');
    ylabel('sin(t)');|   I
    hold on;
    axis([-pi pi -1 1]);
    M(i)=getframe;
end
movie2avi(M,'example2.avi');

```

## 表单文件

利用**MATLAB**处理后的数据可以保存到**Excel**表单文件中，也可以将表单文件中的数据导入到**MATLAB**中。在**MATLAB**中，常用的表单文件处理函数。

表 17.8 常用的表单文件处理函数<sup>1)</sup>

| 函数 <sup>2)</sup>       | 说明 <sup>3)</sup>                    |
|------------------------|-------------------------------------|
| xlsread <sup>4)</sup>  | 读取Microsoft的Excel表单文件 <sup>5)</sup> |
| xlsfinfo <sup>4)</sup> | 获取Excel表单文件的信息 <sup>5)</sup>        |
| xlswrite <sup>4)</sup> | 将数据写入Excel表单文件 <sup>5)</sup>        |
| wk1read <sup>4)</sup>  | 读取Lotus 123表单文件 <sup>5)</sup>       |
| wk1finfo <sup>4)</sup> | 获取Lotus 123表单文件信息 <sup>5)</sup>     |
| wk1write <sup>4)</sup> | 将数据写入Lotus 123表单文件 <sup>5)</sup>    |

## 读取表单文件

在MATLAB中，才用函数xlsread( )读取Excel文档中的数据，该函数的调用格式为：

[numeric, txt, raw]=xlsread('filename'): 该函数读取Excel文件filename, 数值内容保存在numeric中, 文本内容保存在txt中, 未处理的单元内容保存在raw中。

[numeric, txt, raw]=xlsread('filename', 'sheet'): 该函数读取文件filename中的表单sheet的内容。

[numeric, txt, raw]=xlsread('filename', 'sheet', 'range') : 该函数设置读取的表单的范围为range。

```
clear all;
data1=xlsread('student.xls', 'sheet2')
[data2, head2, txt2]=xlsread('student.xls', 'sheet2')
A=xlsread('student.xls', 'sheet2', 'A2:A5')
B=xlsread('student.xls', 'sheet2', 'B2:B5')
```

## 获取表单文件信息

在MATLAB中，利用函数xlsinfo( )获取Excel表单文件的详细信息，该函数的调用格式为：[A, descr, format]=xlsinfo('filename')。该函数返回表单文件filename的类型和表的信息。

```
clear all;
[A, descr, format]=xlsinfo('student.xls')

A =
Microsoft Excel Spreadsheet

descr =
'Sheet1'    'Sheet2'    'Sheet3'

format =
xlWorkbookNormal
```

## 写入表单文件

在MATLAB中，利用函数`xlswrite( )`向Excel表单中写入数据，该函数的调用格式为：

`[success, message]=xlswrite('file', array):` 该函数向Excel文件`file`中写入数据`array`。默认位置为第1个表单的A1。如果写入成功返回值`success`为`true`, `message`为空；如果写入出错，`success`返回`false`, `message`为出错信息。

`[success, message]=xlswrite('file', array, 'sheet'):` 该函数向Excel文件的`sheet`表单写入数据。

`[success, message]=xlswrite('file', array, 'sheet', 'range'):` 该函数向Excel文件中`sheet`表单的`range`范围内写入数据。

```
clear all;
data=[2011005, 76, 88; 2011006, 75, 99];
success=xlswrite('student.xls', data, 'sheet2',
```

## Lotus 123 表单文件

在MATLAB中，还可以对Lotus 123表单文件进行处理。利用函数`wk1read( )`读取Lotus 123表单文件，利用函数`wk1info( )`获取表单文件的详细信息，利用函数`wk1write( )`写入表单文件。需要注意的是，在函数名中是数字1，不是英文字符l。对于这些函数的详细使用情况，用户可以通过MATLAB的帮助系统进行查询。

## 本章小结

本章详细的讲解了MATLAB中文件夹管理和文件I/O操作。文件I/O部分介绍了采用向导进行输入的导入、低级文件I/O和高级文件I/O操作。MATLAB中数据的导入和导出非常的灵活，功能非常丰富，需要用户灵活掌握，多加练习。最后，介绍了常用的图像、声音和视频文件的导入和导出。

# MATLAB 编译器

**MATLAB**的功能非常强大，可以进行数学计算、仿真和数据可视化等。有时希望将**MATLAB**编写的程序变成脱离**MATLAB**环境，可以独立运行的可执行软件。**MATLAB**的编译器可以满足用户的这个要求。本章将对**MATLAB**的编译器进行详细的介绍。

## 编译器概述

**MATLAB**编译器将M文件作为输入，产生可以独立运行的应用程序或相应的组件软件。产生的应用程序及组件软件，都是与平台相关的。**MATLAB**编译器可以产生以下几种应用程序或软件组件：

独立运行的应用程序：

**C**或**C++**共享库：在**Windows**系统中为动态链接库**DLL**，这些库也可以在没有**MATLAB**的用户机器上运行。

## 编译器的安装和配置

利用**MATLAB**编译器可以将M文件编译为可执行文件（在**Window**下为.exe文件），或者动态链接库。下面详细的介绍如何利用命令mbuild进行**MATLAB**编译器的安装和配置。

## 查看系统的信息

在**MATLAB**中，采用mbuild命令进行编译器的配置。

mbuild命令的详细使用情况，用户可以在**MATLAB**的命令行窗口输入：`help mbuild`进行查看。通常情况下，mbuild命令中的**setup**选项就可以进行第三方编译器的设置。

```
>> mbuild -setup
Please choose your compiler for building standalone MATLAB applic
Would you like mbuild to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLAB\sys\lcc
[0] None
Compiler: 1
Please verify your choices:
Compiler: Lcc-win32 C 2.4.1
Location: D:\MATLAB\sys\lcc
```

## 对编译器进行配置

MATLAB的编译器支持微软的Microsoft Visual C/C++编译器和Borland C++编译器。用户在MATLAB的命令行窗口执行mbuild -setup命令时，系统会自动列出所有的C/C++编译器。用户可以选择不同的编译器，从而对编译器进行配置。

## 编译器的使用

用户已经正确安装MATLAB Complier，并对编译器进行了配置。本书采用MATLAB自带的Lcc编译器。下面分别介绍采用命令mcc和deploytool工具进行M程序的编译。

### 编译过程

MATLAB编译器采用MATLAB Component Runtime (MCR) 技术，完全支持MATLAB语言。在利用MATLAB编译器将M文件生成可执行文件或链接库的过程是完全自动的。只要源程序是M文件，就可以采用编译命令mcc进行程序的编译。

利用mcc命令可以生成独立的可执行程序，也可以生成动态链接库。关于mcc命令的详细说明，用户可以在MATLAB的命令行窗口输入：help mcc进行查看。

```
>> mcc -m chap18_1.m
>> dir

.
..
chap18_1.exe
chap18_1.m
chap18_1.prj
chap18_1_delay_load.c
```

```
os C:\WINDOWS\system32\cmd.exe
2011-10-27 14:49      2,890 chap18_1_delay_load.c
2011-10-27 14:49      2,871 readme.txt
2011-10-27 14:49      41,954 chap18_1.prj
2011-10-27 14:49      6,340 chap18_1_mcc_component_data.c
2011-10-27 14:49      86,464 chap18_1.exe
          8 个文件   517,668 字节
          2 个目录  4,532,289,536 可用字节
>> !chap18_1.exe
A =
8     1     6
3     5     7
4     9     2
B =
0.1422   -0.1444    0.0639
-0.0611    0.0222    0.1056
-0.0194    0.1889   -0.1028
0.1472   -0.1444    0.0639
-0.0611    0.0222    0.1056
-0.0194    0.1889   -0.1028
```

```

function [m, n, p]=myfun181(A)
% 计算矩阵元素中大于0、小于0和等于0的个数
m=sum(sum(A>0));
n=sum(sum(A<0));
p=sum(sum(A==0)); //妙！

>> mcc -m chap18_2.m myfun181.m
>> dir

.
..
chap18_2.exe          chap18_2_main.c
chap18_2.m             chap18_2_mcc_component_data.c
chap18_2.prj            mccExcludedFiles.log
                         myfun181.m
                         readme.txt
chap18_2_delay_load.c

>> !chap18_2.exe

A =
-2    -3     4
 0    -1     2
 5     8     0

m =
        I
 4

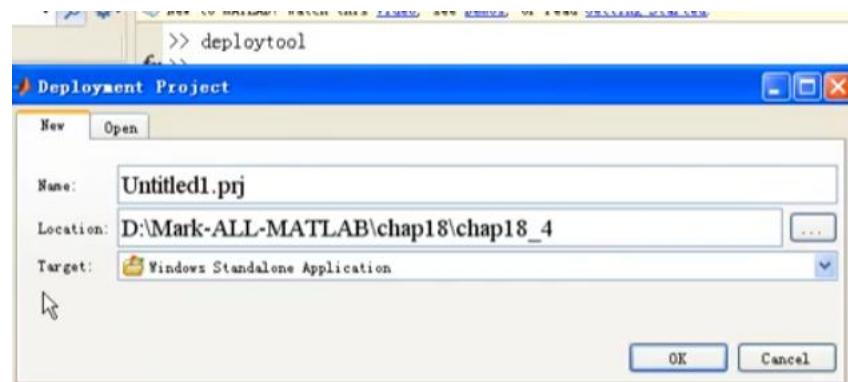
n =
 3

p =

```

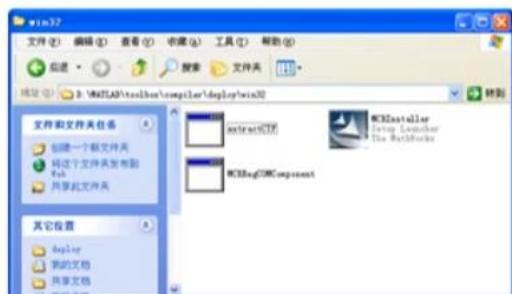
## deploytool 工具

在MATLAB中，提供了图形化的编译工具**deploytool**，在命令窗口输入：**help deploytool**，可以查看**deploytool**的功能。在MATLAB的命令行窗口输入：**deploytool**后将启动该工具，如图18.6所示，可以在M文件所在的文件夹内新建工程文件，默认为：**Untitled1.prj**，也可以打开现有的工程文件。命令行工具**mcc**在运行时，默认产生和M文件同名的工程文件，后缀为**.prj**，可以用**deploytool**直接打开。



## MCR 的安装

为了能够运行MATLAB编译产生的可执行文件，用户的计算机上面必须安装MCR。MCR默认存放在MATLAB的根目录下的\toolbox\compiler\deploy\win32文件夹中，如图18.13所示。



## 本章小结

本章对MATLAB的编译器的安装、配置进行了讲解，并详细的介绍了如何将M文件编译成脱离MATLAB环境的可执行文件，包括命令mcc和工具deploytool。最后介绍了MCR的安装。

# 信号处理工具箱

数字信号处理是一门非常重要的基础学科。在MATLAB中，信号处理工具箱非常的重要。本章首先介绍了常用的离散信号和信号发生器，接着介绍了如何改变信号的采样率。然后介绍了利用MATLAB进行数字滤波，包括FIR滤波器和IIR滤波器等。滤波器是用来消除或减弱干扰噪声，是信号处理中一个非常重要的课题。最后介绍了功率谱估计，包括非参数模型法、参数模型法和子空间法进行功率谱估计。

## 数字信号处理

首先简单介绍了MATLAB的信号处理工具箱，然后介绍了常用的离散序列，包括单位抽样序列、单位阶跃序列、正弦序列、实指数序列、复指数序列、随机序列和周期序列。最后介绍了信号处理工具箱中的信号发生器，能够产生常用的矩形波和三角波等。

## 信号处理工具箱

MATLAB的信号处理工具箱提供了大量的函数，包含各种信号处理算法，包括波形产生、数字滤波器、模拟滤波器、线性系统变换、参数建模、线性预测、统计信号处理、倒谱分析和图形用户接口等。

## 离散信号

实际的信号往往是连续的，经过采样和量化以后变为离散的序列。利用MATLAB可以对离散的序列进行分析和处理。可以采用向量来表示一个有限长度的离散序列。

对于数字信号处理工具箱中的大多数函数，都需要一个时间向量。例如，采样率为1000Hz，时间为一秒的时间向量为：

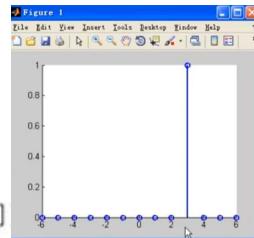
**t=(0:0.001:1)。**

```
clear all;
close all;
t=(0:0.001:1)';
y=sin(2*pi*50*t)+2*sin(2*pi*120*t);
randn('state',0);
yn=y+0.5*randn(size(t));
figure;
subplot(121);
plot(t(1:100), y(1:100));
subplot(122);
plot(t(1:100), yn(1:100));
set(gcf,'position',[200, 200, 800, 300]);
```

```

clear all;
close all;
n=-6:6;
y=[zeros(1, 9), 1, zeros(1, 3)];
figure;
stem(n, y, 'LineWidth', 2);
box off;
set(gcf, 'position', [200, 200, 400, 300])

```

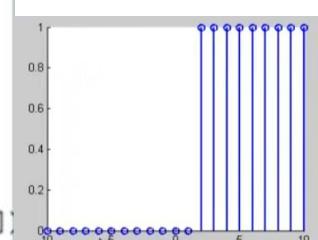


//单位抽样序列

```

clear all;
close all;
n=-10:10;
yn=[zeros(1, 12), ones(1, 9)];
figure;
stem(n, yn, 'LineWidth', 2);
box off;
set(gcf, 'position', [200, 200, 400, 300])

```

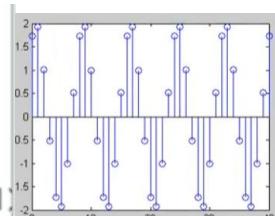


//单位阶跃序列

```

clear all;
close all;
n=0:40;
x=2*sin(pi*n/4+pi/3);
figure;
stem(n, x);
set(gcf, 'position', [200, 200, 400, 300])

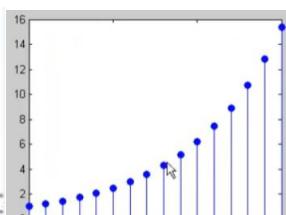
```



```

clear all;
close all;
n=0:15;
x=(6/5).^(1/5);
figure;
stem(n, x, 'filled');
set(gcf, 'position', [200, 200, 400, 300])

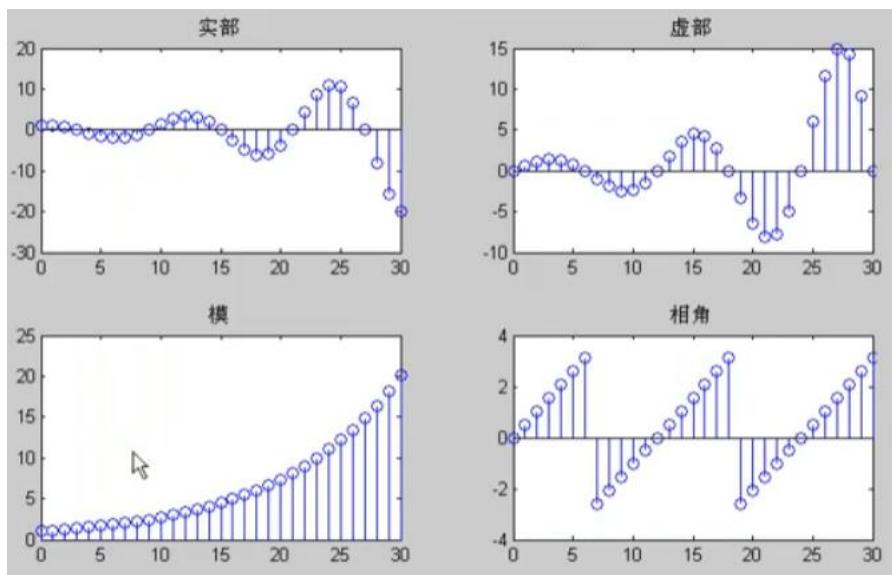
```



```

clear all;
close all;
n=0:30;
x=exp(0.1+i*pi/6).^n;
xreal=real(x);
ximag=imag(x);
xabs=abs(x);
xangle=angle(x);
figure;
subplot(221); stem(n, xreal);
title(' 实部');
subplot(222); stem(n, ximag);
title(' 虚部');
subplot(223); stem(n, xabs);
title(' 模');
subplot(224); stem(n, xangle);
title(' 相角');

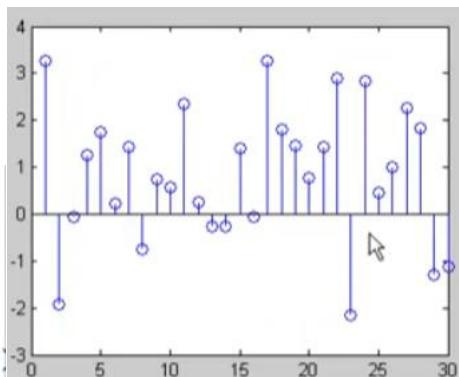
```



```

clear all;
close all;
n=1:30;
x=1+sqrt(2)*randn(1,30);
figure;
stem(n,x);
set(gcf,'position',[200,200,400,300]);

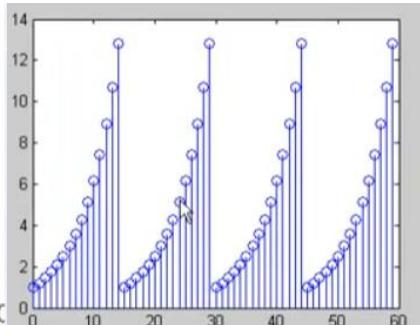
```



```

clear all;
close all;
n1=0:14;
x=(6/5).^n1;
N=15;|
k=4;
n2=0:(k*N-1);
y=x(mod(n2,N)+1);
figure;
stem(n2,y);
set(gcf,'position',[200,200,400,300]);

```



## 波形发生器

在MATLAB的数字信号处理工具箱中，有多个函数用来产生各种常用的信号波形，例如方波、三角波、脉冲信号等。这些函数如表19.1所示。

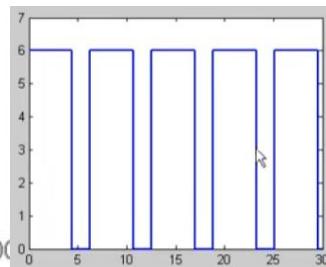
表 19.1 波形发生器函数<sup>④</sup>

| 函数 <sup>④</sup>        | 说明 <sup>④</sup>            | 函数 <sup>④</sup>       | 说明 <sup>④</sup>        |
|------------------------|----------------------------|-----------------------|------------------------|
| chirp <sup>④</sup>     | 扫频信号 <sup>④</sup>          | sawtooth <sup>④</sup> | 锯齿波或三角波信号 <sup>④</sup> |
| diric <sup>④</sup>     | Dirichlet函数波形 <sup>④</sup> | sinc <sup>④</sup>     | sinc函数波形 <sup>④</sup>  |
| gausplus <sup>④</sup>  | 高斯调制正弦波形 <sup>④</sup>      | square <sup>④</sup>   | 方波信号 <sup>④</sup>      |
| gmonopuls <sup>④</sup> | 高斯单脉冲 <sup>④</sup>         | tripuls <sup>④</sup>  | 三角脉冲信号 <sup>④</sup>    |
| pulstran <sup>④</sup>  | 脉冲序列 <sup>④</sup>          | vco <sup>④</sup>      | 压控振荡函数波形 <sup>④</sup>  |
| rectpuls <sup>④</sup>  | 矩形脉冲信号 <sup>④</sup>        |                       |                        |

```

clear all;
close all;
t=0:0.01:30;
x=(square(t, 70)+1)*3;
figure;
plot(t, x, 'LineWidth', 2);
axis([0, 30, 0, 7]);
set(gca, 'xlim', [0 30]);
set(gcf, 'position', [200, 200, 400, 300]);

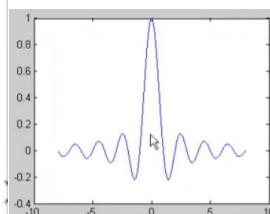
```



```

clear all;
close all;
t=-8:0.01:8;
y=sinc(t);
figure;
plot(t, y); I
set(gcf, 'position', [200, 200, 400, 300]);

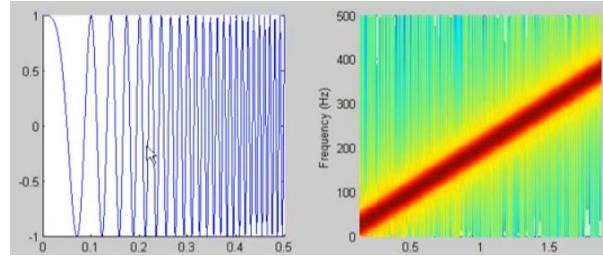
```



```

close all;
fs=1000;
t=0:1/fs:2;
f0=0;
t1=2; I
f1=400;
y=chirp(t, f0, t1, f1);
figure;
subplot(121);
plot(t, y);
axis([0, 0.5, -1, 1]);
subplot(122);

```



```

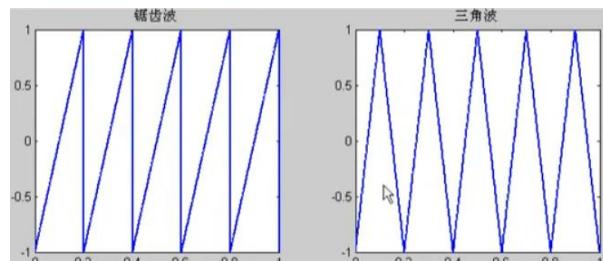
spectrogram(y, 256, 250, 256, fs, 'yaxis'
set(gcf, 'position', [200, 200, 800, 300]);

```

```

close all;
fs=1000;
t=0:1/fs:1;
f=5; I
x=sawtooth(2*pi*f*t);
y=sawtooth(2*pi*f*t, 0.5);
figure;
subplot(121);
plot(t, x, 'LineWidth', 2);
title('锯齿波');
subplot(122);
plot(t, y, 'LineWidth', 2);
title('三角波');

```



```

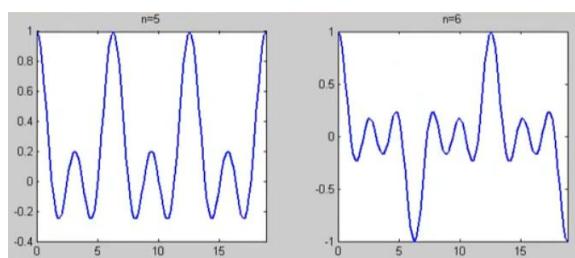
set(gcf, 'position', [200, 200, 800, 300]);

```

```

clear all;
close all;
x=0:pi/20:6*pi;
y1=diric(x, 5);
y2=diric(x, 6);
figure;
subplot(121);
plot(x, y1, 'LineWidth', 2);
title('n=5');
set(gca, 'xlim', [0, 6*pi]);
subplot(122);

```



```

plot(x, y2, 'LineWidth', 2);
title('n=6');
set(gca, 'xlim', [0, 6*pi]);
set(gcf, 'position', [200, 200, 800, 300]);

clear all;
close all;
t=-4:0.01:4;
y1=tripuls(t, 2);
y2=tripuls(t, 2, 0.8);
figure;
subplot(121); I
plot(t, y1, 'LineWidth', 2);
subplot(122);
plot(t, y2, 'LineWidth', 2);
set(gcf, 'position', [200, 200, 800, 300]);
```

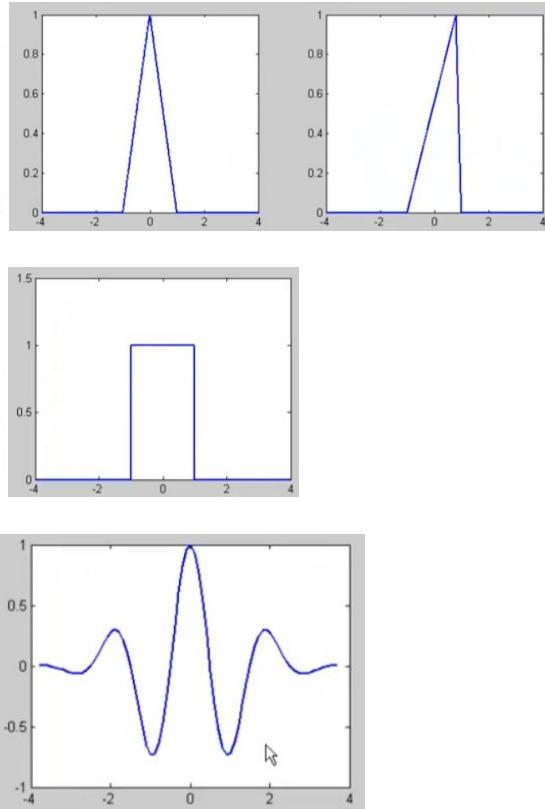
```

clear all;
close all;
t=-4:0.01:4;
y=rectpuls(t, 2);
figure;
plot(t, y, 'LineWidth', 2);
set(gca, 'ylim', [0 1.5]);
set(gcf, 'position', [200, 200, 400, 300]);
```

```

clear all;
close all;
fc=50e3;
bw=0.6;
tc=gauspuls('cutoff', fc, bw, [], -40);
t=-tc:1e-6:tc;
y=gauspuls(t, fc, bw);
figure;
plot(t, y, 'LineWidth', 2);
set(gcf, 'position', [200, 200, 400, 300]);
```



## 改变信号的采样率

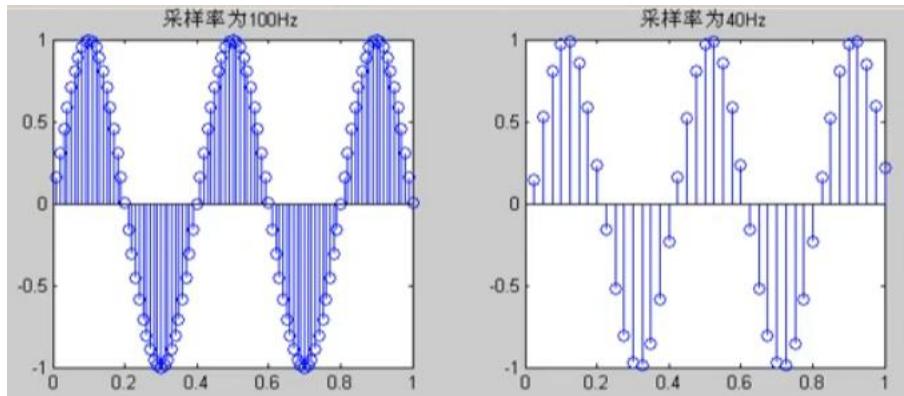
在将信号传送到计算机进行处理时，已经变为数字信号，具有固定的采样率。有时需要改变信号的采样率，提高信号的采样率或降低信号的采样率。在MATLAB的信号处理工具箱提供很多函数来改变信号的采样率。下面进行介绍。

### 改变信号的采样率

在MATLAB中，采用函数`resample()`按有理分数来改变信号的采样率，该函数的常用调用格式为：

`y=resample(x, p, q)`: 该函数对信号x进行重新采样，采样率为原采样率的 $p/q$ 倍，其中p和q必须为正整数。如果x为矩阵，则对x的每一列进行重采样。

`[y, b]=resample(x, p, q)`: 该函数返回在重采样时的滤波器的系数向量b。



## 降低信号的采样率

在MATLAB中，可以采用函数`downsample( )`和函数`decimate( )`来降低信号的采样率。函数`downsample( )`将采样率按整数倍下降，该函数的调用格式为：

`y=downsample(x, n)`: 该函数将采样率降为原来的 $1/n$ ，即每n个采样点中只取其中的第一个采样点。如果x为矩阵，则按矩阵的每一列进行。

`y=downsample(x, n, phase)`: 该函数中参数`phase`为相位，必须为整数，介于0到n-1之间。默认为0。

```
clear all;
close all;
x1=1:10
y1=downsample(x1, 3) I
y2=downsample(x1, 3, 2)
x2=[1:3;4:6;7:9]
y3=downsample(x2, 2)
```

在MATLAB中，函数`decimate( )`通过低通滤波来降低信号的采样率，该函数的调用格式为：

`y=decimate(x, r)`: 该函数降低信号的采样率，输出信号y的长度是信号x的 $1/r$ 。默认的滤波器为8阶的Chebyshev IIR低通滤波器

`y=decimate(x, r, n)`: 该函数中采用n阶Chebyshev IIR滤波器。

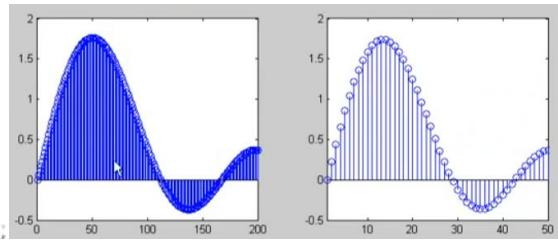
`y=decimate(x, r, 'fir')`: 该函数采用30阶FIR滤波器。

`y=decimate(x, r, n, 'fir')`: 该函数采用n阶FIR滤波器。

```

clear all;
close all;
fs=1e4;
t=0:1/fs:1;
x=sin(2*pi*30*t)+sin(2*pi*60*t);
y=decimate(x, 4);
figure;
subplot(121);
stem(x(1:200));
subplot(122);
stem(y(1:50));
set(gca, 'xlim', [1, 50]);
set(gcf, 'position', [200, 200, 800, 300]);

```



## 提高信号的采样率

在MATLAB中，采用函数`interp()`和函数`upsample()`来提高信号的采样率。函数`upsample()`将信号的采样率提高整数倍，该函数的调用格式为：

`y=upsample(x, n)`: 该函数通过插入n-1个0，将信号的采样率提高n倍。输出信号y的长度为信号x的n倍。如果x为矩阵，则对矩阵的每一列来提高采样率。

`y=upsample(x, n, phase)`: 该函数中参数`phase`为相位，介于0到n-1之间。

```

clear all;
close all;
x1=1:4;
y1=upsample(x1, 3);
y2=upsample(x1, 3, 2)

```

在MATLAB中，还可以通过函数`interp()`来提高信号的采样率。该函数采用低通滤波器来提高信号的采样率，其调用格式为：

`y=interp(x, r)`: 该函数将信号的采样率提高r倍。

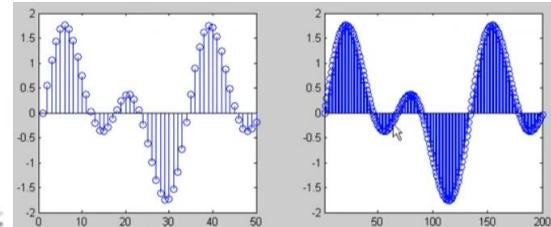
`y=interp(x, r, l, alpha)`: 该函数设置滤波器的长度为l，默认为4，截止频率为alpha，alpha默认为0.5。

`[y, b]=interp(x, r, l, alpha)`: 该函数的返回值b为滤波器的系数。

```

clear all;
close all;
fs=1e3;
t=0:1/fs:1;
x=sin(2*pi*30*t)+sin(2*pi*60*t);
y=interp(x, 4);
figure;
subplot(121);
stem(x(1:50));
subplot(122);
stem(y(1:200));
set(gca, 'xlim', [1, 200]);
set(gcf, 'position', [200, 200, 800, 300]);

```



## FIR 数字滤波器

根据信号的不同特性消除或减弱干扰噪声，提取有用信号的过程称为滤波，而实现滤波功能的系统称为滤波器。经典滤波器从功能上可以分为低通、高通、带通和带阻滤波器，每一种又有模拟滤波器和数字滤波器。对于数字滤波器又分为FIR数字滤波器和IIR数字滤波器。下面介绍FIR数字滤波器。

### FIR 滤波器介绍

FIR滤波器有如下的优点：FIR滤波器总是稳定的；FIR滤波器能够用硬件实现；FIR滤波器具有线性相位。FIR滤波器的缺点是设计相同性能指标的数字滤波器，FIR滤波器需要比IIR滤波器高的多的阶数。FIR滤波器的系统函数只是 $z^{-1}$ 的多项式，IIR滤波器的系统函数采用的是有理分式。FIR滤波器可以采用窗函数法进行设计。

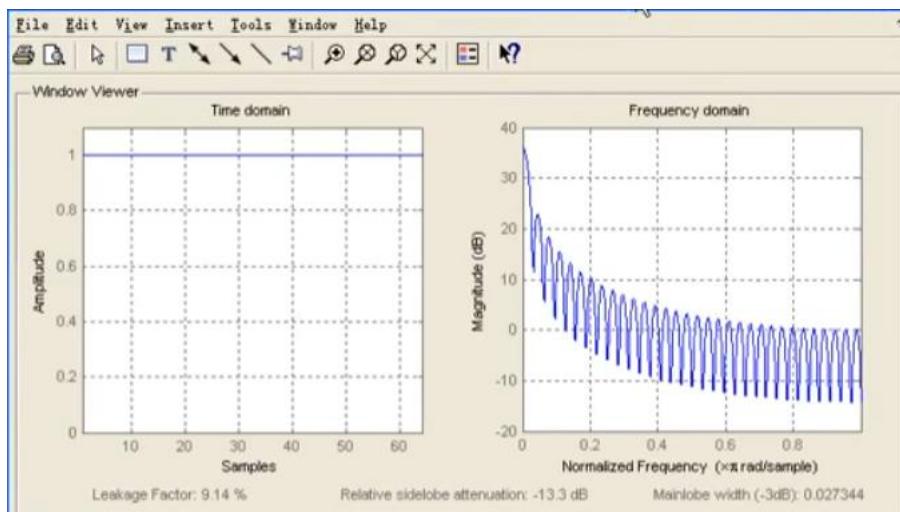
### 常用的窗函数

在MATLAB中，可以采用函数wvtool( )来查看窗函数的时域和频域波形。该函数的调用格式为：

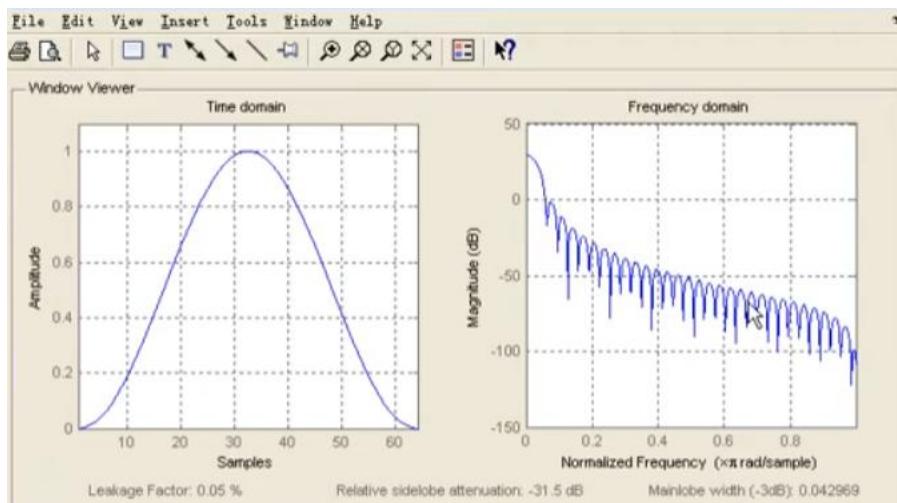
wvtool(winname(n))，其中参数winname为窗函数的名字，n为数据点数。

1. 矩形窗 (Rectangular window)
2. 汉宁窗 (Hanning window)
3. 海明窗 (Hamming window)
4. 布莱克曼窗 (Blackman window)

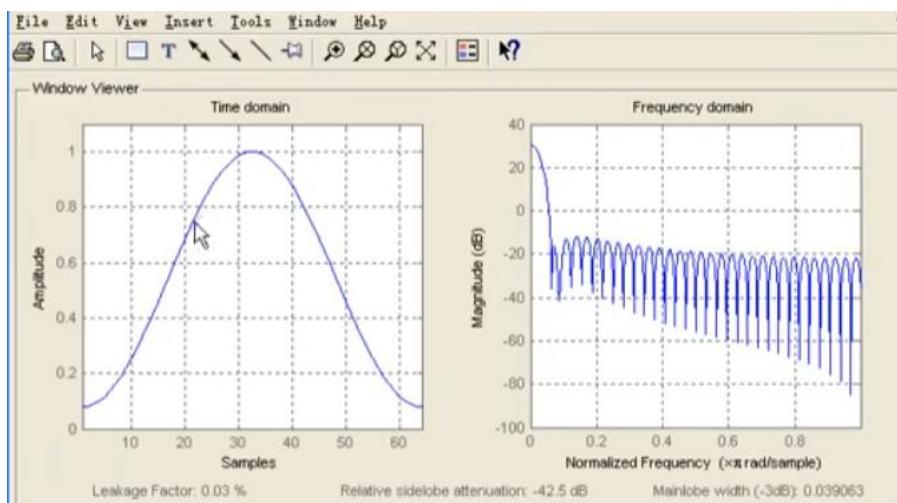
```
>> wvtool(rectwin(64))|
```



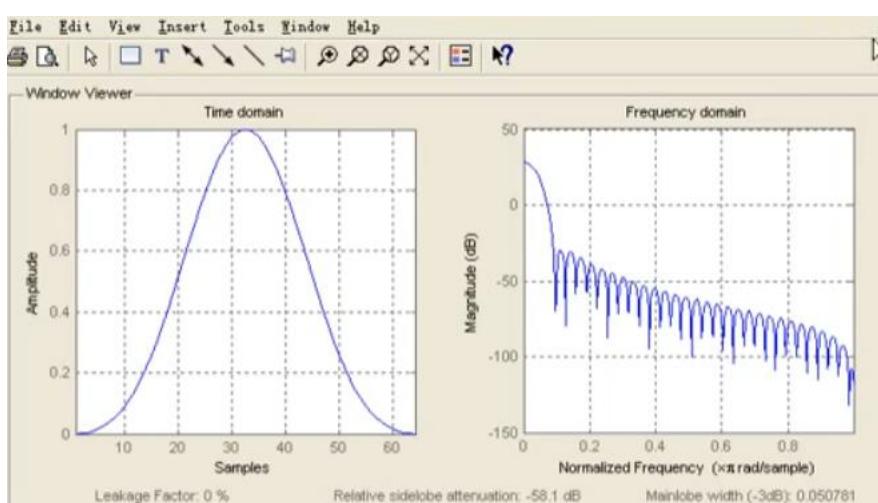
```
>> wvtool(hann(64))
```



```
>> wvtool(hamming(64))
```



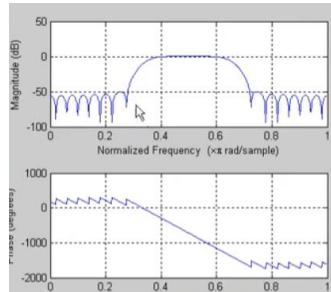
```
>> wvtool(blackman(64))
```



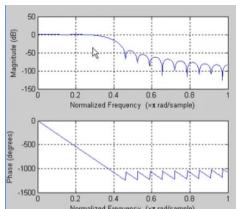
## 基于窗函数的 FIR 滤波器

在MATLAB的信号处理工具箱中，函数fir1( )采用窗函数法设计FIR数字滤波器，能够设计低通、高通、带通、带阻滤波器。该算法的基本思想是，假定 $w(n)$ 为指定的窗函数，理想滤波器的单位冲激响应为 $h(n)$ ，则得到滤波器的系数 $b(n)=w(n)h(n)$ ，其中 $1 \leq n \leq N$ 。

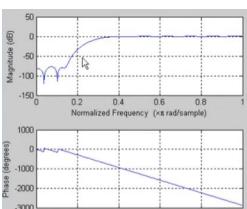
```
clear all;
close all;
n=48;
Wn=[0.35 0.65];
b=fir1(n, Wn);
freqz(b, 1, 512);
set(gcf, 'position', [200, 200, 450, 400]);
```



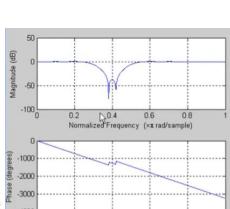
```
clear all;
close all;
n=30;
Wn=0.35;
b=fir1(n, Wn, 'low', hann(n+1));
freqz(b, 1, 512);
set(gcf, 'position', [200, 200, 450, 400]);
```



```
clear all;
close all;
n=36;
Wn=0.3;
b=fir1(n, Wn, 'high', blackman(n+1));
freqz(b, 1, 512);
set(gcf, 'position', [200, 200, 450, 400]);
```



```
clear all;
close all;
n=40;
Wn=[0.3 0.5];
b=fir1(n, Wn, 'stop', hann(n+1));
freqz(b, 1, 512);
set(gcf, 'position', [200, 200, 450, 400]);
```



## 基于频率采样的 FIR 滤波器

在MATLAB中，采用函数fir2( )设计基于频率采样的FIR滤波器。该函数的常用调用格式为：

**b=fir2(n, f, m):** 该函数产生采用频率采样得到的滤波器系数。其中参数n为滤波器的阶数，f为归一化的频率向量，介于[0, 1]，其中1对应的是奈奎斯特频率。m为每个频率点对应的幅度向量，和参数f具有相同的长度。

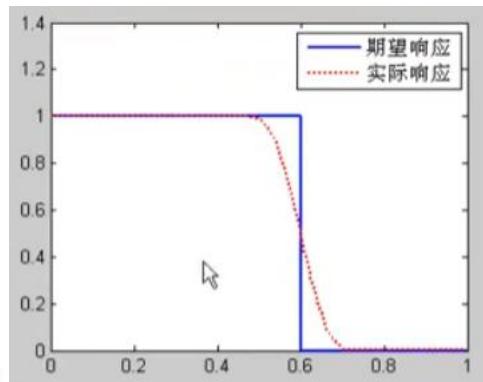
**b=fir2(n, f, m, window):** 该函数设置窗函数，长度为n+1，，默认为汉明窗。

**b=fir2(n, f, m, npt, window):** 该函数中参数npt为频率响应的内插点数。

```

clear all;
close all;
n=30;
f=[0 0.6 0.6 1];
m=[1 1 0 0];
b=fir2(n, f, m);
[h, w]=freqz(b, 1, 128);
figure;
plot(f, m, 'LineWidth', 2);
hold on;
plot(w/pi, abs(h), 'r:', 'LineWidth', 2);
legend('期望响应', '实际响应');
set(gcf, 'position', [200, 200, 400, 300]);

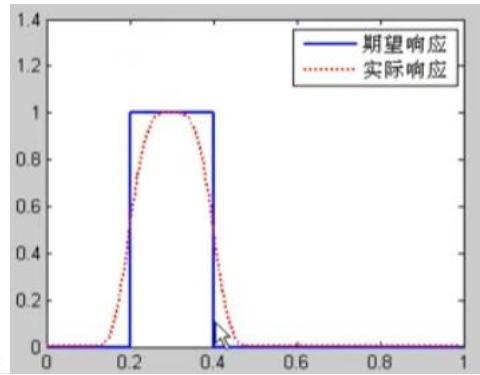
```



```

clear all;
close all;
n=50;
f=[0 0.2 0.2 0.4 0.4 1];
m=[0 0 1 1 0 0];
b=fir2(n, f, m);
[h, w]=freqz(b, 1, 128);
figure;
plot(f, m, 'LineWidth', 2);
hold on;
plot(w/pi, abs(h), 'r:', 'LineWidth', 2);
legend('期望响应', '实际响应');
set(gcf, 'position', [200, 200, 400, 300]);

```



## FIR 滤波器实例分析

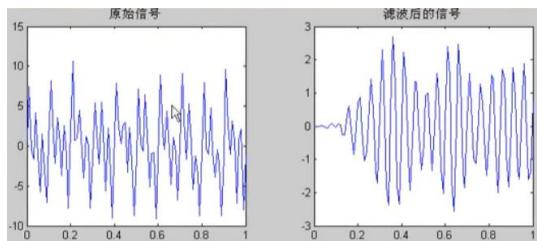
首先建立含有3种频率和白噪声的信号。然后，利用窗函数法设计FIR带通滤波器，并对数字信号进行带通滤波。

```

close all;
fs=100;
t=0:1/fs:1;
x=3*sin(2*pi*10*t)+2*sin(2*pi*20*t)+5*s
n=4;
Wn=[15 25]*2/fs;
b=fir1(n, Wn);
y=filter(b, 1, x);
figure;
subplot(121);
plot(t, x);
title('原始信号');
subplot(122);
plot(t, y);
title('滤波后的信号');
set(gcf, 'position', [200, 200, 800, 300]);

```

//没什么奇怪的，数学算的!!!



## IIR 数字滤波器

IIR滤波器，即无限长冲激响应（infinite impulse response, IIR）滤波器，它的单位冲激响应是无限长的。

设计IIR滤波器，就是寻找滤波器的各个系数 $a_k$ 和 $b_k$ 。

下面对IIR滤波器进行详细的介绍。

## IIR 数字滤波器介绍

IIR数字滤波器的差分方程为：

$$y[n] = -\sum_{k=1}^N a_k y[n-k] + \sum_{k=0}^M b_k x[n-k]$$

IIR数字滤波器的系统函数为：

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

一般应满足  $M \leq N$ ，这类系统称为N阶系统，当  $M > N$  时，可看成一个N阶IIR子系统与一个( $M-N$ )阶的FIR子系统的级联。以下讨论都假定  $M \leq N$ 。设计IIR滤波器，就是获取滤波器的各个系数  $a_k$  和  $b_k$ 。

## IIR 数字滤波器

在MATLAB中，采用函数butter( )设计巴特沃兹滤波器。

函数cheby1( )设计切比雪夫I型滤波器。

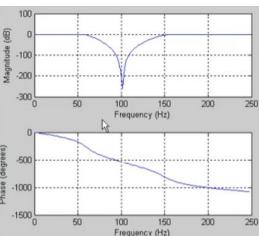
函数cheby2( )设计切比雪夫II型滤波器。

函数ellip( )设计椭圆数字滤波器。

函数yulewalk( )设计递归型的IIR数字滤波器。

```
clear all;
close all;
fs=500;
n=10;
Wn=[50 80]*2/fs;
[b, a]=butter(n, Wn);
freqz(b, a, 256, fs);
set(gcf, 'position', [200, 200, 450, 400]);

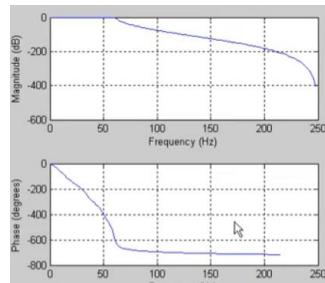
clear all;
close all;
fs=500;
n=6;
Wn=[60 150]*2/fs;
[b, a]=butter(n, Wn, 'stop');
freqz(b, a, 128, fs);
set(gcf, 'position', [200, 200, 450, 400]);
```



```

clear all;
close all;
fs=500;
n=8;
R=2;
Wp=60*pi/2/fs;
[b, a]=cheby1(n, R, Wp, 'low');
freqz(b, a, 128, fs);
set(gcf, 'position', [200, 200, 450, 400]);

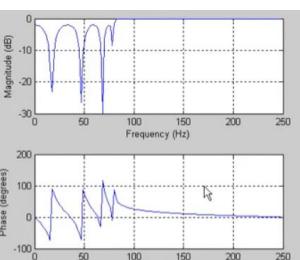
```



```

clear all;
close all;
fs=500;
n=8;
R=2;
Wp=80*pi/2/fs;
[b, a]=cheby2(n, R, Wp, 'high');
freqz(b, a, 128, fs);
set(gcf, 'position', [200, 200, 450, 400]);

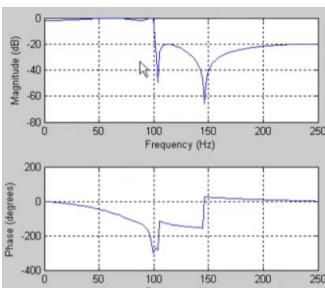
```



```

clear all;
close all;
fs=500;
n=8;
Rp=2;
Rs=20;
Wp=100*pi/2/fs;
[b, a]=ellip(n, Rp, Rs, Wp, 'low');
freqz(b, a, 128, fs);
set(gcf, 'position', [200, 200, 450, 400]);

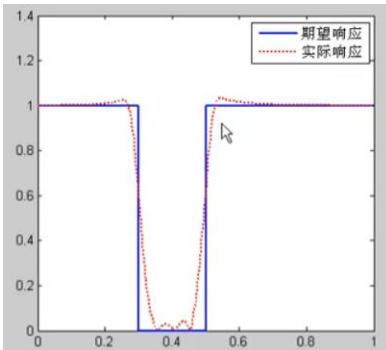
```



```

%visual arr.
close all;
fs=500;
n=14;
f=[0 150 150 250 250 500]/fs;
m=[1 1 0 0 1 1];
[b, a]=fulewalk(n, f, m);
[h, w]=freqz(b, a, 256);
figure;
plot(f, m, 'LineWidth', 2);
hold on;
plot(w/pi, abs(h), 'r:', 'LineWidth', 2);
legend('期望响应', '实际响应');
set(gcf, 'position', [200, 200, 450, 400]);

```



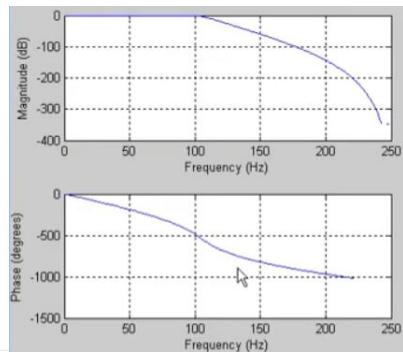
## IIR 数字滤波器阶数的选择

在MATLAB中，可以通过函数来确定IIR滤波器的最小的阶数。对于巴特沃兹滤波器，阶数的选择函数为**buttord()**。对于Chebyshev I型滤波器，阶数的选择函数为**cheb1ord()**。对于Chebyshev II型滤波器，阶数的选择函数为**cheby2ord()**。对于椭圆滤波器，阶数的选择函数为**ellipord()**。

```

clear all;
close all;
fs=500;
Wp=100*pi/2/fs;
Ws=150*pi/2/fs;
Rp=2;
Rs=60;
[n, Wn]=buttord(Wp, Ws, Rp, Rs)
[b, a]=butter(n, Wn);
figure;
freqz(b, a, 256, fs);
set(gcf, 'position', [200, 200, 450, 400]);

```



## IIR 滤波器实例分析

首先建立含有3种频率（10Hz、20Hz和30Hz）正弦信号。

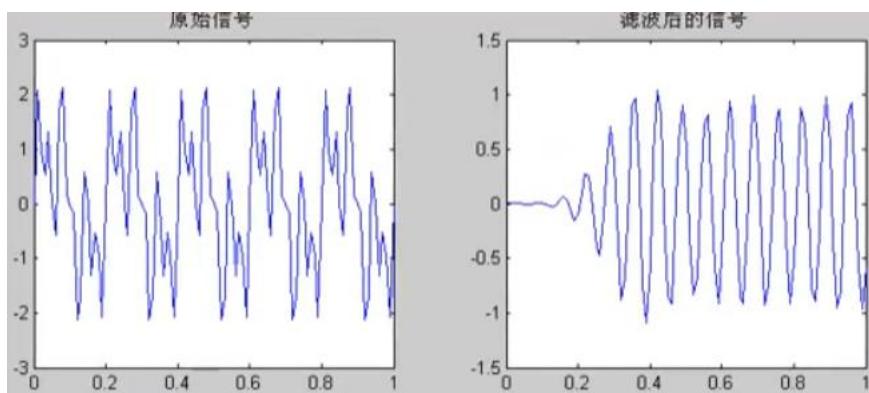
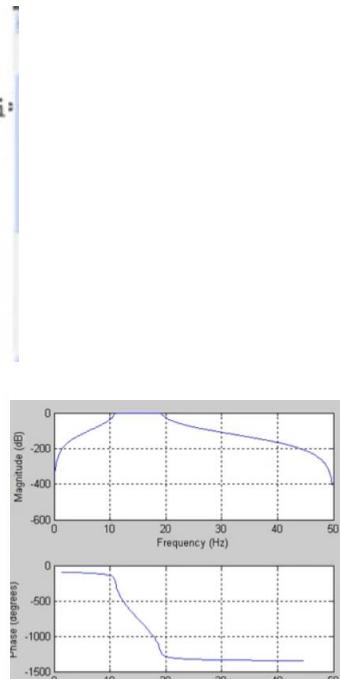
然后设计IIR带通数字滤波器，并对数字信号进行带通滤波。

```

close all;
fs=100;
t=0:1/fs:1;
x=sin(2*pi*5*t)+sin(2*pi*15*t)+sin(2*pi*25*t);
Wp=[11 19]*pi/2/fs;
Ws=[8 22]*pi/2/fs;
Rp=1;
Rs=50;
[n, Wn]=cheblord(Wp, Ws, Rp, Rs)
[b, a]=cheby1(n, Rp, Wn);
figure;
freqz(b, a, 256, fs);

set(gcf, 'position', [200, 200, 450, 400]);
y=filter(b, a, x);
figure;
subplot(121);
plot(t, x);
title('原始信号');
subplot(122);
plot(t, y);
title('滤波后的信号');
set(gcf, 'position', [200, 200, 800, 300]);

```



## 非参数法功率谱估计

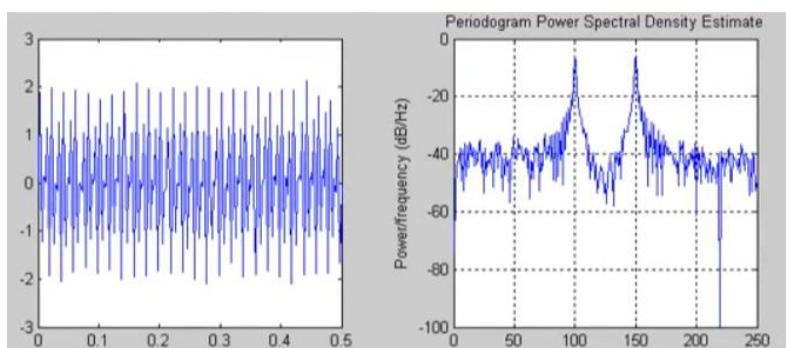
功率谱估计是基于有限的数据寻找信号、随机过程或系统的频率成份。它表示随机信号频域的统计特性，有着明显的物理意义，是信号处理的重要研究内容之一。由于实际中得到的随机信号长度总是有限的，用这种有限长度信号所得到的功率谱只是随机信号真实功率谱的一种估计，所以称其为功率谱估计。周期图法、**Bartlett**法、**Welch** 法、**Nuttall**法，以及最大熵谱估计等属于经典功率谱估计，以傅立叶变换为基础。下面介绍周期图法功率谱估计和**Welch**法功率谱估计。

### 周期图法功率谱估计

周期图法是最简单的非参数模型法，把随机序列 $x(n)$ 的N个观测数据直接进行离散傅立叶变换，然后取其幅值的平方，并除以N，作为序列 $x(n)$ 真实功率谱的估计。

采用函数**periodogram( )**进行周期图法功率谱估计。

```
clear all;
close all;
fs=500;
t=0:1/fs:0.5;
x=sin(2*pi*t*100)+sin(2*pi*t*150)+0.1*r;
figure;
subplot(121);           I
plot(t, x);
subplot(122);
periodogram(x, [], 'onesided', 512, fs);
set(gcf, 'position', [200, 200, 800, 300]);
```



### Welch 法功率谱估计

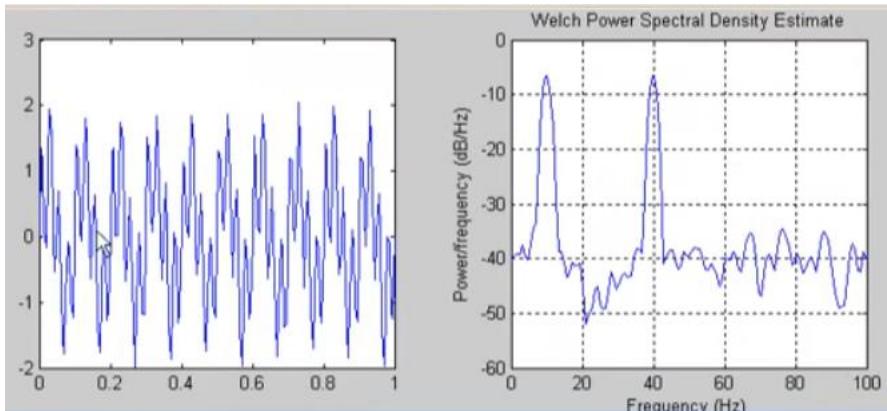
**Welch**法功率谱估计在周期图法的基础上进行了改进，是一种广泛应用的经典功率谱估计方法。**Welch**法功率谱估计能够改善谱失真现象。

采用函数**pwelch( )**进行**Welch**法功率谱估计。

```

clear all;
close all;
fs=200;
t=0:1/fs:1;
x=sin(2*pi*t*10)+sin(2*pi*t*40)+0.1*randn(t);
figure;
subplot(121);
plot(t,x);
subplot(122);
pwelch(x,[],120,256,fs);
set(gcf,'position',[200,200,800,300]);

```



## 参数模型法功率谱估计

参数模型谱估计的方法不是直接的进行功率谱的计算，而是假设随机信号服从一个模型。通过有限的数据记录，对信号模型的参数进行估计，通过模型参数得到信号的功率谱。参数模型法功率谱估计包括：**AR**模型、**MA**模型和**ARMA**模型等。

**AR**模型、**MA**模型和**ARMA**模型，从数学逼近的角度来讲，三者之间可以互相转换。估计**ARMA**或**MA**模型参数一般需要解一组非线性方程，而估计**AR**模型通常只需要解一组线性方程，因此，**AR**模型得到了深入研究和广泛应用

### AR 模型功率谱估计

**AR**模型（又称为自回归模型）是一个全极点模型，可用如下的差分方程表示：

$$x(n) = -\sum_{i=1}^p a_p(i)x(n-i) + \varepsilon(n)$$

其中  $\varepsilon(n)$  为均值为0，方差为  $\sigma^2$  的白噪声序列， $p$  为**AR**模型的阶数。

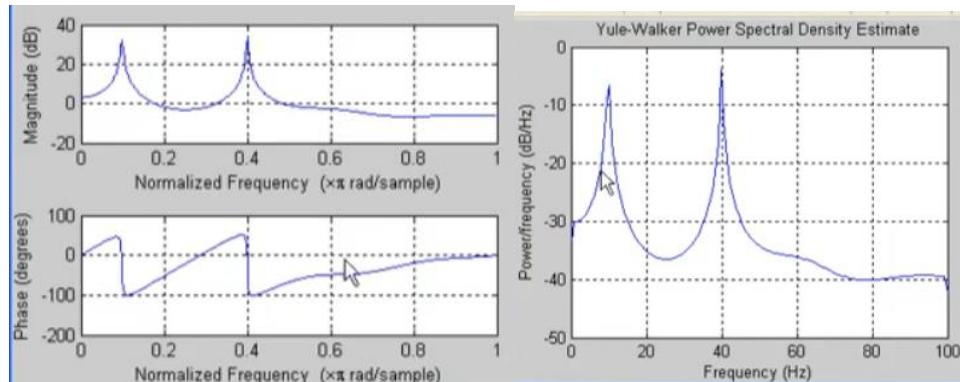
## Yule-Walker 法功率谱估计

在MATLAB中，通过函数`aryule( )`来估计AR模型的参数，该函数的调用格式为：`[a, e]=aryule(data, order)`，其中`x`为输入信号，`order`为AR模型的阶数，`a`为AR模型的参数，`e`为方差。



利用函数`pyulear( )`可以估计AR模型的功率谱。该函数首先采样Yule-Walker方法估计AR模型的参数，然后估计功率谱值。

```
clear all;
close all;
fs=200;
t=0:1/fs:1;
x=sin(2*pi*t*10)+sin(2*pi*t*40)+0.1*randn(t);
[a, e]=aryule(x, 8);
figure;
freqz(1, a);
set(gcf, 'position', [200, 200, 400, 300]);
figure;
pyulear(x, 8, 256, fs);
set(gcf, 'position', [200, 200, 400, 300]);
```



## Burg 法功率谱估计

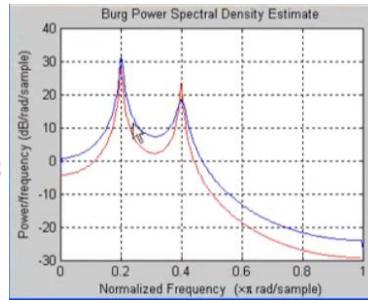
在MATLAB中，通过函数`arburg( )`来估计AR模型的参数，该函数的调用格式为：`[a, e]=arburg(data, order)`，其中`x`为输入信号，`order`为AR模型的阶数，`a`为AR模型的参数，`e`为方差。

Burg法估计功率谱又称为最大熵谱估计。函数`pburg( )`采用Burg法估计AR模型的功率谱。

```

clear all;
close all;
a=[1 -2.2137 2.9403 -2.1679 0.9606];
figure;
[h, w]=freqz(1, a, 256);
plot(w/pi, 20*log10(2*abs(h)/(2*pi)), 'r');
hold on;
x=filter(1, a, randn(256, 1));
pburg(x, 4, 512);
set(gcf, 'position', [200, 200, 400, 300]);

```



## 协方差法功率谱估计

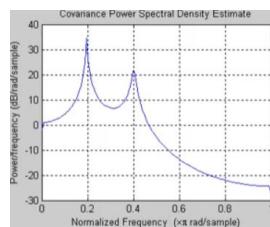
在MATLAB中，函数`arcov()`采用协方差方法来估计功率谱，该函数的调用格式为：`[a, e]=arcov(x, p)`，其中`x`为输入信号，`p`为阶数，`a`为AR模型的系数，`e`为方差。

函数`pcov()`采用协方差方法来估计AR模型的功率谱，该函数的调用方式和函数`pbburg()`完全相同。

```

clear all;
close all;
a=[1 -2.2137 2.9403 -2.1679 0.9606];
rand('state', 1);
x=filter(1, a, randn(256, 1));
figure;
pcov(x, 4);    I
set(gcf, 'position', [200, 200, 400, 300]);

```



## 子空间法功率谱估计

子空间法功率谱估计是一种现代功率谱估计方法，主要包括：特征向量法和MUSIC法。该方法以自相关矩阵的特征分解为基础，将自相关矩阵的特征向量空间分解为信号子空间和噪声子空间。

## 特征向量法功率谱估计

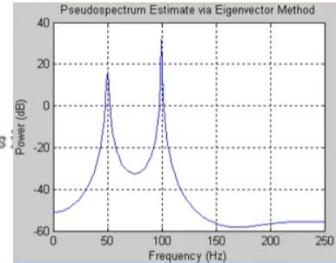
特征向量法主要适合混有高斯白噪声的正弦信号的功率谱估计。

在MATLAB中，采用函数`peig()`进行特征向量法的功率谱估计

```

clear all;
close all;
fs=500;
t=0:1/fs:0.5;
x=sin(2*pi*t*50)+sin(2*pi*t*100)+0.1*randn(s);
figure;
peig(x, 4, [], fs);
set(gcf, 'position', [200, 200, 400, 300]);

```



## MUSIC 算法功率谱估计

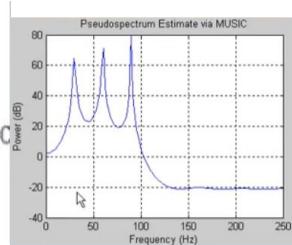
**MUSIC** (multiple signal classification, MUSIC) 是多重信号分类的英文缩写。MUSIC算法功率谱估计，是基于矩阵特征分解的一种非参数模型的现代谱估计方法。  
MUSIC算法的基本思想使通过特征分解把信号从噪声中分离出来。

在MATLAB中，函数pmusic( )利用MUSIC算法进行功率谱估计。

```

clear all;
close all;
fs=500;
t=0:1/fs:0.5;
x=sin(2*pi*t*30)+sin(2*pi*t*60)+sin(2*pi*t*90);
figure;
pmusic(x, 6, [], fs);
set(gcf, 'position', [200, 200, 400, 300]);

```



## 本章小结

本章详细介绍了MATLAB的信号处理工具箱的部分内容，主要包括常用信号的产生，改变信号的采样率，FIR和IIR滤波器的设计，以及功率谱估计等。这些都是信号处理工具箱中最基本的内容，需要熟练掌握，才能进行更进一步的深入研究。

# 小波分析工具箱

本章将对MATLAB的小波分析工具箱进行详细的介绍，首先介绍了小波分析的基本理论，主要包括连续小波变换、离散小波变换、多分辨分析和小波包变换等。然后，重点介绍了一维小波变换、二维小波变换和小波包变换，最后详细介绍了小波分析工具箱的图形界面工具。

## 小波分析

小波分析克服了短时傅立叶变换在单分辨率上的缺陷，具有多分辨率分析的特点。下面对小波分析的基本理论进行介绍，包括连续小波变换、离散小波变换、多分辨分析和小波包分析，最后介绍在小波分析中常用的小波。

### 小波分析简介

小波变换采用随频率改变的时间一频率窗口，是进行信号时频分析和处理的理想工具。在利用小波分析信号分析时，在低频部分采用较低的时间分辨率，提高频率分辨率；在高频部分，采用较低的频率分辨率来换取精确的时间定位。小波包分解与小波分解相比，是一种更精细的分解方法，不仅对低频部分进行分解，对高频部分也进行分解。

小波变换速度快，适合信号的在线分析。小波分析能够通过变换充分突出问题某些方面的特征，因此，小波变换在许多领域都得到了成功的应用。

### 连续小波变换

小波变换的实质是将信号在一个时域和频域上均具有局部化性质的平移伸缩小波权函数进行卷积，从而将信号分解成位于不同时间和频率上的各个成份。设 $\psi(t)$ 的傅立叶变换为 $\psi(w)$ ，满足允许条件：

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < +\infty$$

### 离散小波变换

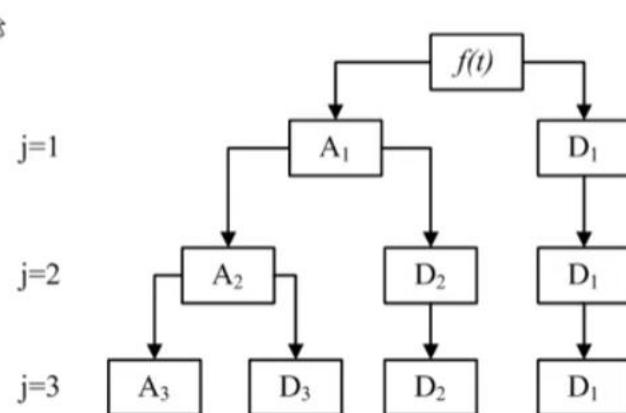
离散小波变换（discrete wavelet transform, DWT）是指对尺度因子 $a$ 和平移因子 $b$ 进行离散化，而不是时间的离散化。离散小波变换的一个重要问题是如何降低计算量和数据量，因为如果对尺度因子 $a$ 和平移因子 $b$ 离散的间隔小，那么计算量和数据量都是相当惊人的。

## 多分辨分析

离散小波变换的一个突破性成果是Mallat于1989年在多分辨分析的基础上提出的快速算法：Mallat算法。Mallat算法在小波分析中的作用相当于快速傅立叶变换（FFT）在傅立叶分析中的作用。Mallat算法由小波滤波器H、G和h、g对信号进行分解和重构。Mallat分解算法为：

$$\begin{cases} A_0[f(t)] = f(t) \\ A_j[f(t)] = \sum_k H(2t - k) A_{j-1}[f(t)] \\ D_j[f(t)] = \sum_k G(2t - k) A_{j-1}[f(t)] \end{cases}$$

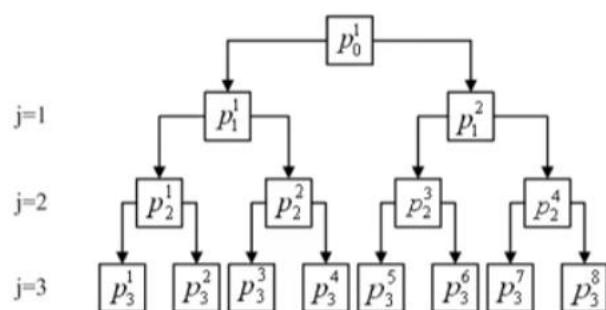
对信号  $f(t)$  进行离散小波的3层分解，近似系数和细节系数。



## 小波包分解

小波包分解的快速算法为：

$$\begin{cases} p_0^1(t) = f(t) \\ p_j^{2i-1} = \sum_k H(k - 2t) p_{j-1}^i(t) \\ p_j^{2i} = \sum_k G(k - 2t) p_{j-1}^i(t) \end{cases}$$



## 常用的小波

在MATLAB的命令行窗口输入：`help wavelet`，可以查询MATLAB的小波工具箱中的所有函数，以及小波工具箱的版本。在MATLAB 2010a版本中小波工具箱的版本为4.5。

用户在命令行窗口输入：**wavedemo**，可以查看例子程序。

采用函数**wavemngr( )**可以获取所有的小波。通过函数**waveinfo( )**可以获取小波的信息。

## 一维小波分解和重构

下面对一维小波的分解和重构进行介绍，包括一维连续小波的分解和重构、一维离散小波的单层分解和重构，以及离散小波的多层分解和重构。

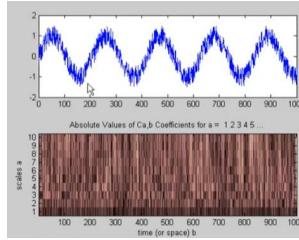
### 一维连续小波分解

在MATLAB中，采用函数**cwt( )**进行一维连续小波分解，该函数的常用调用格式为：

**coefs=cwt(s, scales, 'wname')**: 该函数对信号s进行尺度为scales的连续小波分解，小波为wname，返回值coefs为系数。

**coefs=cwt(s, scales, 'wname', 'plot')**: 该函数通过参数plot显示变换后的图形。

```
clear all;
close all;
load noissin.mat;
figure;
subplot(211);
plot(noissin);
subplot(212);
c=cwt(noissin,1:10,'db2','plot');
```



### 一维离散小波分解和重构

在MATLAB中，采用函数**dwt( )**进行一维小波的单层分解，该函数的常用调用格式为：

**[cA, cD]=dwt(X, 'wname')**: 该函数采用小波wname进行单层分解，cA为近似系数，cD为细节系数。

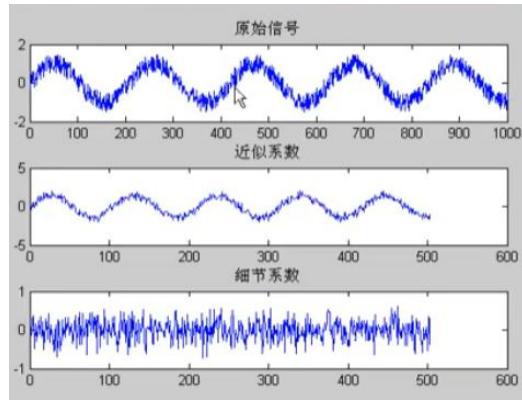
**[cA, cD]=dwt(X, 'wname', 'mode', MODE)**: 该函数设定扩展模式为MODE。

在MATLAB中，采用函数**wavdec( )**进行一维小波的多层次分解。

```

clear all;
close all;
load noissin.mat;
[cA, cD]=dwt(noissin, 'sym4');
figure;
subplot(311);
plot(noissin);
title('原始信号');
subplot(312);
plot(cA);
title('近似系数'); I
subplot(313);
plot(cD);
title('细节系数');
set(gcf, 'position', [50, 50, 600, 400]);

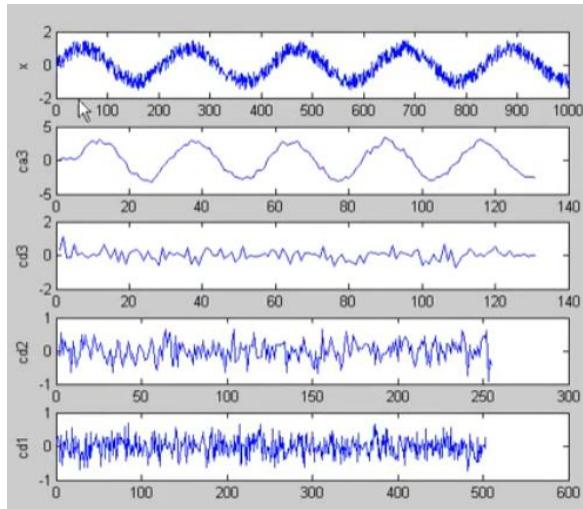
```



```

clear all;
close all;
load noissin;
x=noissin;
[C, L]=wavedec(x, 3, 'db4');
[cd1, cd2, cd3]=detcoef(C, L, [1, 2, 3]);
ca3=appcoef(C, L, 'db4', 3);
figure;
subplot(511);
plot(x);
ylabel('x');
subplot(512);
plot(1:L(1), ca3);
ylabel('ca3');
subplot(513);
plot(1:L(2), cd3);
ylabel('cd3');
subplot(514);
plot(1:L(3), cd2);
ylabel('cd2');
subplot(515);
plot(1:L(4), cd1);

```



在MATLAB中，利用函数idwt( )进行单层小波重构，该函数的调用格式为： $X=idwt(cA, cD, 'wname')$ ， $cA$ 为近似系数， $cD$ 为细节系数， $wname$ 为采用的小波。利用函数wavecrec( )进行多层小波的重构。

在进行小波的重构时，小波的类型必须和分解时保持一致。

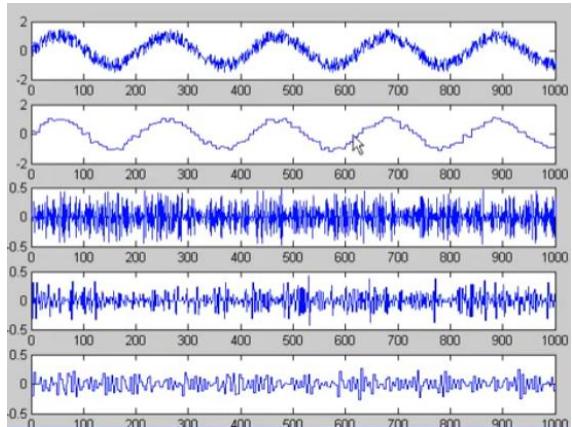
```

clear all;
close all;
load noissin.mat;
x=noissin;
[C, L]=wavedec(x, 3, 'dbl'); | e =
y=waverec(C, L, 'dbl');
e=max(abs(x-y))| 8.8818e-016

clear all;
close all;
load noissin.mat;
x=noissin;
[C, L]=wavedec(x, 3, 'dbl');
A=wrcoef('a', C, L, 'dbl');
D1=wrcoef('d', C, L, 'dbl', 1);
D2=wrcoef('d', C, L, 'dbl', 2);
D3=wrcoef('d', C, L, 'dbl', 3);
figure;
subplot(511); I
plot(x);
subplot(512);

plot(A);|
subplot(513);
plot(D1);
subplot(514);
plot(D2);
subplot(515);
plot(D3);
set(gcf, 'position', [30, 30, 700, 500]);

```



## 二维小波分解和重构

二维小波非常适合进行图像的分析和处理。图像的单层二维小波分解，将图像分解为4个原图 $1/4$ 大小的图像，左上的图像为两个维度都采用低通滤波后的结果；右上的图像为横向采用低通滤波，纵向采用高通滤波后的结果；左下的图像为横向采用高通滤波，纵向采用低通滤波后的结果；右下的图像两个维度都采用高通滤波后的结果。

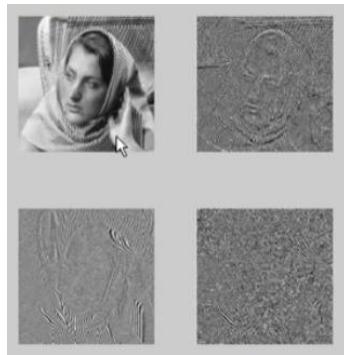
## 二维小波的单层分解和重构

在MATLAB中，采用函数dwt2( )进行二维小波的单层分解，该函数的调用格式为：[cA, cH, cV, cD]=dwt2(X, 'wname')，该函数采用小波wname，对信号X进行单层分解。其中cA为近似系数，cH、cV和cD分别是水平细节系数、垂直细节系数和对角细节系数。

MATLAB中，采用函数idwt2( )进行二维小波的单层重构，该函数的调用格式为：X=idwt2(cA, cH, cV, cD, 'wname')。

```
clear all;
close all;
load woman.mat;
Y=ind2gray(X, map);
[cA, cH, cV, cD]=dwt2(Y, 'db1');
figure;
subplot(221);
imshow(cA, []);
subplot(222);
imshow(cH, []);
subplot(223);
imshow(cV, []);
subplot(224);
imshow(cD, []);

set(gcf, 'position', [50, 50, 400, 400]);
```

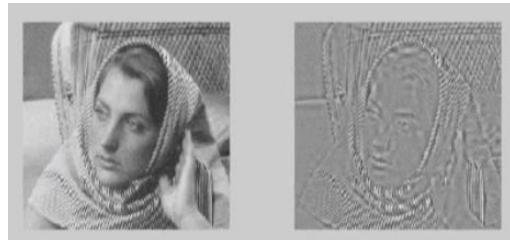


## 二维小波的多层分解和重构

在MATLAB中采用函数wavedec2( )进行二维小波的多层次分解，该函数的调用形式为：[C, S]=wavedec2(X, N, 'wname')，该函数采用小波wanme对信号X进行二维小波的N层分解。

利用函数waveenc2( )进行二维小波的多层次重构，该函数的调用格式为：X=waveenc2(C, S, 'wname')，利用小波wname进行二维小波的多层次重构。

```
clear all;
close all;
load woman.mat;
X=ind2gray(X, map);
[C, S]=wavedec2(X, 3, 'db4');
S
C(1:38*38)=0;
Y=waveenc2(C, S, 'db4');
figure;
subplot(121);
imshow(X, []);
subplot(122);
imshow(Y, []);
set(gcf, 'position', [50, 50, 600, 300]);
```



```

clear all;
close all;
load woman.mat; I
X=ind2gray(X, map);
[C, S]=wavedec2(X, 2, 'db1');
sizeX=size(X)
sizeC=size(C)
S
[chd2, cvd2, cdd2]=detcoef2('all', C, S, 2);
ca2=appcoef2(C, S, 'db1', 2);
sizeCA2=size(ca2)
figure;
subplot(221);imshow(ca2, []);
subplot(222);imshow(chd2, []);
subplot(223);imshow(cvd2, []);
subplot(224);imshow(cdd2, []);

set(gcf, 'position', [50, 50, 400, 400]);

```

---

```

clear all;
close all;
load woman.mat;
X=ind2gray(X, map);
[C, S]=wavedec2(X, 2, 'db4');
A=wrcoef2('a', C, S, 'db4', 2);
H=wrcoef2('h', C, S, 'db4', 2);
V=wrcoef2('v', C, S, 'db4', 2);
D=wrcoef2('d', C, S, 'db4', 2); I
figure;
subplot(221);imshow(A);
subplot(222);imshow(H, []);
subplot(223);imshow(V, []);
subplot(224);imshow(D, []);

set(gcf, 'position', [50, 50, 400, 400]);

```



## 小波包分析

小波分解只在低频部分进行分解，小波包分解和小波分解不同，不仅在低频部分进行分解，在高频部分也进行分解。  
下面介绍MATLAB中的小波包分析。

### 一维小波包的分解和重构

在MATLAB中，采用函数`wpdec()`进行一维小波包分解，

该函数的调用格式为：

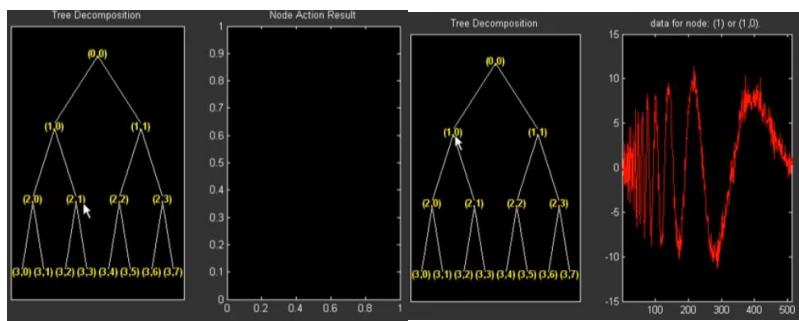
`T=wpdec(X, N, 'wname')`: 该函数对信号X进行N层的小波包分解，采用的小波为wname，返回值T为小波包树。

`T=wpdec(X, N, 'wname', E, P)`: 该函数采用参数E设置采样的熵，默認為'Shannon'。

```

clear all;
close all;
load noisdopp;
x=noisdopp;
wpt=wpdec(x, 3, 'db1', 'shannon')
plot(wpt);

```



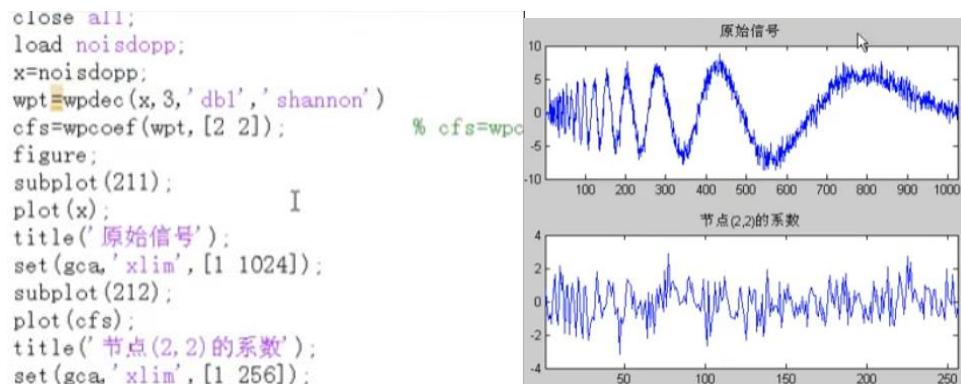
在MATLAB中，通过函数`wpccoef()`获取小波树上某个节点的小波包系数。

在MATLAB中，采用函数`wpsplt()`将小波树上的节点进一步分解。

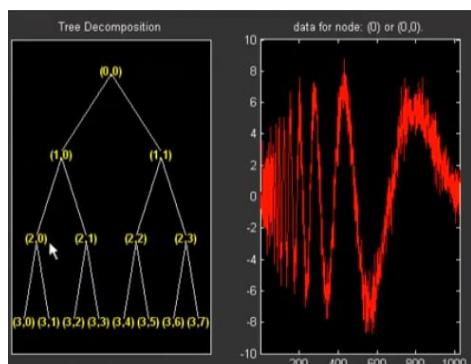
在MATLAB中，采用函数`wpjoin()`进行小波包树上的节点进行合并。

在MATLAB中，采用函数`besttree()`获取最优小波树。

在MATLAB中，通过函数`wprec()`进行一维小波包的重构，该函数的调用格式为：`X=wprec(T)`，其中T为小波包树，返回值X为重构后的信号。

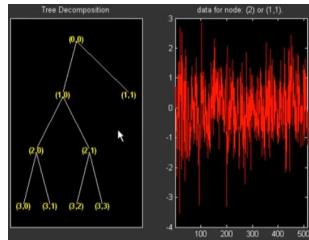


```
clear all;
close all;
load noisdopp;
x=noisdopp;
wpt1=wpdec(x, 3, 'dbl', 'shannon');
plot(wpt1); I
wpt2=wpsplt(wpt1, 9);
plot(wpt2);
```



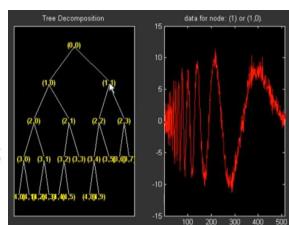
```

clear all;
close all;
load noisdopp;
x=noisdopp;
wpt1=wpdec(x, 3, 'dbl');
plot(wpt1);
wpt2=wpjoin(wpt1, [1 1]);
plot(wpt2);
```



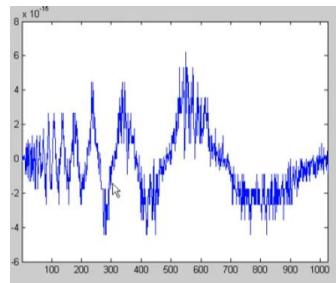
```

clear all;
close all;
load noisdopp;
x=noisdopp;
wpt=wpdec(x, 4, 'dbl', 'shannon');
bt=besttree(wpt);
plot(bt);
```



```

clear all;
close all;
load noisdopp;
x=noisdopp;
wpt=wpdec(x, 3, 'dbl', 'shannon');
y=wprec(wpt);
figure;
plot(x-y);
max(abs(x-y));
set(gca, 'xlim', [1 1024]);
```



## 二维小波的分解和重构

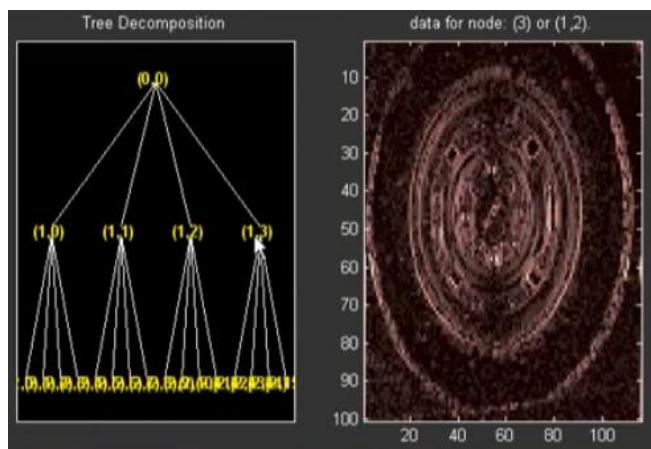
在MATLAB中，采用函数`wpdec2( )`进行二维小波包的分解，该函数的调用格式为：`T=wpdec2(X, N, 'wname')`，该函数采用`wname`小波，对数据`X`进行二维`N`层小波包分解。

同一维小波分解类似，采用函数`wpcoef( )`获取二维小波包树上某个节点的系数；采用函数`wpsplt( )`进行二维小波包树的进一步分解；利用函数`wpjoin( )`进行二维小波包树上节点的合并；利用函数`besttree( )`获取最优的二维小波包树。

通过函数`wprec2( )`进行二维小波包的重构，该函数的调用格式为：`X=wprec2(T)`，其中`T`为二维小波包树，`X`为重构后的二维数据。

```

clear all;
close all;
load tire.mat
t=wpdec2(X, 2, 'dbl');
plot(t);
```



## 小波能量和小波熵

在MATLAB中，采用函数wenergy( )计算进行归一化后的小波能量。该函数可以用于一维小波和小波包。该函数的调用格式为：

[Ea, Ed]=wenergy(C, L): 该函数用于计算一维小波的能量。  
E=wenergy(T): 该函数用于计算小波包的能量。

在MATLAB中，采用函数wentropy( )计算小波包的熵。

```
clear all;
close all;
load noisbump.mat
x=noisbump;
[C, L]=wavedec(x, 4, 'sym4');
[Ea, Ed]=wenergy(C, L)
T=wpdec(x, 2, 'sym4');
E=wenergy(T)

clear all;
close all;
x=rand(1, 100);
e1=wentropy(x, 'shannon')
e2=wentropy(x, 'log energy')
e3=wentropy(x, 'threshold', 0.2)
e4=wentropy(x, 'sure', 3)
e5=wentropy(x, 'norm', 1.3)

clear all;
close all;
Load noisbump.mat
x=noisbump;
[C, L]=wavedec(x, 4, 'sym4');
[Ea, Ed]=wenergy(C, L)
E1=wentropy([Ea, Ed], 'shannon')
E2=wentropy([Ea, Ed], 'log energy')
'
```

## 小波工具箱的 GUI 工具

下面介绍小波工具箱的另一种实现方式，即GUI工具。小波工具箱的GUI工具界面友好，在解决特定问题时非常的直观和灵活，提供了大量的例子程序，而且可以非常方便的进行数据的导入和导出。用户不用编写程序，就可以采用小波分析对一维信号或二维图像进行压缩和去除噪声等。

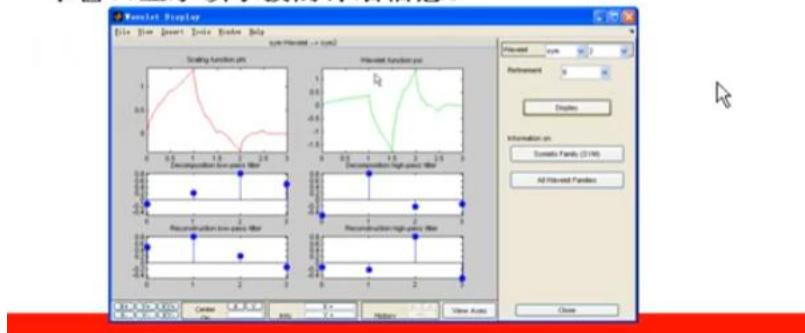
### 小波工具箱介绍

在MATLAB的命令行窗口输入： wavemenu，会出现小波工具箱的GUI界面如图20.24所示。小波工具箱的GUI提供源代码，高级用户可以根据自己的需要对源代码进行修改。用户可以在MATLAB的命令行窗口输入： edit wavemenu.m可以查看wavemenu工具的源代码。



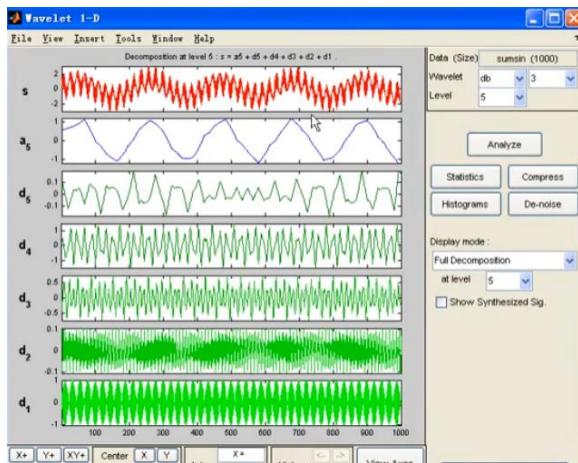
## 小波和小波包的显示工具

显示工具主要用来进行小波和小波包的显示。在小波工具箱的GUI界面中，单击Display中的按钮Wavelet Display，会出现小波显示界面，在右上角选择要显示的小波和小波的参数，然后单击按钮Display则会显示该小波函数和相应的滤波器，如图20.25所示。选择的小波为sym，参数为2，如果单击按钮Symlets Family (SYM)则会弹出一个窗口显示该小波的详细信息。



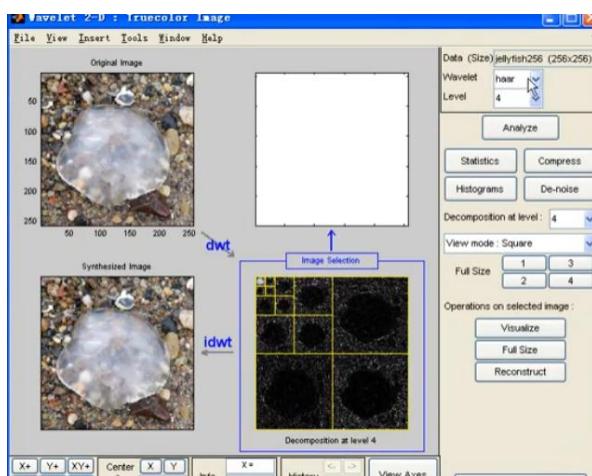
## 一维小波分析工具

在小波工具箱的GUI界面中，单击按钮Wavelet 1-D后，出现一维小波分解界面，单击File菜单下的Example Analysis子菜单中的with db3 at level 5 → sum of sines选项后，如图20.27所示。该例子程序自动调用数据文件sumsin.mat，采用db3小波进行5层分解。在图20.27中，可以对小波的类型和分解的层数等进行设置，然后单击按钮Analyze进行重新分解。



## 二维小波分析工具

在小波工具箱的GUI界面中，二维小波分析工具包括：二维小波分解（Wavelet 2-D）和二维小波包分解（Wavelet Packet 2-D）。下面介绍其中的二维小波分解。用户单击按钮Wavelet 2-D会出现二维小波分解窗口，单击File菜单下的Example Analysis菜单的子菜单TrueColor Images的At level 4 with haar --> Jelly Fish后，二维小波分解界面如图20.32所示。用户可以对小波的类型和分解的层数重新进行设置，然后单击按钮Analyze重新进行分解。同样，二维小波分析工具箱可以显示图像的统计信息和直方图，以及对图像进行压缩和去噪。



## 本章小结

本章首先介绍了小波分析的基本理论，包括连续小波变换、离散小波变换、多分辨分析和小波包变换等。然后介绍了利用MATLAB进行一维小波分析、二维小波分析和小波包分析等，这是本章的重点和难点。最后介绍了MATLAB中小波工具箱的GUI工具，可以非常简单和直观的对信号进行小波分解和重构，以及信号的压缩和去除噪声等。

# 图像处理工具箱

在MATLAB的图像处理工具箱（Image Processing Toolbox）提供了大量的用于图像处理的函数。本章首先对图像处理工具箱进行简单的介绍，然后详细的介绍了图像的类型、图像类型之间的转换、图像的读取和显示等基本操作。接着介绍了图像的常见操作、图像的变换、图像的增强和边缘检测等。最后介绍了利用MATLAB进行图像和视频的采集。

## 图像处理基础

首先对数字图像处理和MATLAB的图像处理工具箱进行了介绍，然后详细介绍了MATLAB中的图像类型和图像类型之间的转换。最后详细的介绍了在MATLAB中如何进行图像的读取、写入和显示，最后介绍了MATLAB的图像浏览器imtool。

## 数字图像处理

从物理和数学角度看，图像是记录物体辐射能量的空间分布，这个分布是空间坐标、时间坐标和波长的函数，即  $I=f(x,y,z,\lambda,t)$ ，这样的图像能被计算机处理，计算机图像处理即数字图像处理MATLAB的长处就是处理矩阵运算，因此用MATLAB处理数字图像非常方便。计算机图像处理是利用计算机对数字图像进行一系列操作，从而获得预期的结果的技术。

## 图像处理工具箱介绍

MATLAB对图像的处理功能主要集中在它的图像处理工具箱（Image Processing Toolbox）中。图像处理工具箱是由一系列支持图像处理操作的函数组成，可以进行诸如几何操作、线性滤波和滤波器设计、图像变换、图像分析与图像增强、图像注册以及形态学处理等图像处理操作。

## MATLAB 中图像类型

在MATLAB中，有4种基本的图像类型，分别是：

RGB真彩色图像（Truecolor Image）

索引图像（Indexed Image）

灰度图像（Grayscale Image）

二值图像（Binary Image）

## MATLAB 图像类型转换

在图像处理过程中，有时需要对图像的类型进行转换。例如，将索引图像转换为RGB图像，或将RGB图像转换为灰度图像等。常见的图像类型转换函数，如表21.1所示。

下面进行详细的介绍。

表 21.1 图像类型转换函数<sup>①</sup>

| 函数 <sup>②</sup>        | 说明 <sup>③</sup>                |
|------------------------|--------------------------------|
| dither <sup>④</sup>    | 图像抖动，将灰度图变成二值图，或将RGB图像抖动成索引图像。 |
| gray2ind <sup>⑤</sup>  | 将灰度图像转换为索引图像。                  |
| grayslice <sup>⑥</sup> | 通过设定阈值将灰度图像转换为索引图像。            |
| im2bw <sup>⑦</sup>     | 通过设定阈值将RGB、索引或灰度图像转换成二值图像。     |
| imd2gray <sup>⑧</sup>  | 将索引图像转换成灰度图像。                  |
| imd2rgb <sup>⑨</sup>   | 将索引图像转换成RGB图像。                 |
| mat2gray <sup>⑩</sup>  | 将数据矩阵转换成灰度图像。                  |
| rgb2gray <sup>⑪</sup>  | 将RGB图像转换成灰度图像。                 |
| rgb2ind <sup>⑫</sup>   | 将RGB图像转换成索引图像。                 |

```
clear all;
close all;
[X, map]=imread('trees.tif','tif');
RGB=ind2rgb(X, map);
figure;
imshow(RGB);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
RGB=imread('peppers.png');
[X, map]=rgb2ind(RGB, 16);
figure;
imshow(X, map);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
RGB=imread('peppers.png','png');
I=rgb2gray(RGB);
figure;
imshow(I);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
[X, map]=imread('trees.tif','tif');
newmap=rgb2gray(map);
figure;
imshow(X, newmap);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
X=imread('rice.png');
Y=grayslice(X, 16);
figure;
imshow(Y, jet(16));
set(gcf, 'position', [200, 200, 300, 300]);
```



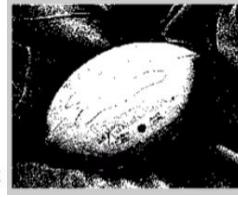
```
clear all;
close all;
[X, map]=imread('trees.tif','tif');
I=ind2gray(X, map);
figure;
imshow(I);
set(gcf, 'position', [200, 200, 300, 300]);
```



```

clear all;
close all;
RGB=imread('football.tif');
bw=im2bw(RGB, 0.3);
figure;
imshow(bw);
set(gcf, 'position', [200, 200, 400, 300]);

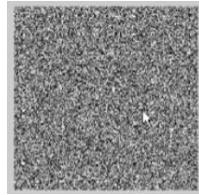
```



```

clear all;
close all;
A=rand(200, 200);
I=mat2gray(A);
figure;
imshow(I);
set(gcf, 'position', [200, 200, 400, 300]);

```



```

clear all;
close all;
I=imread('cameraman.tif');
BW=dither(I);
subplot(121);
imshow(I);
subplot(122);
imshow(BW)

```



## 图像的读写和显示

在MATLAB中，采用函数`imread()`和函数`imwrite()`分别进行图像的读取和写入。

采用函数`imfinfo()`获取图像文件的信息。

采用函数`imshow()`和函数`montage()`显示图像。此外，还可以将多帧图像转换为电影，下面分别进行介绍。

```

clear all;
close all;
info=imfinfo('onion.png','png')

info =

```

|                  |                        |
|------------------|------------------------|
| Filename:        | [1x42 char]            |
| FileModDate:     | '07-十一月-2009 20:01:52' |
| FileSize:        | 44638                  |
| Format:          | 'png'                  |
| FormatVersion:   | []                     |
| Width:           | 198                    |
| Height:          | 135                    |
| BitDepth:        | 24                     |
| ColorType:       | 'truecolor'            |
| FormatSignature: | [1x8 double]           |
| Colormap:        | []                     |
| Histogram:       | []                     |

```

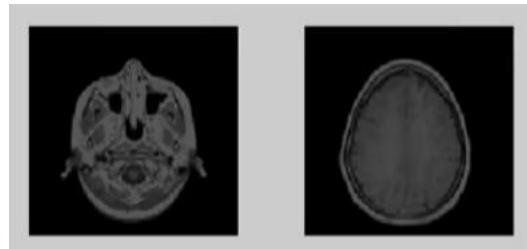
clear all;
close all;
RGB=imread('football.tif','tif');
imwrite(RGB,'myfootball.png','png');

```

```

clear all;
close all;
I1=imread('mri.tif',1);
I20=imread('mri.tif',20);
figure;
subplot(121);
imshow(I1); I
subplot(122);
imshow(I20)
set(gcf,'position',[200,200,500,300]);

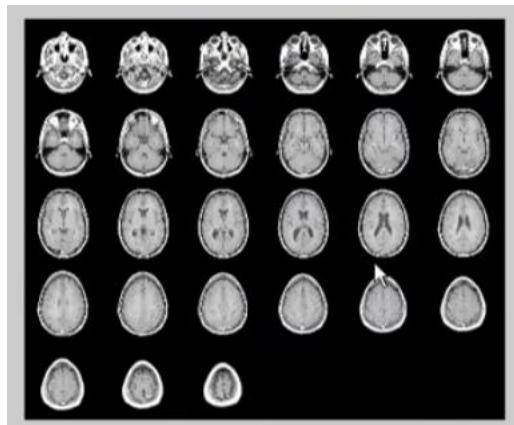
```



```

clear all;
close all;
mri=uint8(zeros(128,128,1,27));
for i=1:27
    [mri(:,:,i),map]=imread('mri.tif',i);
end
montage(mri,map);

```



```

clear all;
close all;
mri=uint8(zeros(128,128,1,27));
for i=1:27
    [mri(:,:,i),map]=imread('mri.tif',i);
end
mov=immovie(mri,map);
implay(mov);

```

//哦吼，电影!!!

## 图像浏览器

在MATLAB中，采用imtool进行图像的显示。在MATLAB以前的版本中，采用函数imview( )，函数imview( )将会逐渐被函数imtool( )替代。

## 图像的常用操作

下面介绍图像处理中经常用到的图像基本运算、图像的直方图均衡化、对比度增强、图像的旋转、图像中添加噪声、图像的剪切等。这些都是图像处理中最常用的基本操作。

## 图像基本运算

在利用**MATLAB**进行图像处理时，由于图像数据类型为**unit8**，而在矩阵运算中要求所有的运算变量为**double**型。因此必须将图像数据转换成双精度型数据。在**MATLAB**中，通过函数**im2double()**将图像数据转换为双精度浮点型。

两幅图像的加法运算

两幅图像的减法运算

两幅图像的乘法运算

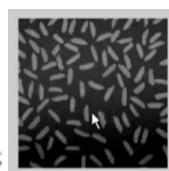
两幅图像的除法运算

两幅图像的绝对差异

```
clear all;
close all;
I=imread('rice.png');
J=imread('cameraman.tif');
K=imadd(I, J, 'uint16');
imshow(K, []);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
I=imread('rice.png');
J=imsubtract(I, 60);
imshow(J);
set(gcf, 'position', [200, 200, 300, 300]);
```



```
clear all;
close all;
I=imread('moon.tif', 'tif');
J=immultiply(I, 0.5);
subplot(121);
imshow(I);
subplot(122);
imshow(J);
set(gcf, 'position', [200, 200, 600, 300]);
```



```
clear all;
close all;
X=uint8([ 255 10 75; 44 225 100]);
Y=uint8([ 50 20 50; 50 50 50 ]);
Z=imdivide(X, Y)

clear all;
close all;
I=imread('cameraman.tif');
J=uint8(filter2(fspecial('gaussian'), I));
K=imabsdiff(I, J);
subplot(121);
imshow(I);
subplot(122);
imshow(K, []);
set(gcf, 'position', [200, 200, 600, 300]);
```



## 图像的直方图和直方图均衡化

在MATLAB中，采用函数imhist( )计算和显示图象的直方图  
，该函数的调用格式为：

imhist(I): 该函数绘制灰度图像的直方图。

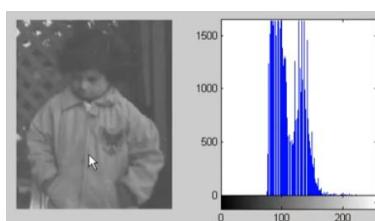
imhist(I, n): 该函数指定灰度级的数目为n。

imhist(X, map): 该函数绘制索引图像的直方图。

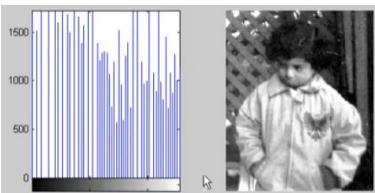
[counts, x]=imhist(...): 该函数返回直方图的数据，通过  
函数stem(x, counts)可以绘制直方图。

在MATLAB中，采用函数histeq( )实现直方图的均衡化。该  
函数可以用于灰度图像和索引图像。

```
clear all;
close all;
I=imread('pout.tif');
subplot(121);
imshow(I);
subplot(122);
imhist(I);
set(gcf,'position',[200, 200, 600, 300]);
```



```
clear all;
close all;
I=imread('pout.tif');
J=histeq(I);
imhist(J);
subplot(122);
imshow(J);
set(gcf,'position',[200, 200, 600, 300]);
```



## 图像的对比度增强

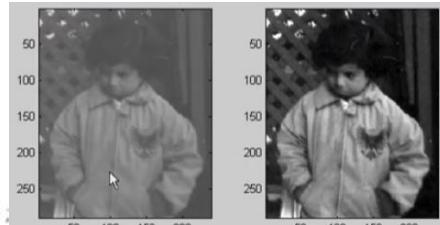
在MATLAB中，通过函数imadjust( )进行图像的对比度增  
强。该函数的调用格式为：

J=imadjust(I): 该函数对灰度图像I进行对比度增强。

J=imadjust(I, [low\_in; high\_in], [low\_out; high\_out]):  
该函数中[low\_in; high\_in]为原图象中要变换的灰度范  
围，[low\_out; high\_out]变换后的灰度范围。

RGB2=imadjust(RGB1, ...): 该函数对RGB图像进行对比  
度增强。

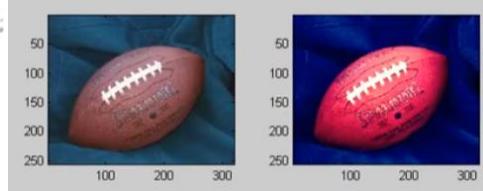
```
clear all;
close all;
I=imread('pout.tif');
J=imadjust(I, [0.3 0.7], []);
subplot(121);
subimage(I);
subplot(122);
subimage(J); I
set(gcf,'position',[200, 200, 600, 300]);
```



```

clear all;
close all;
RGB1=imread('football.jpg');
RGB2=imadjust(RGB1,[0.2 0.3 0.6 0.7 1],[]);
subplot(121);
subimage(RGB1);
subplot(122); I
subimage(RGB2);
set(gcf,'position',[200,200,600,300]);

```



## 图像的插值

插值是常用的数学运算，通常是利用曲线拟合的方法，通过离散的采样点建立一个连续函数来逼近真实的曲线，用这个重建的函数便可以求出任意位置的函数值。在 MATLAB 中通过插值，可以实现图像的缩放和旋转。

在 MATLAB 中，函数 `imresize()` 采用插值的方法来改变图像的大小。

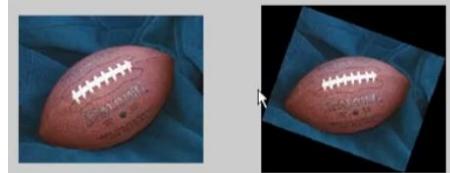
在 MATLAB 中，采用函数 `imrotate()` 进行图像的旋转。

```

clear all;
close all;
[X, map]=imread('trees.tif','tif');
[Y, newmap]=imresize(X, map, 0.5);
figure;
subimage(X, map);
figure;
subimage(Y, newmap);

clear all;
close all;
A=imread('football.jpg','jpg');
B=imrotate(A, -20, 'nearest');
figure;
subplot(121); I
imshow(A);
subplot(122);
imshow(B);
set(gcf,'position',[200,200,600,300]);

```



## 图像中添加噪声

在 MATLAB 中，可以通过函数 `imnoise()` 给图像添加噪声，该函数的调用格式为：

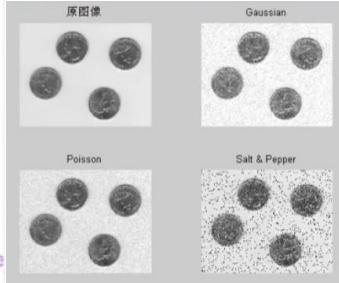
**J=imnoise(I, type):** 该函数对图像 I 添加类型为 type 的噪声。  
参数 Type 对应的噪声类型如下：'gaussian' 为高斯白噪声；'localvar' 为 0 均值白噪声；'poisson' 为泊松噪声；'salt & pepper' 为椒盐噪声；'speckle' 为乘性噪声

**J=imnoise(I, type, parameters):** 该函数对类型为 type 的噪声的属性进行设置。

```

clear all,close all;
I=imread('eight.tif','tif');
J1=imnoise(I,'gaussian',0.1); I
J2=imnoise(I,'poisson');
J3=imnoise(I,'salt & pepper',0.1);
figure;
subplot(221),imshow(I),title('原图像')
subplot(222),imshow(J1),title('Gaussian')
subplot(223),imshow(J2),title('Poisson')
subplot(224),imshow(J3),title('Salt & Pepper')

```



## 图像剪切

在MATLAB中，通过函数`imcrop()`实现图像的剪切，获取图像的一部分（矩形区域）。关于函数`imcrop()`的详细调用格式，用户可以查阅MATLAB的帮助系统。

```

clear all;
close all;
I=imread('football.jpg','jpg');
J=imcrop(I,[40,40,120,150]);
figure;
imshow(I);
figure;
imshow(J);

```

## 图像的变换技术

在数字图像处理中，图像的变换包括：二维快速傅立叶变换和反变换、二维离散余弦变换和反变换。下面分别进行介绍。

### 图像的傅里叶变换

在MATALB中，采用函数`fft2()`计算图像的二维快速傅立叶变换，该函数的调用格式为：

`Y=fft2(X)`: 该函数计算图像数据X的二维傅立叶变换。

`Y=fft2(X, m, n)`: 该函数通过补0，来指定数据的大小。

```

clear all;
close all;
I=imread('cameraman.tif','tif');
J=fft2(I,256,256);
K1=fftshift(J);
K2=log(abs(K1));
figure;
subplot(121);
imshow(I);
subplot(122);
imshow(K2,[]);
set(gcf,'position',[200,200,600,300]);

```



```

clear all;
close all;
I=imread('cameraman.tif','tif');
J=fft2(I, 256, 256);
K=ifft2(J, 256, 256);
figure;
subplot(121);I
imshow(log(abs(J)), []);
subplot(122);
imshow(K, []);
set(gcf, 'position', [200, 200, 600, 300]);

```



## 图像的离散余弦变换

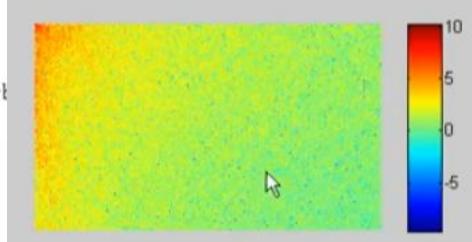
离散余弦变换（Discrete Cosine Transform）是采用不同频率和幅值的余弦函数来逼近图像。离散余弦变换常用来进行图像的压缩，例如JPEG格式的图像就采用了离散余弦变换进行压缩。

在MATLAB中，采用函数dct2()进行二维离散余弦变换，采用函数idct2()进行二维离散余弦变换的反变换。

```

clear all;
close all;
RGB=imread('autumn.tif');
I=rgb2gray(RGB);
J=dct2(I);
figure;
imshow(log(abs(J)), [], colormap(jet), color
J(abs(J)<10)=0;
K=idct2(J);
figure;
imshow(K,[0 255]);
% set(gcf, 'position', [200, 200, 700, 300]);

```



## 图像增强技术

数字图像的增强是图像处理中的一个重要研究内容之一，是图像处理的一项基本技术。下面介绍如何获取图像的像素值及其统计，最后介绍通过图像的数字滤波进行图像的增强。

### 像数值及其统计

在MATLAB中，采用函数impixel()来获取图像的像素值。  
该函数的常用调用格式为：

**p=impixel(l):** 该函数通过鼠标单击获取灰度图像中一点的像素值。

**p=impixel(X, map):** 该函数通过鼠标单击获取索引图像中一点的像素值。

**p=impixel(RGB):** 该函数通过鼠标单击获取RGB图像中一点的像素值。

**p=impixel(l, c, r):** 该函数获取灰度图像中，行为c，列为r的像素点的像素值。

**p=impixel(X, map, c, r):** 该函数获取索引图像中，行为c，列为r的像素点的像素值。

**p=impixel(RGB, c, r):** 该函数获取RGB图像中，行为c，列为r的像素点的像素值。

```

clear all;
RGB=imread('peppers.png') | p =
r=[10 20 30 50];
c=[30 50 70 85];
p=impixel(RGB, r, c)

```

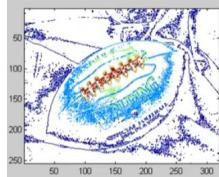
在MATLAB中，利用函数`impixelinfo()`获取图像中任意点的像素值。此外，利用函数`imcontour()`可以绘制灰度图的等高线。

```

clear all;
close all;
imshow('football.jpg');
h=impixelinfo;
set(h, 'position', [200 320 330 20]);

clear all;
close all;
RGB=imread('football.jpg');
I=rgb2gray(RGB);
figure;
imcontour(I);
set(gcf, 'position', [200, 200, 400, 350]);

```



## 图像的滤波

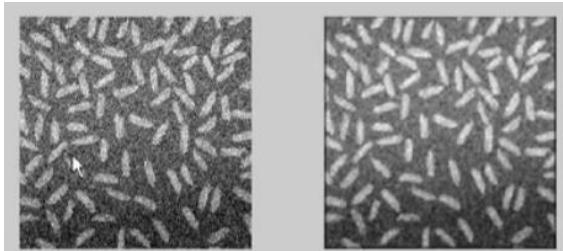
对于含有噪声的图像，可以对图像进行滤波，使图像变的更清晰。常用的滤波方法有邻域平均法、中值滤波法和自适应滤波法。

在MATLAB中，采用函数`medfilt2()`对图像进行中值滤波。  
采用函数`wiener2()`进行自适应滤波，根据图像的局部均值和方差进行自动调整，而且该函数还可以估计噪声的类型。

```

close all;
I=imread('rice.png');
I=imnoise(I, 'Gaussian', 0, 0.02);
h=[1 1 1;1 1 1;1 1 1];
h=h/9;
J=conv2(I, h);
figure;
subplot(121);
imshow(I);
subplot(122);
imshow(uint8(J), []);
set(gcf, 'position', [200, 200, 600, 300]);

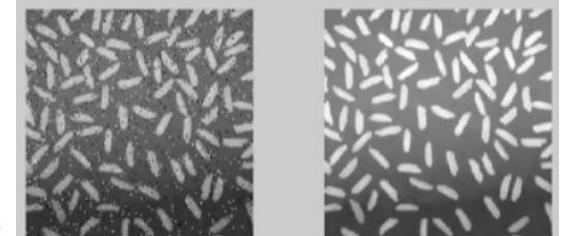
```



```

clear all;
close all;
I=imread('rice.png');
I=imnoise(I, 'salt & pepper', 0.03);
J=medfilt2(I, [3 3]);
figure;
subplot(121);
imshow(I);
subplot(122);
imshow(uint8(J), []);
set(gcf, 'position', [200, 200, 600, 300]);

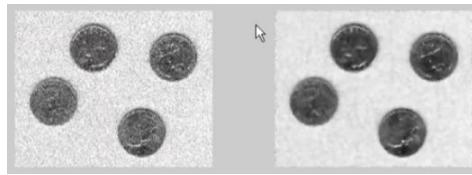
```



```

clear all;
close all;
I=imread('eight.tif');
I=imnoise(I,'Gaussian',0,0.01);
J=wiener2(I,[5 5]);
figure;
subplot(121);
imshow(I);
subplot(122);
imshow(uint8(J),[]);
set(gcf,'position',[200,200,700,300]);

```



## 图像的边缘检测

在进行图像的分析和处理时，图像的边缘包含许多重要的信息，可以利用边缘检测来对图像进行分割。

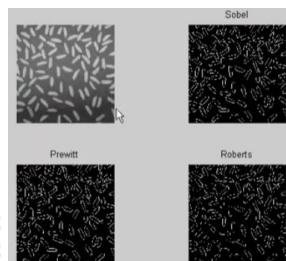
在MATLAB中，采用函数edge()来对图像的边缘进行检测

。在进行边缘检测时，常用的算子有Sobel算子、Prewitt算子、Roberts算子、LOG算子和Canny算子等。该函数的详细调用形式，用户可以查阅MATLAB的帮助系统。

```

clear all; close all;
I=imread('rice.png');
J1=edge(I,'Sobel');
J2=edge(I,'prewitt');
J3=edge(I,'Roberts');
figure;
subplot(221), imshow(I);
subplot(222), imshow(J1); title('Sobel');
subplot(223), imshow(J2); title('Prewitt');
subplot(224), imshow(J3); title('Roberts');

```



## 数字图像的采集

本节介绍如何利用MATLAB软件对USB摄像头进行设置，并采集数字图像和视频。这些内容属于MATLAB的图像采集工具箱（Image Acquisition Toolbox）。

### USB摄像头设置

在MATLAB中通过函数imaqhwinfo()检测用户的MATLAB软件是否安装了图像采集工具箱，并显示图像采集工具箱的版本。

USB摄像头的适配器为winvideo。

```

>> imaqlinfo=imaqhwinfo
imaqlinfo =
    InstalledAdapters: {'coreco' 'winvideo'}
    MATLABVersion: '7.10 (R2010a)'
    ToolboxName: 'Image Acquisition Toolbox'
    ToolboxVersion: '3.5 (R2010a)'


```

```

>> winvideoinfo=imaqhwinfo('winvideo')
I
winvideoinfo =
    AdaptorDllName: [1x60 char]
    AdaptorDllVersion: '3.5 (R2010a)'
    AdaptorName: 'winvideo'
    DeviceIDs: {[1]}
    DeviceInfo: [1x1 struct]

>> winvideoinfo.DeviceInfo
ans =
    DefaultFormat: 'YUY2_160x120'
    DeviceFileSupported: 0 I
    DeviceName: 'USB 视频设备'
    DeviceID: 1
    ObjectConstructor: 'videoinput(''winvideo'', 1)'
    SupportedFormats: {1x5 cell}

>> winvideoinfo.DeviceInfo.SupportedFormats
ans =
    Columns 1 through 3
    'YUY2_160x120'    'YUY2_176x144'    'YUY2_320x240'
    Columns 4 through 5
    'YUY2_352x288'    'YUY2_640x480'

clear all;
close all;
obj=videoinput('winvideo',1,'YUY2_640x480');
vformat=get(obj,'VideoFormat')
vresolution=get(obj,'VideoResolution')

```

## 图像和视频的采集

在MATLAB中，通过函数`videoinput( )`建立摄像头对象后，可以采用函数`preview( )`进行视频采集和显示。该函数的调用格式为：`preview(obj)`，其中`obj`为摄像头对象。利用函数`stopPreview( )`暂停视频的采集和显示，利用函数`closePreview( )`关闭摄像头对象。

在MATLAB中，利用函数`obj=videoinput(...)`建立摄像头对象`obj`后，可以采用函数`start(obj)`开始视频的采集，利用函数`stop(obj)`暂停视频的采集，利用函数`delete( obj)`删除摄像头对象。

```

clear all;
close all;
obj=videoinput('winvideo', 1, 'YUY2_640x480');
preview(obj);
pause(10);
closepreview(obj);
delete(obj);
clear| I

clear all;
close all;
obj=videoinput('winvideo', 1, 'YUY2_640x480');
set(obj, 'FramesPerTrigger', 120);
start(obj);
pause(20);
data=getdata(obj, 120);
size(data)
delete(obj);
clear obj| I

camera=imaqhwinfo;
usbCamera=camera.InstalledAdaptors{2};
obj=videoinput(usbCamera, 1, 'YUY2_640x480');
preview(obj);
start(obj);
for i=1:10
    frame=getsnapshot(obj);
    filename=int2str(i);
    filename=['im',filename,'.jpg'];
    imwrite(frame,filename);
    pause(0.5);
end

stop(obj);
closepreview(obj);
delete(obj);
clear obj;

vid=videoinput('winvideo', 1, 'YUY2_640x480');
figure('Toolbar','none',...
       'Menubar','none',...
       'NumberTitle','Off',...
       'Name','My Preview Window');
vidRes=get(vid, 'VideoResolution');
nBands=get(vid, 'NumberOfBands');
hImage=image(zeros(vidRes(2), vidRes(1), nBands));
preview(vid, hImage);

```

# imaqtool 工具

在MATLAB中，可以采用imaqtool工具直接进行视频的预览。用户可以在MATLAB的命令行窗口直接输入：imaqtool来调用该工具，出现视频设置和采集窗口，如图21.57所示。在Imaqtool工具界面中，用户可以通过单击按钮Start Preview进行视频的预览，单击Start Acquisition进行视频的采集。



## 本章小结

本章介绍了利用**MATLAB**的图像处理工具箱进行数字图像的处理和分析。首先介绍了图像的类型、图像的读取和显示等，然后介绍了图像的常用操作，例如图像的直方图和直方图均衡化、图像的对比度增强等。接着介绍了图像的变换和增强技术，重点介绍了图像的滤波，包括邻域平均法、中值滤波和自适应滤波等。最后介绍了图像的边缘检测，以及利用**USB**摄像头进行图像和视频的采集。本章是数字图像处理的基础，需要读者熟练掌握，多动手自己编写程序。

# 神经网络工具箱

//看看 10 年轻的东西 hhh

人工神经网络，是对人类大脑系统的一阶特性的一种描述。

它是一个数学模型，可以用电子线路来实现，也可以用计算机程序来模拟，是人工智能研究的一种方法。在人工神经网络的发展中，对生物神经系统从不同的角度进行了不同层次的描述和模拟，提出了各种各样的神经网络模型。

本章首先对人工神经网络的基本原理进行介绍，然后重点介绍了常用的感知器神经网络、线性神经网络、BP网络、径向基网络、概率神经网络等。人工神经网络发展迅速，已经广泛应用于许多领域。

## 人工神经网络概述

神经元网络的发展非常迅速，在许多的领域都有很广泛的应用。下面对神经元网络的基本理论进行介绍，然后介绍 MATLAB 的神经网络工具箱。

## 人工神经网络介绍

人工神经网络（artificial neural network, ANN）是在生物神经网络（biological neural network, BNN）的基础上发展起来的，是由大量处理单元广泛互联而成的网络，反映了人脑功能的基本特性。人工神经网络是对人脑的某种抽象、简化与模拟，是模拟人的智能的一种途径，但没有完全的真正反映大脑的功能。

神经元是神经网络的基本处理单元，一个简化的神经元是多输入、单输出的非线性元件。单个神经元的结构和功能都比较简单，但是大量简单神经元相互连接而成的非线性动态系统非常复杂，在自学习、自组织、联想及容错方面具有较强的能力。

## 人工神经元

神经细胞由细胞体、树突和轴突组成。树突由细胞体向各个方向长出，是用来接收信号的。轴突也有许多的分支，通过分支的末梢和其它神经细胞的树突相接触，形成突触。神经细胞通过轴突和突触把产生的信号送到其它神经细胞。

大脑的神经细胞只有两种状态：兴奋和抑制。神经细胞把所有从树突上突触进来的信号进行相加，如果全部信号的总和超过某个阈值，就会激发神经细胞进入兴奋状态，这时会有电信号通过轴突发送出去给其它神经细胞。如果信号总和没有达到阈值，神经细胞就不会兴奋。

## MATLAB 神经网络工具箱

MATLAB的神经网络工具箱，提供了大量的函数用于建立神经网络、网络的学习和训练，以及显示等。

在命令行窗口中输入： **help nnet**， 查看神经网络工具箱的版本和函数。 

### 感知器神经网络

美国学者Rosenblatt于1958年提出了一种神经网络模型，即感知器（Perception），其重要贡献是提出了一种感知器的训练算法。感知器可以用于模式分类。感知器神经网络在神经网络的研究中有着重要意义和地位，其学习算法的自组织、自适应思想，是神经网络学习算法的基础。

#### 感知器神经网络

在MATLAB中，采用函数**newp( )**建立感知器神经网络，该函数的调用格式为：**net=newp(P, T, TF, LF)**，各个参数如下：

**net**为建立的感知器网络。

**P**为R×2的矩阵，由R组输入向量的最大值和最小值组成。

**TF**为网络的传输函数，可以为**hardlim**或**hardlims**，默认为**hardlim**。

**LF**为网络的学习函数，可以为**learnp**或**learnpn**，默认为**learnp**。

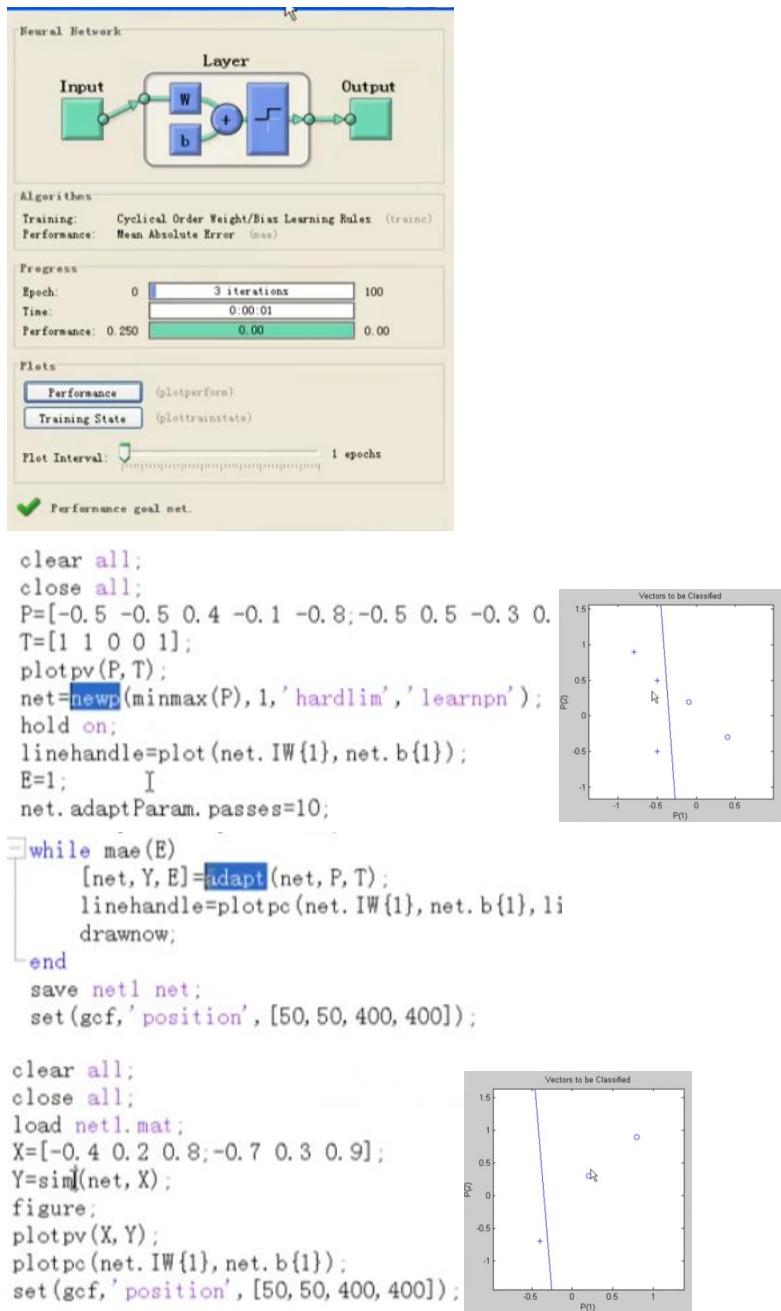
### 设计实例分析

利用函数**newp( )**建立感知器网络，然后采用函数**train( )**对网络进行训练，采用函数**sim( )**对网络进行仿真。

在MATLAB中，采用函数**plotpv( )**绘制感知器的输入向量和目标向量。

在MATLAB中，采用函数**plotpc( )**在感知器向量图中绘制分界线。

```
clear all;
close all;
P=[0 0 1 1;0 1 0 1];
T=[0 1 1 1];
net=newp(minmax(P),1,'hardlim','learnp');
net=train(net,P,T);
Y=sim(net,P);
plotpv(P,T);
plotpc(net.iw{1},net.b{1});
```



## 线性神经网络

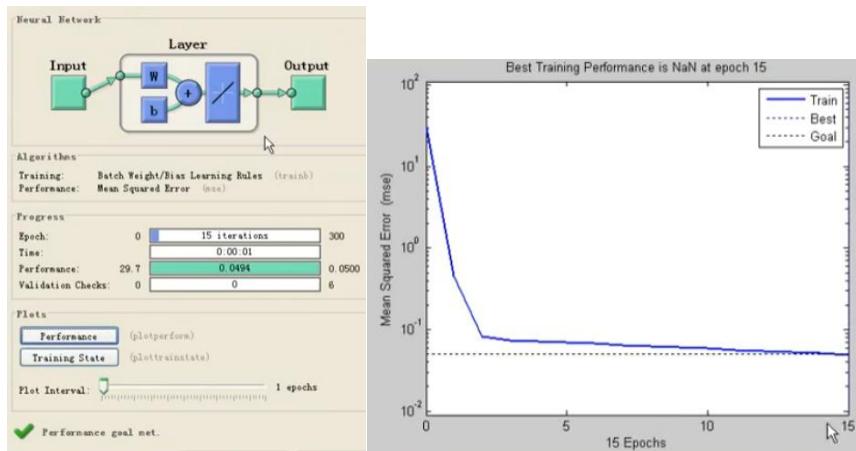
线性神经网络的学习算法比感知器神经网络的学习算法的收敛速度和精度都有很大的提高，主要用于函数逼近、信号预测、系统辨识和模式识别等。下面对线性神经网络进行介绍。

## 线性神经网络

在MATLAB中，采用函数newlin( )建立线性神经网络的一个线性层，利用该函数建立的线性神经网络还需要进行训练。

利用函数maxlinlr( )获取最大学习速率，然后采用函数newlin( )建立线性神经网络，并采用函数train( )对网络进行训练。最后采用函数sim( )对网络进行仿真。

```
clear all;
close all;
P=[1.0 2.1 3 4];
T=[2.0 4.01 5.9 8.0];
lr=maxlinlr(P);
net=newlin(minmax(P), 1, 0, lr);
net.trainParam.epochs=300;
net.trainParam.goal=0.05;
net=train(net, P, T);
Y=sim(net, P)
```

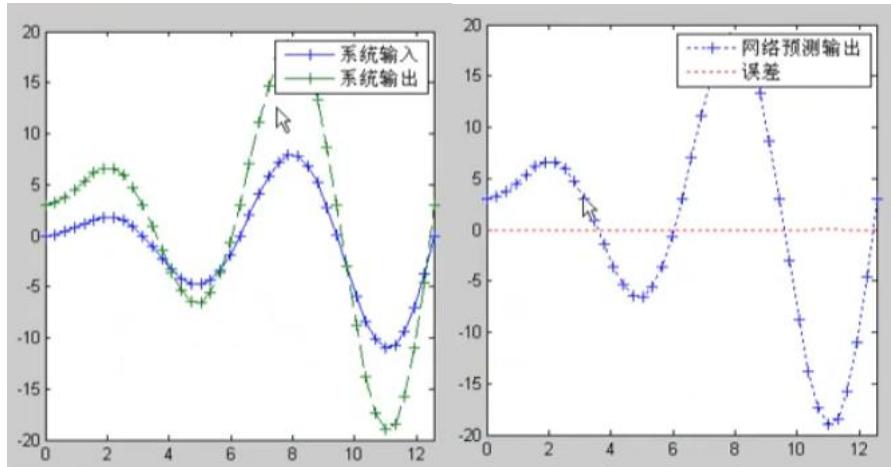


```
clear all;
close all;
P=[1.1 2.2 3];
T=[2.1 4.3 5.9];
net=newlin(P, T);
Y=sim(net, P)
I | 2.1000 4.3000 5.9000
```

## 设计实例分析

利用线性神经网络能够预测时变的信号序列，下面通过一个例子进行说明，通过函数newlind( )建立线性神经网络。

```
close all;
t=0:pi/10:4*pi;
X=t.*sin(t);
T=2*X+3;
figure;
plot(t, X, '+-', t, T, '+-');
legend('系统输入', '系统输出');
set(gca, 'xlim', [0 4*pi]);
set(gcf, 'position', [50, 50, 400, 400]);
net=newlind(X, T);
y=sim(net, X);
figure;
plot(t, y, '+:', t, y-T, 'r:');
legend('网络预测输出', '误差');
set(gca, 'xlim', [0 4*pi]);
set(gcf, 'position', [50, 50, 400, 400]);
```

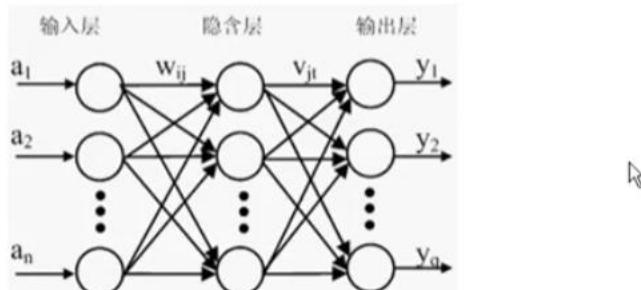


## BP 网络

BP网络是一种带有隐含层的多层前馈网络，在人工神经元网络的实际应用中，BP网络广泛用于函数逼近、模式识别、数据压缩等，BP网络或它的变化形式，体现了人工神经元网络最精华的部分。

### BP 网络介绍

多层前馈网络的反向传播学习算法，简称BP算法，由该算法训练的神经网络，称之为BP神经网络。



### BP 网络的创建

在MATLAB中，通过函数newcf( )创建级联前向网络，通过函数newff( )创建前向BP网络，通过函数newffd( )创建存在输入延迟的前向网络。本书只介绍利用函数newff( )创建BP网络。

### BP 网络设计实例

BP网络的设计主要包括以下几个方面：

网络层数。

输入层的节点数。

输出层的节点数。

隐含层的节点数。

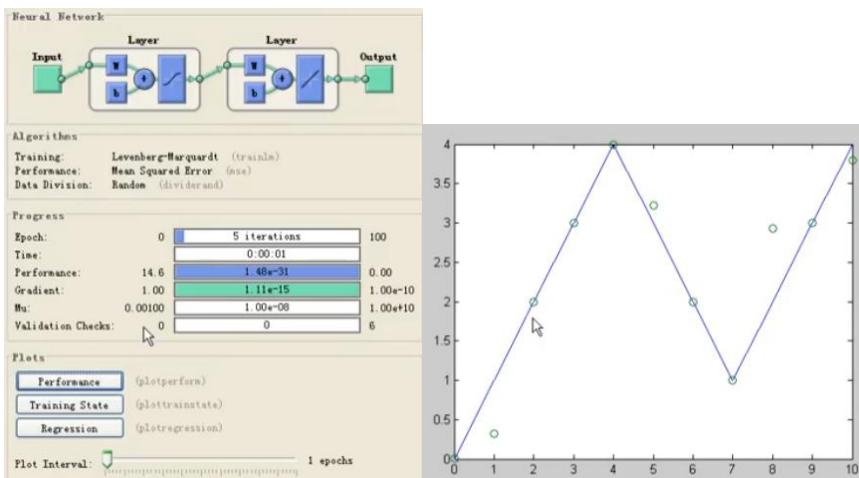
传输函数。

BP网络的训练方法。

```

clear all;
clear all;
P=[0 1 2 3 4 5 6 7 8 9 10];
T=[0 1 2 3 4 3 2 1 2 3 4];
net=newff(P,T,10);
net.trainParam.epochs=100;
net=train(net,P,T);
Y=sim(net,P);
figure;
plot(P,T,P,Y,'o');

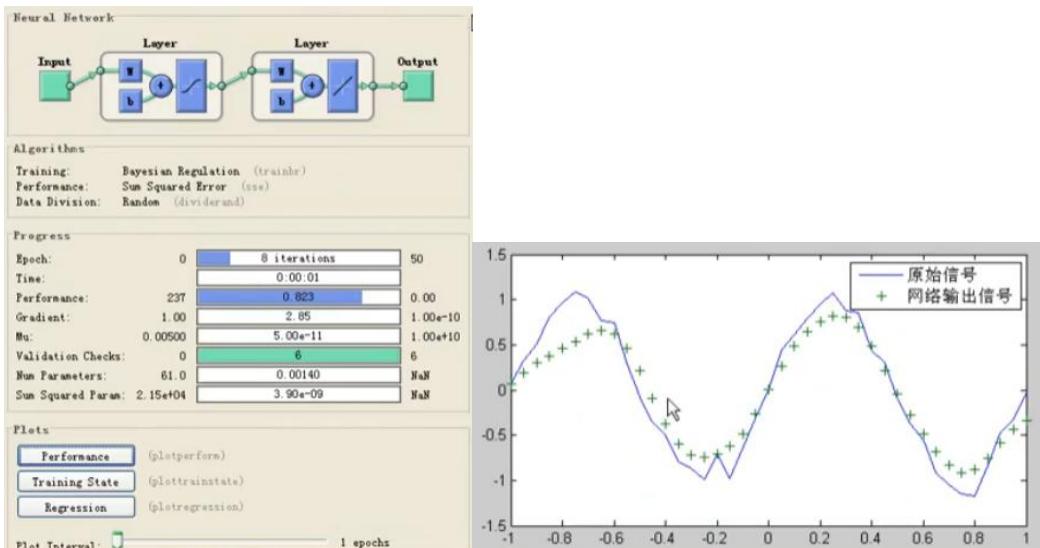
```



```

clear all;
clear all;
P=-1:0.05:1;
T=sin(2*pi*P)+0.1*randn(size(P));
net=newff(P,T,20,{},'trainbr');
net.trainParam.show=10;
net.trainParam.epochs=50;
net=train(net,P,T);
Y=sim(net,P);
figure;
plot(P,T,'-',P,Y,'+');
legend('原始信号','网络输出信号');
set(gcf,'position',[50,50,600,300]);

```



## 径向基神经网络

径向基函数神经网络（RBFNN），简称径向基神经网络，是由Moody和Darken于20世纪80年代提出的一种神经网络结构。径向基神经网络是一种三层前馈网络，能够以任意精度逼近任意连续函数。径向基神经元和线性神经元可以建立广义回归神经网络（Generalized Regression Neural Network, GRNN）。径向基神经元和竞争神经元建立概率神经网络（Probabilistic Neural Network, PNN）。下面分别进行介绍。

### 径向基神经网络

径向基神经网络是一种三层前馈网络，包括：输入层、隐含层和输出层。第一层为输入层，由信号源节点构成，将网络与外界环境连接起来，节点数由输入数据的维数决定。第二层为隐含层，其节点由径向基函数构成，其作用是从输入空间到隐含层空间进行变换。第三层为输出层，对输入模式做出响应，输出节点由隐含层节点给出的基函数的线性组合计算。从输入层到隐含层的变换是非线性的，而从隐含层到输出层的变换是线性的。

在MATLAB中，通过函数newrb( )建立径向基神经网络，通过函数newrbe( )建立精确的径向基神经网络。

```
clear all;
close all;
P=[1 2 3 4 5];
T=[2.1 3.4 5.4 6.9 5.6];
net=newrb(P,T);
x=2:0.5:5
y=sim(net,x)

clear all;
close all;
P=1:10;
T=[2.523 2.434 3.356 4.115 5.834 6.967 7.098
net=newrbe(P,T,2);
y=sim(net,P)
figure;
plot(P,y-T,':+');
title('误差曲线');
```

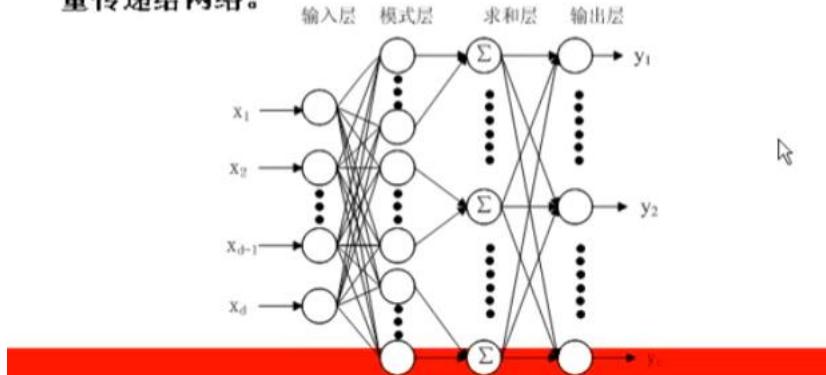
### 广义回归神经网络

广义回归神经网络（GRNN）是径向基神经网络的一种变化形式，由径向基神经元和线性神经元组成。GRNN网络的训练速度快，非线性映射能力强，因此常用于函数逼近。

在MATLAB中，采用函数newgrnn( )建立广义回归神经网络。该函数的调用格式为：`net=newgrnn(P, T, spread)`，其中P为输入向量，T为输出向量，spread为径向基函数的分布密度。

## 概率神经网络

概率神经网络由径向基神经元和竞争神经元组成，经常用来解决分类问题。概率神经网络共分四层，分别为输入层、模式层、求和层和输出层。输入层的每个神经元均为单输入单输出，其传递函数也为线性的，只是将特征向量传递给网络。



```
clear all;
close all;
P=[1:8];
Tc=[2 3 1 2 3 2 1 1];
T=ind2vec(Tc);
net=newpnn(P,T);
Y=sim(net,P);
Yc=vec2ind(Y)
```

## 神经网络 GUI 工具

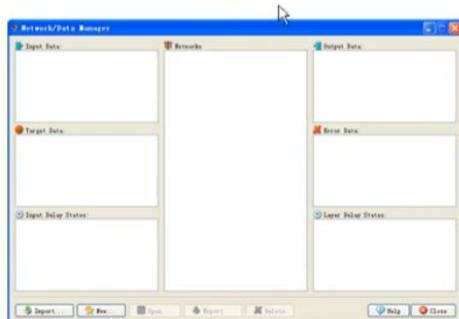
在MATLAB的神经网络工具箱提供了很多的GUI工具：

**nntool**工具、**nctool**工具、**nftool**工具、**nprtool**工具和**nntraintool**工具。**nntool**工具用于神经网络的创建、数据管理等。**nctool**工具是利用神经网络进行分类。**nftool**工具利用神经网络进行数据的聚类，采用SOM神经网络。**nprtool**工具利用神经网络进行模式识别。**nntraintool**工具对神经网络进行训练。此外，还可以采用函数**view( )**在图形界面中查看建立的神经网络。

### nntool 工具

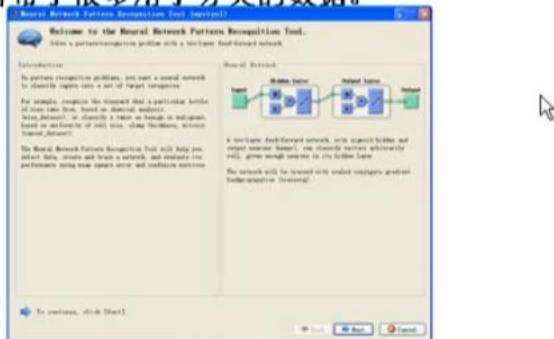
在MATLAB的命令行窗口输入：**nntool**，出现Network/Data Manager窗口如图22.23所示。在MATLAB的命令行窗口输入如下程序，建立输入向量和目标向量：

```
P=0.2:0.2:2*pi;
T=cos(P);
```



## nprtool 工具

在MATLAB中，采用nprtool工具进行模式识别，采用的是BP网络。在MATLAB的命令行窗口输入：nprtool后，出现如图22.29的界面，启动利用nprtool进行模式识别的向导。单击按钮Next后进入数据选择界面，可以选择要分类的数据，而且该工具自带了很多用于分类的数据。



## 本章小结

本章对MATLAB中的神经网络工具箱进行了介绍。首先介绍了神经网络的发展和历史，然后详细的介绍了感知器神经网络、线性神经网络、BP神经网络、径向基神经网络和概率神经网络等。

最后介绍了神经网络的GUI工具。利用神经网络工具箱的GUI工具，不需要编程就能通过神经网络进行分类和数据拟合等，非常简单和方便。

# MATLAB 与 Word 和 Excel 的接口

MATLAB软件具有强大的数据计算和可视化能力，而且提供了大量的工具箱，方便各个领域的用户。MATLAB软件提供了强大的接口，可以和Word和Excel等软件进行数据交互。通过MATLAB软件自带的Notebook工具可以和Word软件结合在一起，在Word文档中执行MATLAB函数和程序，然后将结果返回到Word文档中。Excel link是MATLAB软件和Excel软件进行链接的插件，可以方便的进行MATLAB的工作空间和Excel文件之间进行数据交换，还可以在Excel文件中执行MATLAB函数和程序。本章将详细的介绍Notebook工具和Excel link工具。

## MATLAB 与 Word 混合使用

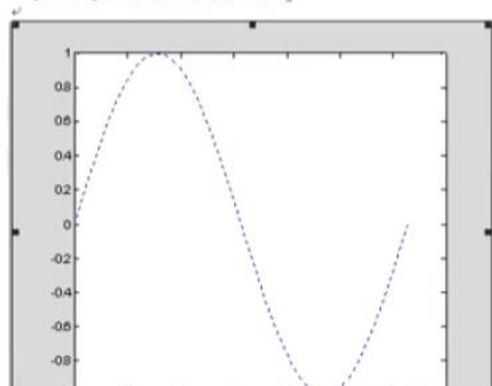
MATLAB软件通过Notebook可以和Word软件进行数据的交换。微软公司的Word软件在文字处理领域应用非常的广泛。MATLAB中的Notebook允许用户在Word中编写程序，然后送到MATLAB软件中执行，最后将结果返回到Word软件中。

### Notebook 安装

MATLAB的Notebook制作的M-book文档不仅拥有Word的全部文字处理功能，而且具备MATLAB无与伦比的矩阵计算和数据可视化功能。

#### notebook -setup

```
[t=0:pi/20:2*pi;     
figure; plot(t,sin(t),':');] %
```



//做笔记？？？

### Notebook 的使用环境

Notebook的功能都是通过Word软件中创建的Notebook菜单来实现的。下面对Notebook菜单的各个菜单项进行详细的介绍。

- 1. Define Input Cell
- 2. Define AutoInit Cell
- 3. Define Calc Zone
- 4. Undefine Cells
- 5. Purge Selected Output Cells
- 6. Group Cells
- 7. Ungroup Cells
- 8. Hide Cell Markers
- 9. Toggle Graph Output for Cell
  
- 10. Evaluate Cell
- 11. Evaluate Calc Zone
- 12. Evaluate M-book
- 13. Evaluate Loop
- 14. Bring MATLAB to Front
- 15. Notebook Options

## Notebook 实例分析

在M-book文档中，可以使用MATLAB工作空间中的变量。

M-book文档中的函数或程序传送到MATLAB中进行执行，然后把结果返回给M-book文档。在使用循环或分支等配对使用的语句时，必须将输入单元合并变为单元组，否则程序会出错。例如，在M-book文档中，输入如下程序：

```
for i=1:10  
    disp(i)  
end
```

## MATLAB 与 Excel 混合使用

Excel Link是Excel软件和MATLAB进行链接的插件。通过Excel Link工具，用户可以在Excel软件中调用MATLAB中的函数进行数据处理和图像处理等，也可以将MATLAB软件中计算的结果或图像传送到Excel文件中。Excel link工具实现了Excel软件和MATLAB软件的数据交换和同步更新。

### Excel link 的安装

首先需要正确的安装Excel软件，然后安装MATLAB 2010a和Excel Link工具。在安装MATLAB的工具箱时，需要选中Excel Link。安装结束后，还需要在Excel Link进行设置，完成Excel软件和MATLAB的Excel Link进行连接。

## 设置启动方式

用户可以在Excel软件中手动启动MATLAB软件。单击Excel软件的工具|宏|宏...，或者按快捷键Alt+F8，会出现宏窗口，如图23.21所示。在宏名中输入：MATLABinit，然后单击按钮执行，即可启动MATLAB软件。由于MATLAB 2010a软件比较大，启动的速度可能会比较慢，需要用户耐心等待。



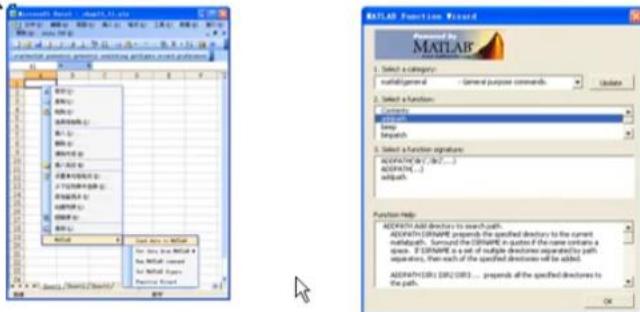
## 启动和终止 Excel link

如果在启动Excel软件时，没有启动MATLAB软件和Excel link。用户可以在Excel软件的数据表中输入：  
=MLopen( )，来启动MATLAB软件。或者，用户也可以在宏窗口中执行宏命令：MLopen，来启动MATLAB软件和Excel link。

如果用户需要在Excel文件中终止MATLAB软件和Excel link的运行，可以在Excel文件的数据表中输入：  
=MLclose( )来关闭MATLAB软件，终止Excel link的执行。此外，用户还可以在宏窗口中，执行宏命令：  
MLclose，来关闭MATLAB软件，从而终止Excel link的执行。

## Excel link 工具栏

安装Excel link后，在Excel软件中显示Excel link工具栏，如图23.22所示。此外，用户还可以用鼠标选中单元格然后右键选择菜单MATLAB下的子菜单来执行Excel link工具，和工具栏的功能完全相同。用户单击工具栏中的按钮wizard会弹出MATLAB的函数向导对话框，如图23.23所示。



## 实例分析

利用函数MLPutMatrix( )将Excel文件中的数据导入到MATLAB的工作空间中，该函数的调用格式为  
**MLPutMatrix(var\_name, mdat)**，或者宏命令格式：  
**MLPutMatrix var\_name, mdat**，其中**var\_name**为变量的名称，**mdat**为Excel工作表的地址。

利用函数MLGetMatrix( )将MATLAB工作空间中的数据传送到Excel文件中。利用函数MLGetFigure( )将MATLAB软件中的当前图形窗口导入到Excel软件中。利用函数MLDeleteMatrix( )来删除MATLAB工作空间中的变量。此外，还可以利用函数MLStartDir( )设置MATLAB软件启动后的当前目录。

## 本章小结

本章详细的介绍了MATLAB软件与Word软件和Excel软件的接口，将MATLAB的强大计算功能和数据可视化功能和Office软件相结合。通过Notebook工具可以非常方便的进行MATLAB软件和Word文档的数据交换。**Excel link**插件将MATLAB软件和Excel软件相结合，方便的进行数据的交换。

完结撒花!!!  
(持续更新 ing)  
特别鸣谢  
Bilibili、GitHub

By Prodefi