



讲解 Matlab 在数学各分支中的应用
讲解 Matlab 函数和命令
荟萃 Matlab 应用实例及技巧
提供实例丰富的多媒体光盘

Design / 何东琳设计工作室



ISBN 7-900335-92-7
定价: 49.00 元



MATLAB 6.0 数学手册

蒲俊 吉家锋 伊良忠 编著

浦东电子出版社
Pudong ePress

MATLAB 6.0 数学手册

蒲俊 吉家锋 伊良忠 著

讲解 Matlab 在数学各分支中的应用
讲解 Matlab 函数和命令
荟萃 Matlab 应用实例及技巧
提供实例丰富的多媒体光盘

浦东电子出版社
Pudong ePress

MATLAB6.0 数学手册

蒲 俊 吉家锋 伊良忠 编著



 **浦东电子出版社**
PeP Pudong ePress

内 容 提 要

MATLAB 已成为多学科、多种工作平台的功能强大、界面友好、语言自然并且开放性强的应用软件，目前的最高版本是 6.0 版。本教程以 6.0 版为基础，从高等工科院校的数学课程出发，提供了使用 MATLAB 的实践性指导。本教程以教学的手段，系统详细地介绍了 MATLAB 在高等数学、数值分析、函数作图、线性代数、概率统计和优化理论中的应用，并配备了大量的例题，让读者能很快掌握 MATLAB 的运算技巧。

本教程按逻辑编排，自始至终用实例描述，既适用于初学者自学，也适用于高级 MATLAB 用户。可作为高等数学、数值分析、工程数学、数学建模、线性规划等课程的教学参考书，也可作为科技工作者学习和使用 MATLAB 的参考书，还可作为数学实验的教学用书，特别适合用作理工科大学学生学习数学课程的教学辅导书。

本教程的光盘内容详尽、实例丰富：包含了 MATLAB 实例的源文件，函数/命令及注解，程序实例。

书 名：MATLAB6.0 数学手册

文本著作者：蒲 俊 吉家锋 伊良忠

C D 制 作 者：海搏多媒体制作中心

责 任 编 辑：舒红梅

出版、发行者：浦东电子出版社

地 址：上海浦东郭守敬路 498 号上海浦东软件园内 201203

电话：021-38954510，38953321，38953323（发行部）

经 销：各地新华书店、软件连锁店

排 版：四川中外科技文化交流中心排版制作中心

C D 生 产 者：东方光盘制造有限公司

文本印刷者：成都地图出版社印刷厂

开 本 / 规 格：787×1092 毫米 16 开本 19.5 印张 240 千字

版 次 / 印 次：2002 年 1 月第一版 2002 年 1 月第一次印刷

印 数：0001——8000 册

本 版 号：ISBN 7—900346—16—3

定 价：33.00 元（1CD 配使用手册）

说明：凡我社光盘配套图书有缺页、倒页、脱页、自然破损，本社发行部负责调换。

前 言

MATLAB 是美国 MathWorks 公司自 20 世纪 80 年代中期推出的数学软件，优秀的数值计算能力和卓越的数据可视化能力使其很快在数学软件中脱颖而出。到目前为止，其最高版本 6.0 版已经推出。随着版本的不断升级，它在数值计算及符号计算功能上得到了进一步完善。**MATLAB** 已经发展成为多学科、多种工作平台的功能强大的大型软件。在欧美等高校，**MATLAB** 已经成为线性代数、自动控制理论、概率论及数理统计、数字信号处理、时间序列分析、动态系统仿真等高级课程的基本教学工具，是攻读学位的大学生、硕士生、博士生必须掌握的基本技能。

MATLAB 的主要特点是：

- 有高性能数值计算的高级算法，特别适合矩阵代数领域；
- 有大量事先定义的数学函数，并且有很强的用户自定义函数的能力；
- 有强大的绘图功能以及具有教育、科学和艺术学的图解和可视化的二维、三维图；
- 基于 HTML 的完整的帮助功能；
- 适合个人应用的强有力的面向矩阵(向量)的高级程序设计语

言；

- 与其它语言编写的程序结合和输入输出格式化数据的能力；
- 有在多个应用领域解决难题的工具箱。

本教程提供了使用 **MATLAB** 的实践性指导，它基于 **MATLAB6.0** 版，内容由浅入深，特别是本书对每一条命令的使用格式都作了详细而又简单明了的说明，并配备了例题加以说明其用法，因此，对于初学者自学是很有帮助的；同时，又对数学中的一些深入问题如数值分析、稀疏矩阵、优化理论以及模糊数学等问题进行了较为详细的论述，因此，该书也可作为科技工作者的科学计算工具书。

本教程的具体特点是：

- 它是以简明方法写就的一本易于掌握的数学手册；
- 编写逻辑性强，内容由浅入深，对于初学者能很快掌握 **MATLAB** 的用法；
- 易于查找命令和问题，给读者灵感与启迪，以解决实际问题；
- 对每一条命令，都进行了详细论述；
- 对于每一条命令，几乎都有易懂的实例；
- 内容按数学分类进行描述。

本教程的光盘内容详尽、实例丰富：包含了 **MATLAB** 实例的源文件，函数/命令及注解，程序实例。

目 录

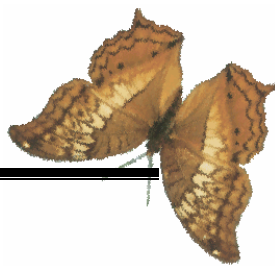
第 1 章 矩阵及其基本运算	1
1.1 矩阵的表示	1
1.1.1 数值矩阵的生成	1
1.1.2 符号矩阵的生成	2
1.1.3 大矩阵的生成	3
1.1.4 多维数组的创建	3
1.1.5 特殊矩阵的生成	4
1.2 矩阵运算	9
1.2.1 加、减运算	9
1.2.2 乘法	9
1.2.3 集合运算	12
1.2.4 除法运算	15
1.2.5 矩阵乘方	16
1.2.6 矩阵函数	16
1.2.7 矩阵转置	17
1.2.8 方阵的行列式	17
1.2.9 逆与伪逆	18
1.2.10 矩阵的迹	19
1.2.11 矩阵和向量的范数	19
1.2.12 条件数	20
1.2.13 矩阵的秩	20
1.2.14 特殊运算	21
1.2.15 符号矩阵运算	26
1.2.16 矩阵元素个数的确定	29
1.3 矩阵分解	29
1.3.1 Cholesky 分解	29
1.3.2 LU 分解	30
1.3.3 QR 分解	30
1.3.4 Schur 分解	32
1.3.5 实 Schur 分解转化成复 Schur 分解	32
1.3.6 特征值分解	33
1.3.7 奇异值分解	33
1.3.8 广义奇异值分解	34
1.3.9 特征值问题的 QZ 分解	35
1.3.10 海森伯格形式的分解	35
1.4 线性方程的组的求解	35
1.4.1 求线性方程组的唯一解或特解（第一类问题）	35
1.4.2 求线性齐次方程组的通解	38
1.4.3 求非齐次线性方程组的通解	39
1.4.4 线性方程组的 LQ 解法	41
1.4.5 双共轭梯度法解方程组	41
1.4.6 稳定双共轭梯度方法解方程组	42
1.4.7 复共轭梯度平方法解方程组	43
1.4.8 共轭梯度的 LSQR 方法	44
1.4.9 广义最小残差法	44
1.4.10 最小残差法解方程组	45
1.4.11 预处理共轭梯度方法	46
1.4.12 准最小残差法解方程组	46

1.5	特征值与二次型.....	47
1.5.1	特征值与特征向量的求法.....	47
1.5.2	提高特征值的计算精度.....	48
1.5.3	复对角矩阵转化为实对角矩阵.....	48
1.5.4	正交基.....	49
1.5.5	二次型.....	49
1.6	秩与线性相关性.....	50
1.6.1	矩阵和向量组的秩以及向量组的线性相关性.....	50
1.6.2	求行阶梯矩阵及向量组的基.....	50
1.7	稀疏矩阵技术.....	51
1.7.1	稀疏矩阵的创建.....	51
1.7.2	将稀疏矩阵转化为满矩阵.....	52
1.7.3	稀疏矩阵非零元素的索引.....	53
1.7.4	外部数据转化为稀疏矩阵.....	53
1.7.5	基本稀疏矩阵.....	53
1.7.6	稀疏矩阵的运算.....	55
1.7.7	画稀疏矩阵非零元素的分布图形.....	56
1.7.8	矩阵变换.....	56
1.7.9	稀疏矩阵的近似欧几里得范数和条件数.....	59
1.7.10	稀疏矩阵的分解.....	59
1.7.11	稀疏矩阵的特征值分解.....	61
1.7.12	稀疏矩阵的线性方程组.....	61
第 2 章	数值计算与数据分析.....	62
2.1	基本数学函数.....	62
2.1.1	三角函数与双曲函数.....	62
2.1.2	其他常用函数.....	69
2.2	插值、拟合与查表.....	76
2.2.1	插值命令.....	77
2.2.2	查表命令.....	83
2.3	数值积分.....	84
2.3.1	一元函数的数值积分.....	84
2.3.2	二元函数重积分的数值计算.....	86
2.4	常微分方程数值解.....	87
2.5	偏微分方程的数值解.....	90
2.5.1	单的 Poisson 方程.....	91
2.5.2	双曲型偏微分方程.....	92
2.5.3	抛物型偏微分方程.....	93
第 3 章	符号运算.....	95
3.1	算术符号操作.....	95
3.2	基本运算.....	97
3.2.1	函数计算器.....	108
3.2.2	微积分.....	109
3.2.3	符号函数的作图.....	112
3.2.4	积分变换.....	118
3.2.5	Taylor 级数.....	123
3.2.6	其它.....	124
第 4 章	概率统计.....	134
4.1	随机数的产生.....	134
4.1.1	二项分布的随机数据的产生.....	134
4.1.2	正态分布的随机数据的产生.....	134

4.1.3	常见分布的随机数产生	135
4.1.4	通用函数求各分布的随机数据	135
4.2	随机变量的概率密度计算	136
4.2.1	通用函数计算概率密度函数值	136
4.2.2	专用函数计算概率密度函数值	137
4.2.3	常见分布的密度函数作图	138
4.3	随机变量的累积概率值(分布函数值)	141
4.3.1	通用函数计算累积概率值	141
4.3.2	专用函数计算累积概率值(随机变量 $X \leq K$ 的概率之和)	141
4.4	随机变量的逆累积分布函数	143
4.4.1	通用函数计算逆累积分布函数值	143
4.4.2	专用函数 <code>inv</code> 计算逆累积分布函数	143
4.5	随机变量的数字特征	145
4.5.1	平均值、中值	145
4.5.2	数据比较	147
4.5.3	期望	148
4.5.4	方差	149
4.5.5	常见分布的期望和方差	151
4.5.6	协方差与相关系数	152
4.6	统计作图	153
4.6.1	正整数的频率表	153
4.6.2	经验累积分布函数图形	154
4.6.3	最小二乘拟合直线	154
4.6.4	绘制正态分布概率图形	154
4.6.5	绘制威布尔(Weibull)概率图形	155
4.6.6	样本数据的盒图	155
4.6.7	给当前图形加一条参考线	156
4.6.8	在当前图形中加入一条多项式曲线	156
4.6.9	样本的概率图形	157
4.6.10	附加有正态密度曲线的直方图	157
4.6.11	在指定的界线之间画正态密度曲线	158
4.7	参数估计	158
4.7.1	常见分布的参数估计	158
4.7.2	非线性模型置信区间预测	160
4.7.3	对数似然函数	164
4.8	假设检验	165
4.8.1	σ^2 已知, 单个正态总体的均值 μ 的假设检验 (U 检验法)	165
4.8.2	σ^2 未知, 单个正态总体的均值 μ 的假设检验 (t 检验法)	166
4.8.3	两个正态总体均值差的检验 (t 检验)	167
4.8.4	两个总体一致性的检验——秩和检验	168
4.8.5	两个总体中位数相等的假设检验——符号秩检验	168
4.8.6	两个总体中位数相等的假设检验——符号检验	169
4.8.7	正态分布的拟合优度测试	169
4.8.8	正态分布的拟合优度测试	170
4.8.9	单个样本分布的 Kolmogorov-Smirnov 测试	170
4.8.10	两个样本具有相同的连续分布的假设检验	171
4.9	方差分析	172
4.9.1	单因素方差分析	172
4.9.2	双因素方差分析	174
第 5 章	优化问题	176

5.1	线性规划问题.....	176
5.2	foptions 函数.....	177
5.3	非线性规划问题.....	178
5.3.1	有约束的一元函数的最小值.....	178
5.3.2	无约束多元函数最小值.....	179
5.3.3	有约束的多元函数最小值.....	181
5.3.4	二次规划问题.....	183
5.4	“半无限”有约束的多元函数最优解.....	185
5.5	极小化极大 (Minmax) 问题.....	189
5.6	多目标规划问题.....	191
5.7	最小二乘最优问题.....	194
5.7.1	约束线性最小二乘.....	194
5.7.2	非线性数据 (曲线) 拟合.....	195
5.7.3	非线性最小二乘.....	196
5.7.4	非负线性最小二乘.....	198
5.8	非线性方程 (组) 求解.....	198
5.8.1	非线性方程的解.....	198
5.8.2	非线性方程组的解.....	199
第 6 章	模糊逻辑.....	201
6.1	隶属函数.....	201
6.1.1	高斯隶属函数.....	201
6.1.2	两边型高斯隶属函数.....	201
6.1.3	建立一般钟型隶属函数.....	202
6.1.4	两个 sigmoid 型隶属函数之差组成的隶属函数.....	202
6.1.5	通用隶属函数计算.....	203
6.1.6	建立 Π 型隶属函数.....	203
6.1.7	通过两个 sigmoid 型隶属函数的乘积构造隶属函数.....	204
6.1.8	建立 Sigmoid 型隶属函数.....	204
6.1.9	建立 S 型隶属函数.....	205
6.1.10	建立梯形隶属函数.....	206
6.1.11	建立三角形隶属函数.....	207
6.1.12	建立 Z 型隶属函数.....	208
6.1.13	两个隶属函数之间转换参数.....	209
6.1.14	基本 FIS 编辑器.....	209
6.1.15	隶属函数编辑器.....	211
6.2	模糊推理结构 FIS.....	212
6.2.1	不使用数据聚类方法从数据生成 FIS 结构.....	212
6.2.2	使用减法聚类方法从数据生成 FIS 结构.....	213
6.2.3	生成一个 FIS 输出曲面.....	213
6.2.4	将 mamdan 型 FIS 转换为 Sugeno FIS.....	214
6.2.5	完成模糊推理计算.....	214
6.2.6	模糊 c 均值聚类.....	215
6.2.7	模糊均值和减法聚类.....	215
6.2.8	绘制一个 FIS.....	216
6.2.9	绘制给定变量的所有隶属的曲线.....	216
6.2.10	从磁盘装入一个 FIS.....	217
6.2.11	从 FIS 中删除某一隶属函数.....	218
6.2.12	从 FIS 中删除变量.....	218
6.2.13	设置模糊系统属性.....	219
6.2.14	以分行形式显示 FIS 结构的所有属性.....	220

6.2.15	完成模糊运算.....	221
6.2.16	解析模糊规则.....	222
6.2.17	规则编辑器和语法编辑器.....	223
6.2.18	规则观察器和模糊推理框图.....	224
6.2.19	保存 FIS 到磁盘上.....	224
6.2.20	显示 FIS 的规则.....	225
6.2.21	显示 FIS 结构的所有属性.....	226
第 7 章	绘图与图形处理.....	228
7.1	二维图形.....	228
7.1.1	基本平面图形命令.....	228
7.1.2	特殊平面图形命令.....	235
7.1.3	二维图形注释命令.....	241
7.2	三维图形.....	245
7.2.1	三维曲线、面填色命令.....	245
7.2.2	三维图形等高线.....	247
7.2.3	曲面与网格图命令.....	250
7.2.4	三维数据的其他表现形式命令.....	254
7.3	通用图形函数命令.....	260
7.3.1	图形对象句柄命令.....	260
7.3.2	轴的产生和控制命令.....	271
7.3.3	图形句柄操作命令.....	272
7.3.4	图形窗口的控制命令.....	274
7.4	颜色与光照模式命令.....	276
7.4.1	颜色控制命令.....	276
7.4.2	色图控制命令.....	278
第 8 章	MATLAB 编程.....	280
8.1	MATLAB 的注释和标点.....	280
8.2	MATLAB 的编程语言.....	280
8.2.1	M-文件编写的函数.....	280
8.2.2	交互式输入.....	288
8.2.3	程序控制流.....	289
8.2.3	逻辑函数.....	296
8.3	M-文件的出错信息与调试.....	297
8.3.1	M-文件执行的出错信息.....	297
8.3.2	函数的调试命令.....	298



第1章 矩阵及其基本运算

MATLAB，即“矩阵实验室”，它是以矩阵为基本运算单元。因此，本书从最基本的运算单元出发，介绍 MATLAB 的命令及其用法。

1.1 矩阵的表示

1.1.1 数值矩阵的生成

1. 实数值矩阵输入

MATLAB 的强大功能之一体现在能直接处理向量或矩阵。当然首要任务是输入待处理的向量或矩阵。

不管是任何矩阵（向量），我们可以直接按行方式输入每个元素：同一行中的元素用逗号（,）或者用空格符来分隔，且空格个数不限；不同的行用分号（;）分隔。所有元素处于一方括号（[]）内；当矩阵是多维（三维以上），且方括号内的元素是维数较低的矩阵时，会有多重的方括号。如：

```
>> Time = [11 12 1 2 3 4 5 6 7 8 9 10]
Time =
    11    12     1     2     3     4     5     6     7     8     9    10
>> X_Data = [2.32 3.43; 4.37 5.98]
X_Data =
    2.43    3.43
    4.37    5.98
>> vect_a = [1 2 3 4 5]
vect_a =
     1     2     3     4     5
>> Matrix_B = [1 2 3;
>>             2 3 4; 3 4 5]
Matrix_B =
     1     2     3
     2     3     4
     3     4     5
>> Null_M = [] %生成一个空矩阵
```

2. 复数矩阵输入

复数矩阵有两种生成方式：

第一种方式

例 1-1

```
>> a=2.7;b=13/25;
>> C=[1,2*a+i*b,b*sqrt(a); sin(pi/4),a+5*b,3.5+1]
C=
    1.0000          5.4000 + 0.5200i    0.8544
    0.7071          5.3000          4.5000
```



第2种方式

例 1-2

```
>> R=[1 2 3;4 5 6], M=[11 12 13;14 15 16]
R =
     1     2     3
     4     5     6
M =
    11    12    13
    14    15    16
>> CN=R+i*M
CN =
    1.0000 +11.0000i    2.0000 +12.0000i    3.0000 +13.0000i
    4.0000 +14.0000i    5.0000 +15.0000i    6.0000 +16.0000i
```

1.1.2 符号矩阵的生成

在 MATLAB 中输入符号向量或者矩阵的方法和输入数值类型的向量或者矩阵在形式上很相像，只不过要用到符号矩阵定义函数 `sym`，或者是用到符号定义函数 `syms`，先定义一些必要的符号变量，再像定义普通矩阵一样输入符号矩阵。

1. 用命令 `sym` 定义矩阵：

这时的函数 `sym` 实际是在定义一个符号表达式，这时的符号矩阵中的元素可以是任何的符号或者是表达式，而且长度没有限制，只是将方括号置于用于创建符号表达式的单引号中。如下例：

例 1-3

```
>> sym_matrix = sym ('[a b c; Jack, Help Me!, NO WAY!]', 's')
sym_matrix =
     [a          b          c]
     [Jack   Help Me!   NO WAY!]
>> sym_digits = sym ('[1 2 3; a b c; sin (x) cos (y) tan (z) ]')
sym_digits =
     [1          2          3]
     [a          b          c]
     [sin (x) cos (y) tan (z) ]
```

2. 用命令 `syms` 定义矩阵

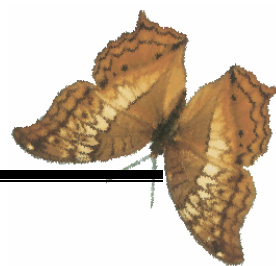
先定义矩阵中的每一个元素为一个符号变量，而后像普通矩阵一样输入符号矩阵。

例 1-4

```
>> syms a b c ;
>> M1 = sym ('Classical');
>> M2 = sym ('Jazz');
>> M3 = sym ('Blues')
>> syms_matrix = [a b c; M1, M2, M3; int2str ([2 3 5])]
syms_matrix =
     [ a          b          c]
     [Classical  Jazz  Blues]
     [ 2          3          5]
```

把数值矩阵转化成相应的符号矩阵。

数值型和符号型在 MATLAB 中是不相同的，它们之间不能直接进行转化。MATLAB 提供了一个将数值型转化成符号型的命令，即 `sym`。

**例 1-5**

```
>> Digit_Matrix = [1/3 sqrt(2) 3.4234; exp(0.23) log(29) 23^(-11.23)]
>> Syms_Matrix = sym(Digit_Matrix)
```

结果是:

```
Digit_Matrix =
    0.3333    1.4142    3.4234
    1.2586    3.3673    0.0000

Syms_Matrix =
[
    1/3,                                sqrt(2),                                17117/5000]
[5668230535726899*2^(-52), 7582476122586655*2^(-51), 5174709270083729*2^(-103)]
```

注意: 矩阵是用分数形式还是浮点形式表示的, 将矩阵转化成符号矩阵后, 都将以最接近原值的有理数形式表示或者是函数形式表示。

1.1.3 大矩阵的生成

对于大型矩阵, 一般创建 M 文件, 以便于修改:

例 1-6 用 M 文件创建大矩阵, 文件名为 example.m

```
exm=[ 456    468    873     2   579    55
      21    687    54    488     8    13
      65   4567    88    98    21     5
      456     68   4589   654     5   987
      5488    10     9     6    33    77]
```

在 MATLAB 窗口输入:

```
>>example;
>>size(exm) %显示 exm 的大小
ans=
     5     6 %表示 exm 有 5 行 6 列。
```

1.1.4 多维数组的创建

函数 cat

格式 A=cat(n,A1,A2,...,Am)

说明 n=1 和 n=2 时分别构造[A1; A2]和[A1, A2], 都是二维数组, 而 n=3 时可以构造出三维数组。

例 1-7

```
>> A1=[1,2,3;4,5,6;7,8,9];A2=A1';A3=A1-A2;
>> A4=cat(3,A1,A2,A3)
A4(:,:,1) =
     1     2     3
     4     5     6
     7     8     9
A4(:,:,2) =
     1     4     7
     2     5     8
     3     6     9
A4(:,:,3) =
     0    -2    -4
     2     0    -2
     4     2     0
```



或用另一种原始方式可以定义：

例 1-8

```
>> A1=[1,2,3;4,5,6;7,8,9];A2=A1';A3=A1-A2;
>> A5(:,1)=A1, A5(:,2)=A2, A5(:,3)=A3
A5(:,1) =
     1     2     3
     4     5     6
     7     8     9
A5(:,2) =
     1     4     7
     2     5     8
     3     6     9
A5(:,3) =
     0    -2    -4
     2     0    -2
     4     2     0
```

1.1.5 特殊矩阵的生成

命令 全零阵

函数 **zeros**

格式 $B = \text{zeros}(n)$ %生成 $n \times n$ 全零阵
 $B = \text{zeros}(m,n)$ %生成 $m \times n$ 全零阵
 $B = \text{zeros}([m \ n])$ %生成 $m \times n$ 全零阵
 $B = \text{zeros}(d1,d2,d3 \dots)$ %生成 $d1 \times d2 \times d3 \times \dots$ 全零阵或数组
 $B = \text{zeros}([d1 \ d2 \ d3 \dots])$ %生成 $d1 \times d2 \times d3 \times \dots$ 全零阵或数组
 $B = \text{zeros}(\text{size}(A))$ %生成与矩阵 A 相同大小的全零阵

命令 单位阵

函数 **eye**

格式 $Y = \text{eye}(n)$ %生成 $n \times n$ 单位阵
 $Y = \text{eye}(m,n)$ %生成 $m \times n$ 单位阵
 $Y = \text{eye}(\text{size}(A))$ %生成与矩阵 A 相同大小的单位阵

命令 全 1 阵

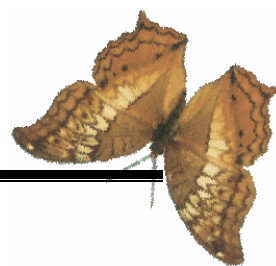
函数 **ones**

格式 $Y = \text{ones}(n)$ %生成 $n \times n$ 全 1 阵
 $Y = \text{ones}(m,n)$ %生成 $m \times n$ 全 1 阵
 $Y = \text{ones}([m \ n])$ %生成 $m \times n$ 全 1 阵
 $Y = \text{ones}(d1,d2,d3 \dots)$ %生成 $d1 \times d2 \times d3 \times \dots$ 全 1 阵或数组
 $Y = \text{ones}([d1 \ d2 \ d3 \dots])$ %生成 $d1 \times d2 \times d3 \times \dots$ 全 1 阵或数组
 $Y = \text{ones}(\text{size}(A))$ %生成与矩阵 A 相同大小的全 1 阵

命令 均匀分布随机矩阵

函数 **rand**

格式 $Y = \text{rand}(n)$ %生成 $n \times n$ 随机矩阵，其元素在 (0, 1) 内
 $Y = \text{rand}(m,n)$ %生成 $m \times n$ 随机矩阵



```

Y = rand([m n])           %生成 m×n 随机矩阵
Y = rand(m,n,p,...)       %生成 m×n×p×...随机矩阵或数组
Y = rand([m n p...])      %生成 m×n×p×...随机矩阵或数组
Y = rand(size(A))         %生成与矩阵 A 相同大小的随机矩阵
rand                       %无变量输入时只产生一个随机数
s = rand('state')         %产生包括均匀发生器当前状态的 35 个元素的向量
rand('state', s)          %使状态重置为 s
rand('state', 0)          %重置发生器到初始状态
rand('state', j)          %对整数 j 重置发生器到第 j 个状态
rand('state', sum(100*clock)) %每次重置到不同状态

```

例 1-9 产生一个 3×4 随机矩阵

```

>> R=rand(3,4)
R =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919

```

例 1-10 产生一个在区间[10, 20]内均匀分布的 4 阶随机矩阵

```

>> a=10;b=20;
>> x=a+(b-a)*rand(4)
x =
    19.2181    19.3547    10.5789    11.3889
    17.3821    19.1690    13.5287    12.0277
    11.7627    14.1027    18.1317    11.9872
    14.0571    18.9365    10.0986    16.0379

```

命令 正态分布随机矩阵

函数 **randn**

```

格式 Y = randn(n)           %生成 n×n 正态分布随机矩阵
      Y = randn(m,n)        %生成 m×n 正态分布随机矩阵
      Y = randn([m n])      %生成 m×n 正态分布随机矩阵
      Y = randn(m,n,p,...)  %生成 m×n×p×...正态分布随机矩阵或数组
      Y = randn([m n p...]) %生成 m×n×p×...正态分布随机矩阵或数组
      Y = randn(size(A))    %生成与矩阵 A 相同大小的正态分布随机矩阵
randn                       %无变量输入时只产生一个正态分布随机数
s = randn('state')         %产生包括正态发生器当前状态的 2 个元素的向量
s = randn('state', s)      %重置状态为 s
s = randn('state', 0)      %重置发生器为初始状态
s = randn('state', j)      %对于整数 j 重置状态到第 j 状态
s = randn('state', sum(100*clock)) %每次重置到不同状态

```

例 1-11 产生均值为 0.6，方差为 0.1 的 4 阶矩阵

```

>> mu=0.6; sigma=0.1;
>> x=mu+sqrt(sigma)*randn(4)
x =
    0.8311    0.7799    0.1335    1.0565
    0.7827    0.5192    0.5260    0.4890

```



```
0.6127    0.4806    0.6375    0.7971
0.8141    0.5064    0.6996    0.8527
```

命令 产生随机排列

函数 **randperm**

格式 `p = randperm(n)` %产生 1~n 之间整数的随机排列

例 1-12

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

命令 产生线性等分向量

函数 **linspace**

格式 `y = linspace(a,b)` %在(a, b)上产生 100 个线性等分点

`y = linspace(a,b,n)` %在(a, b)上产生 n 个线性等分点

命令 产生对数等分向量

函数 **logspace**

格式 `y = logspace(a,b)` %在 $(10^a, 10^b)$ 之间产生 50 个对数等分向量

`y = logspace(a,b,n)`

`y = logspace(a,pi)`

命令 计算矩阵中元素个数

`n = numel(a)` %返回矩阵 A 的元素的个数

命令 产生以输入元素为对角线元素的矩阵

函数 **blkdiag**

格式 `out = blkdiag(a,b,c,d,...)` %产生以 a,b,c,d,...为对角线元素的矩阵

例 1-13

```
>> out = blkdiag(1,2,3,4)
out =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

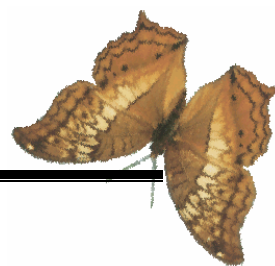
命令 友矩阵

函数 **companion**

格式 `A = companion(u)` %u 为多项式系统向量, A 为友矩阵, A 的第 1 行元素为 $-u(2:n)/u(1)$, 其中 u(2:n)为 u 的第 2 到第 n 个元素, A 为特征值就是多项式的特征根。

例 1-14 求多项式 $(x-1)(x-2)(x+3) = x^3 - 7x + 6$ 的友矩阵和根

```
>> u=[1 0 -7 6];
>> A=companion(u)      %求多项式的友矩阵
A =
     0     7    -6
     1     0     0
     0     1     0
>> eig(A)      %A 的特征值就是多项式的根
ans =
```

-3.0000
2.0000
1.0000

命令 **hadamard** 矩阵

函数 **hadamard**

格式 $H = \text{hadamard}(n)$ %返回 n 阶 hadamard 矩阵

例 1-15

```
>> h=hadamard(4)
h =
     1     1     1     1
     1    -1     1    -1
     1     1    -1    -1
     1    -1    -1     1
```

命令 **Hankel** 方阵

函数 **hankel**

格式 $H = \text{hankel}(c)$ %第 1 列元素为 c , 反三角以下元素为 0。

$H = \text{hankel}(c,r)$ %第 1 列元素为 c , 最后一行元素为 r , 如果 c 的最后一个元素与 r 的第一个元素不同, 交叉位置元素取为 c 的最后一个元素。

例 1-16

```
>> c=1:3,r=7:10
c =
     1     2     3
r =
     7     8     9    10
>> h=hankel(c,r)
h =
     1     2     3     8
     2     3     8     9
     3     8     9    10
```

命令 **Hilbert** 矩阵

函数 **hilb**

格式 $H = \text{hilb}(n)$ %返回 n 阶 Hilbert 矩阵, 其元素为 $H(i,j)=1/(i+j-1)$ 。

例 1-17 产生一个 3 阶 Hilbert 矩阵

```
>> format rat %以有理形式输出
>> H=hilb(3)
H =
     1          1/2          1/3
    1/2          1/3          1/4
    1/3          1/4          1/5
```

命令 逆 Hilbert 矩阵

函数 **invhilb**

格式 $H = \text{invhilb}(n)$ %产生 n 阶逆 Hilbert 矩阵

命令 **Magic(魔方)** 矩阵

函数 **magic**

格式 $M = \text{magic}(n)$ %产生 n 阶魔方矩阵

例 1-18



```
>> M=magic(3)
```

```
M =  
      8      1      6  
      3      5      7  
      4      9      2
```

命令 Pascal 矩阵

函数 **pascal**

格式 **A = pascal(n)** %产生 n 阶 Pascal 矩阵，它是对称、正定矩阵，它的元素由 Pascal 三角组成，它的逆矩阵的所有元素都是整数。

A = pascal(n,1) %返回由下三角的 Cholesky 系数组成的 Pascal 矩阵

A = pascal(n,2) %返回 Pascal(n,1)的转置和交换的形式

例 1-19

```
>> A=pascal(4)
```

```
A =  
      1      1      1      1  
      1      2      3      4  
      1      3      6     10  
      1      4     10     20
```

```
>> A=pascal(3,1)
```

```
A =  
      1      0      0  
      1     -1      0  
      1     -2      1
```

```
>> A=pascal(3,2)
```

```
A =  
      1      1      1  
     -2     -1      0  
      1      0      0
```

命令 托普利兹矩阵

函数 **toeplitz**

格式 **T = toeplitz(c,r)** %生成一个非对称的托普利兹矩阵，将 c 作为第 1 列，将 r 作为第 1 行，其余元素与左上角相邻元素相等。

T = toeplitz(r) %用向量 r 生成一个对称的托普利兹矩阵

例 1-20

```
>> c=[1 2 3 4 5];
```

```
>> r=[1.5 2.5 3.5 4.5 5.5];
```

```
>> T=toeplitz(c,r)
```

```
T =  
      1      5/2      7/2      9/2     11/2  
      2      1      5/2      7/2      9/2  
      3      2      1      5/2      7/2  
      4      3      2      1      5/2  
      5      4      3      2      1
```

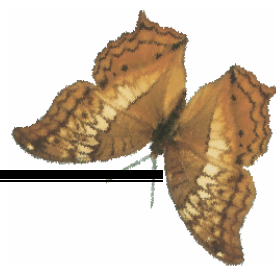
命令 Wilkinson 特征值测试阵

函数 **wilkinson**

格式 **W = wilkinson(n)** %返回 n 阶 Wilkinson 特征值测试阵

例 1-21

```
>> W=wilkinson(4)
```



```

W =
    3/2         1         0         0
         1        1/2         1         0
         0         1        1/2         1
         0         0         1        3/2

>> W=wilkinson(7)
W =
     3     1     0     0     0     0     0
     1     2     1     0     0     0     0
     0     1     1     1     0     0     0
     0     0     1     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     0     1     2     1
     0     0     0     0     0     1     3

```

1.2 矩阵运算

1.2.1 加、减运算

运算符：“+”和“-”分别为加、减运算符。

运算规则：对应元素相加、减，即按线性代数中矩阵的“+”，“-”运算进行。

例 1-22

```

>>A=[1, 1, 1; 1, 2, 3; 1, 3, 6]
>>B=[8, 1, 6; 3, 5, 7; 4, 9, 2]
>>A+B=A+B
>>A-B=A-B
结果显示：A+B=
     9     2     7
     4     7    10
     5    12     8
A-B=
    -7     0    -5
    -2    -3    -4
    -3    -6     4

```

1.2.2 乘法

运算符：*

运算规则：按线性代数中矩阵乘法运算进行，即放在前面的矩阵的各行元素，分别与放在后面的矩阵的各列元素对应相乘并相加。

1. 两个矩阵相乘

例 1-23

```

>>X=[2 3 4 5;
     1 2 2 1];
>>Y=[0 1 1;
     1 1 0;
     0 0 1;
     1 0 0];
Z=X*Y

```



结果显示为:

Z=

```
8 5 6
3 3 3
```

2. 矩阵的数乘: 数乘矩阵

上例中: $a=2*X$

则显示: $a=$

```
4 6 8 10
2 4 4 2
```

向量的点乘 (内积): 维数相同的两个向量的点乘。

数组乘法:

$A.*B$ 表示 A 与 B 对应元素相乘。

3. 向量点积

函数 **dot**

格式 $C = \text{dot}(A,B)$ %若 A 、 B 为向量, 则返回向量 A 与 B 的点积, A 与 B 长度相同; 若为矩阵, 则 A 与 B 有相同的维数。

$C = \text{dot}(A,B,\text{dim})$ %在 dim 维数中给出 A 与 B 的点积

例

```
>>X=[-1 0 2];
>>Y=[-2 -1 1];
>>Z=dot(X,Y)
```

则显示: $Z=$

```
4
```

还可使用另一种算法:

```
sum(X.*Y)
```

ans=

```
4
```

4. 向量叉乘

在数学上, 两向量的叉乘是一个过两相向量的交点且垂直于两向量所在平面的向量。

在 Matlab 中, 用函数 **cross** 实现。

函数 **cross**

格式 $C = \text{cross}(A,B)$ %若 A 、 B 为向量, 则返回 A 与 B 的叉乘, 即 $C=A \times B$, A 、 B 必须是 3 个元素的向量; 若 A 、 B 为矩阵, 则返回一个 $3 \times n$ 矩阵, 其中的列是 A 与 B 对应列的叉积, A 、 B 都是 $3 \times n$ 矩阵。

$C = \text{cross}(A,B,\text{dim})$ %在 dim 维数中给出向量 A 与 B 的叉积。 A 和 B 必须具有相同的维数, $\text{size}(A,\text{dim})$ 和 $\text{size}(B,\text{dim})$ 必须是 3。

例 1-24 计算垂直于向量(1, 2, 3)和(4, 5, 6)的向量。

```
>>a=[1 2 3];
>>b=[4 5 6];
>>c=cross(a,b)
```

结果显示:

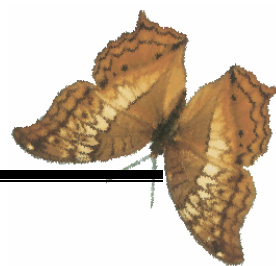
c=

```
-3 6 -3
```

可得垂直于向量(1, 2, 3)和(4, 5, 6)的向量为 $\pm(-3, 6, -3)$

5. 混合积





混合积由以上两函数实现:

例 1-25 计算向量 $a=(1, 2, 3)$ 、 $b=(4, 5, 6)$ 和 $c=(-3, 6, -3)$ 的混合积 $a \cdot (b \times c)$

解:

```
>>a=[1 2 3]; b=[4 5 6]; c=[-3 6 -3];
>>x=dot(a, cross(b, c))
结果显示: x =
        54
```

注意: 先叉乘后点乘, 顺序不可颠倒。

6. 矩阵的卷积和多项式乘法

函数 **conv**

格式 $w = \text{conv}(u, v)$ % u 、 v 为向量, 其长度可不相同。

说明 长度为 m 的向量序列 u 和长度为 n 的向量序列 v 的卷积(Convolution)定义为:

$w(k) = \sum_{j=1}^k u(j) v(k+1-j)$ 式中: w 向量序列的长度为 $(m+n-1)$, 当 $m=n$ 时,

```
w(1) = u(1)*v(1)
w(2) = u(1)*v(2)+u(2)*v(1)
w(3) = u(1)*v(3)+u(2)*v(2)+u(3)*v(1)
...
w(n) = u(1)*v(n)+u(2)*v(n-1)+ ... +u(n)*v(1)
...
w(2*n-1) = u(n)*v(n)
```

例 1-26 展开多项式 $(s^2 + 2s + 2)(s + 4)(s + 1)$

```
解: >> w=conv([1,2,2],conv([1,4],[1,1]))
      w =
          1         7        16        18         8
>> P=poly2str(w,'s') %将 w 表示成多项式
      P =
      s^4 + 7 s^3 + 16 s^2 + 18 s + 8
```

7. 反褶积(解卷)和多项式除法运算

函数 **deconv**

格式 $[q, r] = \text{deconv}(v, u)$ % 多项式 v 除以多项式 u , 返回商多项式 q 和余多项式 r 。

注意: v 、 u 、 q 、 r 都是按降幂排列的多项式系数向量。

例 1-27 $(x^3 + 2x^2 + 3x + 4)(10x^2 + 20x + 30)$, 则其卷积为

```
>>u=[1 2 3 4]
>>v=[10 20 30]
>>c=conv(u,v)
      c =
         10         40        100        160        170        120
则反褶积为
>>[q,r]=deconv(c,u)
      q =
         10        20        30
      r =
         0         0         0         0         0         0
```

8. 张量积

**函数 kron****格式** $C = \text{kron}(A, B)$ %A 为 $m \times n$ 矩阵, B 为 $p \times q$ 矩阵, 则 C 为 $mp \times nq$ 矩阵。

说明 A 与 B 的张量积定义为: $C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$ $A \otimes B$ 与 $B \otimes A$

均为 $mp \times nq$ 矩阵, 但一般地 $A \otimes B \neq B \otimes A$ 。

例 1-28 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ 求 $A \otimes B$ 。

```
>> A=[1 2;3 4];B=[1 2 3;4 5 6;7 8 9];
>> C=kron(A,B)
C =
     1     2     3     2     4     6
     4     5     6     8    10    12
     7     8     9    14    16    18
     3     6     9     4     8    12
    12    15    18    16    20    24
    21    24    27    28    32    36
```

1.2.3 集合运算

1. 两个集合的交集

函数 intersect

格式 $c = \text{intersect}(a, b)$ %返回向量 a、b 的公共部分, 即 $c = a \cap b$ 。
 $c = \text{intersect}(A, B, 'rows')$ %A、B 为相同列数的矩阵, 返回元素相同的行。
 $[c, ia, ib] = \text{intersect}(a, b)$ %c 为 a、b 的公共元素, ia 表示公共元素在 a 中的位置, ib 表示公共元素在 b 中位置。

例 1-29

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
A =
     1     2     3     4
     1     2     4     6
     6     7     1     4
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
B =
     1     2     3     8
     1     1     4     6
     6     7     1     4
>> C=intersect(A,B,'rows')
C =
     6     7     1     4
```

例 1-30

```
>> A = [1 9 6 20]; B = [1 2 3 4 6 10 20];
>> [c,ia,ib] = intersect(A,B)
```

2. 检测集合中的元素

格式	k = ismember(a,S)	%当 a 中元素属于 S 时，k 取 1，否则，k 取 0。
	k = ismember(A,S,'rows')	%A、S 有相同的列，返回行相同 k 取 1，不相同取 0 的列向量。

```
>> S=[0 2 4 6 8 10 12 14 16 18 20];
>> a=[1 2 3 4 5 6];
>> k=ismember(a,S)
k =
    0     1     0     1     0     1    %1表示相同元素的位置
```

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
>> k=ismember(A,B,'rows')
k =
    0
    0
    1 %1 表示元素相同的行
```

格式	<code>c = setdiff(a,b)</code>	%返回属于 a 但不属于 b 的不同元素的集合， $C = a - b$ 。
	<code>c = setdiff(A,B,'rows')</code>	%返回属于 A 但不属于 B 的不同行
	<code>[c,i] = setdiff(...)</code>	%c 与前面一致，i 表示 c 中元素在 A 中的位置。

```
>> A = [1 7 9 6 20]; B = [1 2 3 4 6 10 20];
>> c = setdiff(A,B)
c =
    7     9
```

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
>> c=setdiff(A,B,'rows')
c =
    1     2     3     4
    1     2     4     6
```

格式	c = setxor(a,b)	%返回集合 a、b 交集的非
	c = setxor(A,B,'rows')	%返回矩阵 A、B 交集的非，A、B 有相同列数。
	[c,ia,ib] = setxor(...)	%ia、ib 表示 c 中元素分别在 a (或 A)、b(或 B)中位置



例 1-35

```
>> A=[1 2 3 4];
>> B=[2 4 5 8];
>> C=setxor(A,B)
C =
    1     3     5     8
```

例 1-36

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
A =
    1     2     3     4
    1     2     4     6
    6     7     1     4
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
B =
    1     2     3     8
    1     1     4     6
    6     7     1     4
>> [C,ia,ib]=setxor(A,B,'rows')
C =
    1     1     4     6
    1     2     3     4
    1     2     3     8
    1     2     4     6
ia =
    1
    2
ib =
    2
    1
```

5. 两集合的并集

函数 **union**

格式 $c = \text{union}(a,b)$ %返回 a 、 b 的并集，即 $c = a \cup b$ 。

$c = \text{union}(A,B,'rows')$ %返回矩阵 A 、 B 不同行向量构成的大矩阵，其中相同行向量只取其一。

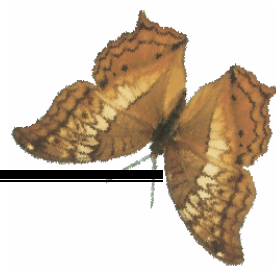
$[c,ia,ib] = \text{union}(\cdots)$ % ia 、 ib 分别表示 c 中行向量在原矩阵(向量)中的位置

例 1-37

```
>> A=[1 2 3 4];
>> B=[2 4 5 8];
>> c=union(A,B)
则结果为
c =
    1     2     3     4     5     8
```

例 1-38

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

```
>> [c,ia,ib]=union(A,B,'rows')
c =
     1     1     4     6
     1     2     3     4
     1     2     3     8
     1     2     4     6
ia =
     1
     2
ib =
     2
     1
```

6. 取集合的单值元素

函数

格式 `b = unique (a)` %取集合 a 的不重复元素构成的向量
 `b = unique (A,'rows')` %返回 A、B 不同行元素组成的矩阵
 `[b,i,j] = unique (···)` %i、j 体现 b 中元素在原向量（矩阵）中的位置

例 1-39

```
>> A=[1 1 2 2 4 4 6 4 6]
A =
     1     1     2     2     4     4     6     4     6
>> [c,i,j]=unique(A)
c =
     1     2     4     6
i =
     2     4     8     9
j =
     1     1     2     2     3     3     4     3     4
```

例 1-40

```
>> A=[1 2 2 4;1 1 4 6;1 1 4 6]
A =
     1     2     2     4
     1     1     4     6
     1     1     4     6
>> [c,i,j]=unique(A,'rows')
c =
     1     1     4     6
     1     2     2     4
i =
     3
     1
j =
     2
     1
     1
```

1.2.4 除法运算

Matlab 提供了两种除法运算：左除 (`\`) 和右除 (`/`)。一般情况下， $x=a \backslash b$ 是方程 $a * x = b$ 的解，而 $x=b/a$ 是方程 $x * a = b$ 的解。

例：a=[1 2 3;4 2 6;7 4 9]



```

b=[4; 1; 2];
x=a\b
则显示: x=
    -1.5000
     2.0000
     0.5000

```

如果 a 为非奇异矩阵, 则 $a \backslash b$ 和 b/a 可通过 a 的逆矩阵与 b 阵得到:

```

a\b = inv(a)*b
b/a = b*inv(a)

```

数组除法:

$A ./ B$ 表示 A 中元素与 B 中元素对应相除。

1.2.5 矩阵乘方

运算符: \wedge

运算规则:

(1) 当 A 为方阵, P 为大于 0 的整数时, A^P 表示 A 的 P 次方, 即 A 自乘 P 次; P 为小于 0 的整数时, A^P 表示 A^{-1} 的 P 次方。

(2) 当 A 为方阵, p 为非整数时, 则 $A^p = V \begin{bmatrix} d_{11}^p & & \\ & \ddots & \\ & & d_{nn}^p \end{bmatrix} V^{-1}$ 其中 V 为 A 的特征向

量, $\begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{nn} \end{bmatrix}$ 为特征值对角矩阵。如果有重根, 以上指令不成立。

(3) 标量的矩阵乘方 P^A , 标量的矩阵乘方定义为 $P^A = V \begin{bmatrix} p^{d_{11}} & & \\ & \ddots & \\ & & p^{d_{nn}} \end{bmatrix} V^{-1}$ 式中 V, D 取

自特征值分解 $AV=AD$ 。

(4) 标量的数组乘方 $P.^A$, 标量的数组乘方定义为 $P.^A = \begin{bmatrix} p^{a_{11}} & \dots & p^{a_{1n}} \\ \vdots & & \vdots \\ p^{a_{m1}} & \dots & p^{a_{mn}} \end{bmatrix}$ 数组乘方:

$A.^P$: 表示 A 的每个元素的 P 次乘方。

1.2.6 矩阵函数

命令 方阵指数

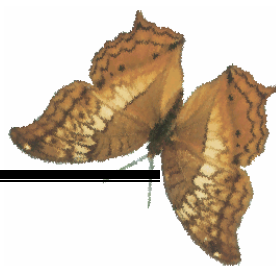
函数 **expm**

格式 $Y = \text{expm}(A)$ %使用 Pade 近似算法计算 e^A , 这是一个内部函数, A 为方阵。

$Y = \text{expm1}(A)$ %使用一个 M 文件和内部函数相同的算法计算 e^A

$Y = \text{expm2}(A)$ %使用泰勒级数计算 e^A

$Y = \text{expm3}(A)$ %使用特征值和特征向量计算 e^A



命令 矩阵的对数

函数 **logm**

格式 $Y = \text{logm}(X)$ %计算矩阵 X 的对数, 它是 $\text{expm}(X)$ 的反函数。
 $[Y, \text{esterr}] = \text{logm}(X)$ %esterr 为相对残差的估计值: $\text{norm}(\text{expm}(Y)-X)/\text{norm}(X)$

例 1-41

```
>> A=[1 1 0;0 0 2;0 0 -1];
>> Y=expm(A)
Y =
    2.7183    1.7183    1.0862
         0    1.0000    1.2642
         0         0    0.3679
>> A=logm(Y)
A =
    1.0000    1.0000    0.0000
         0         0    2.0000
         0         0   -1.0000
```

命令 方阵的函数

函数 **funm**

格式 $F = \text{funm}(A, \text{fun})$ % A 为方阵, 计算由 fun 指定的 A 的矩阵函数, fun 可以是任意基本函数, 如 \sin 、 \cos 等等, 例如:
 $\text{funm}(A, 'exp') = \text{expm}(A)$ 。

$[F, \text{esterr}] = \text{funm}(A, \text{fun})$ %esterr 为结果所产生的相对误差的估计值。

命令 矩阵的方根

函数 **sqrtm**

格式 $X = \text{sqrtm}(A)$ %矩阵 A 的平方根 $A^{1/2}$, 相当于 $X*X=A$, 求 X 。若 A 的特征值有非负实部, 则 X 是唯一的; 若 A 的特征值有负的实部, 则 X 为复矩阵; 若 A 为奇异矩阵, 则 X 不存在。

$[X, \text{resnorm}] = \text{sqrtm}(A)$ % resnorm 为结果产生的相对误差

$[X, \alpha, \text{condest}] = \text{sqrtm}(A)$ % α 为稳定因子, condest 为结果的条件数的估计值。

命令 矩阵 A 的多项式

函数 **polyvalm**

格式 $\text{polyvalm}(P, A)$ % P 为多项式系数向量, 方阵 A 为多项式变量, 返回多项式值。

1.2.7 矩阵转置

运算符: $'$

运算规则: 若矩阵 A 的元素为实数, 则与线性代数中矩阵的转置相同。

若 A 为复数矩阵, 则 A 转置后的元素由 A 对应元素的共轭复数构成。

若仅希望转置, 则用如下命令: $A.'$ 。

1.2.8 方阵的行列式

函数 **det**



格式 $d = \det(X)$ %返回方阵 X 的多项式的值

例 1-42

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> D=det(A)
D =
     0
```

1.2.9 逆与伪逆

命令 逆

函数 **inv**

格式 $Y = \text{inv}(X)$ %求方阵 X 的逆矩阵。若 X 为奇异阵或近似奇异阵，将给出警告信息。

例 1-43 求 $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 4 & 3 \end{pmatrix}$ 的逆矩阵

方法一

```
>> A=[1 2 3;2 2 1;3 4 3];
>> Y=inv(A)或 Y=A^(-1)
则结果显示为
Y =
     1.0000     3.0000    -2.0000
    -1.5000    -3.0000     2.5000
     1.0000     1.0000    -1.0000
```

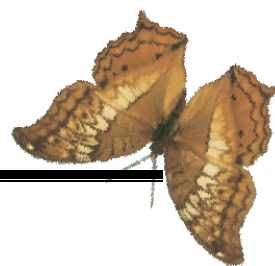
方法二：由增广矩阵 $B = \begin{pmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 1 & 0 \\ 3 & 4 & 3 & 0 & 0 & 1 \end{pmatrix}$ 进行初等行变换

```
>> B=[1, 2, 3, 1, 0, 0; 2, 2, 1, 0, 1, 0; 3, 4, 3, 0, 0, 1];
>> C=rref(B)    %化行最简形
>> X=C(:, 4:6)    %取矩阵 C 中的 A^(-1)部分
显示结果如下:
```

```
C =
     1.0000         0         0     1.0000     3.0000    -2.0000
         0     1.0000         0    -1.5000    -3.0000     2.5000
         0         0     1.0000     1.0000     1.0000    -1.0000
X =
     1.0000     3.0000    -2.0000
    -1.5000    -3.0000     2.5000
     1.0000     1.0000    -1.0000
```

例 1-44

```
>> A=[2 1 -1;2 1 2;1 -1 1];
>> format rat    %用有理格式输出
>> D=inv(A)
D =
     1/3         0         1/3
         0         1/3        -2/3
    -1/3         1/3         0
```



命令 伪逆

函数 **pinv**

格式 $B = \text{pinv}(A)$ %求矩阵 A 的伪逆

$B = \text{pinv}(A, \text{tol})$ %tol 为误差: $\max(\text{size}(A)) * \text{norm}(A) * \text{eps}$

说明 当矩阵为长方阵时, 方程 $AX=I$ 和 $XA=I$ 至少有一个无解, 这时 A 的伪逆能在某种程度上代表矩阵的逆, 若 A 为非奇异矩阵, 则 $\text{pinv}(A) = \text{inv}(A)$ 。

例 1-45

>> $A = \text{magic}(5);$ %产生 5 阶魔方阵。

>> $A = A(:, 1:4)$ %取 5 阶魔方阵的前 4 列元素构成矩阵 A 。

$A =$

17	24	1	8
23	5	7	14
4	6	13	20
10	12	19	21
11	18	25	2

>> $X = \text{pinv}(A)$ %计算 A 的伪逆

$X =$

-0.0041	0.0527	-0.0222	-0.0132	0.0069
0.0437	-0.0363	0.0040	0.0033	0.0038
-0.0305	0.0027	-0.0004	0.0068	0.0355
0.0060	-0.0041	0.0314	0.0211	-0.0315

1.2.10 矩阵的迹

函数 **trace**

格式 $b = \text{trace}(A)$ %返回矩阵 A 的迹, 即 A 的对角线元素之和。

1.2.11 矩阵和向量的范数

命令 向量的范数

函数 **norm**

格式 $n = \text{norm}(X)$ % X 为向量, 求欧几里德范数, 即 $\|X\|_2 = \sqrt{\sum |x_k|^2}$ 。

$n = \text{norm}(X, \text{inf})$ %求 ∞ -范数, 即 $\|X\| = \max(\text{abs}(X))$ 。

$n = \text{norm}(X, 1)$ %求 1-范数, 即 $\|X\|_1 = \sum_k |x_k|$ 。

$n = \text{norm}(X, -\text{inf})$ %求向量 X 的元素的绝对值的最小值, 即 $\|X\| = \min(\text{abs}(X))$ 。

$n = \text{norm}(X, p)$ %求 p -范数, 即 $\|X\|_p = \sqrt[p]{\sum_k |x_k|^p}$, 所以 $\text{norm}(X, 2) = \text{norm}(X)$ 。

命令 矩阵的范数

函数 **norm**

格式 $n = \text{norm}(A)$ % A 为矩阵, 求欧几里德范数 $\|A\|_2$, 等于 A 的最大奇异值。

$n = \text{norm}(A, 1)$ %求 A 的列范数 $\|A\|_1$, 等于 A 的列向量的 1-范数的最大值。

$n = \text{norm}(A, 2)$ %求 A 的欧几里德范数 $\|A\|_2$, 和 $\text{norm}(A)$ 相同。

$n = \text{norm}(A, \text{inf})$ %求行范数 $\|A\|_\infty$, 等于 A 的行向量的 1-范数的最大值



即: $\max(\text{sum}(\text{abs}(A')))$ 。

$n = \text{norm}(A, 'fro')$ %求矩阵 A 的 Frobenius 范数 $\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}$,

即 $\sqrt{\text{sum}(\text{diag}(A'*A))}$, 不能用矩阵 p-范数的定义来求。

命令 范数的估计值

函数 **normest**

格式 $\text{nrm} = \text{normest}(A)$ %矩阵 A 的 2-范数(欧几里德范数)的估计值, 相对误差小于 10^6 。

$\text{nrm} = \text{normest}(A, \text{tol})$ %tol 为指定相对误差

$[\text{nrm}, \text{count}] = \text{normest}(\dots)$ %count 给出计算估计值的迭代次数

1.2.12 条件数

命令 矩阵的条件数

函数 **cond**

格式 $c = \text{cond}(X)$ %求 X 的 2-范数的条件数, 即 X 的最大奇异值和最小奇异值的商。

$c = \text{cond}(X, p)$ %求 p-范数的条件数, p 的值可以是 1、2、inf 或者 'fro'。

说明 线性方程组 $AX=b$ 的条件数是一个大于或者等于 1 的实数, 用来衡量关于数据中的扰动, 也就是 A/或 b 对解 X 的灵敏度。一个差条件的方程组的条件数很大。条件数的定义为: $\text{cond}(A) = \|A\| \|A^{-1}\|$

命令 1-范数的条件数估计

函数 **condest**

格式 $c = \text{condest}(A)$ %方阵 A 的 1-范数的条件数的下界估值。

$[c, v] = \text{condest}(A)$ %v 为向量, 满足 $\|Av\| = \frac{\|A\| \cdot \|v\|}{c}$, 即 $\text{norm}(A*v, 1) = \text{norm}(A, 1) * \text{norm}(v, 1) / c$ 。

$[c, v] = \text{condest}(A, t)$ %求上面的 c 和 v, 同时显示出关于计算的步骤信息。如果 $t=1$, 则计算的每步都显示出来; 如果 $t=-1$, 则给出商 $c/\text{rcond}(A)$ 。

命令 矩阵可逆的条件数估值

函数 **rcond**

格式 $c = \text{rcond}(A)$ %对于差条件矩阵 A 来说, 给出一个接近于 0 的数; 对于好条件矩阵 A, 则给出一个接近于 1 的数。

命令 特征值的条件数

函数 **condeig**

格式 $c = \text{condeig}(A)$ %返回矩阵 A 的特征值的条件数

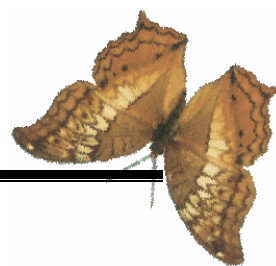
$[V, D, c] = \text{condeig}(A)$ %D 为 A 的特征值对角阵, V 为 A 的特征向量。

1.2.13 矩阵的秩

函数 **rank**

格式 $k = \text{rank}(A)$ %求矩阵 A 的秩





`k = rank(A, tol)` %tol 为给定误差

1.2.14 特殊运算

1. 矩阵对角线元素的抽取

函数 **diag**

格式 `X = diag(v, k)` %以向量 v 的元素作为矩阵 X 的第 k 条对角线元素，当 $k=0$ 时， v 为 X 的主对角线；当 $k>0$ 时， v 为上方第 k 条对角线；当 $k<0$ 时， v 为下方第 k 条对角线。

`X = diag(v)` %以 v 为主对角线元素，其余元素为 0 构成 X 。

`v = diag(X, k)` %抽取 X 的第 k 条对角线元素构成向量 v 。 $k=0$ ：抽取主对角线元素； $k>0$ ：抽取上方第 k 条对角线元素； $k<0$ 抽取下方第 k 条对角线元素。

`v = diag(X)` %抽取主对角线元素构成向量 v 。

例 1-46

```
>> v=[1 2 3];
>> x=diag(v,-1)
x =
     0     0     0     0
     1     0     0     0
     0     2     0     0
     0     0     3     0
>> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> v=diag(A,1)
v =
     2
     6
```

2. 上三角阵和下三角阵的抽取

函数 **tril** %取下三角部分

格式 `L = tril(X)` %抽取 X 的主对角线的下三角部分构成矩阵 L

`L = tril(X, k)` %抽取 X 的第 k 条对角线的下三角部分； $k=0$ 为主对角线； $k>0$ 为主对角线以上； $k<0$ 为主对角线以下。

函数 **triu** %取上三角部分

格式 `U = triu(X)` %抽取 X 的主对角线的上三角部分构成矩阵 U

`U = triu(X, k)` %抽取 X 的第 k 条对角线的上三角部分； $k=0$ 为主对角线； $k>0$ 为主对角线以上； $k<0$ 为主对角线以下。

例 1-47

```
>> A=ones(4)    %产生 4 阶全 1 阵
A =
     1     1     1     1
     1     1     1     1
     1     1     1     1
```



```

      1      1      1      1
>> L=tril(A,1) %取下三角部分
L =
      1      1      0      0
      1      1      1      0
      1      1      1      1
      1      1      1      1
>> U=triu(A,-1) %取上三角部分
U =
      1      1      1      1
      1      1      1      1
      0      1      1      1
      0      0      1      1

```

3. 矩阵的变维

矩阵的变维有两种方法，即用“:”和函数“reshape”，前者主要针对 2 个已知维数矩阵之间的变维操作；而后者是对于一个矩阵的操作。

(1) “:” 变维

例 1-48

```

> A=[1 2 3 4 5 6;6 7 8 9 0 1]
A =
      1      2      3      4      5      6
      6      7      8      9      0      1
>> B=ones(3,4)
B =
      1      1      1      1
      1      1      1      1
      1      1      1      1
>> B(:)=A(:)
B =
      1      7      4      0
      6      3      9      6
      2      8      5      1

```

(2) Reshape 函数变维

格式 $B = \text{reshape}(A, m, n)$ % 返回以矩阵 A 的元素构成的 $m \times n$ 矩阵 B

$B = \text{reshape}(A, m, n, p, \dots)$ % 将矩阵 A 变维为 $m \times n \times p \times \dots$

$B = \text{reshape}(A, [m \ n \ p \ \dots])$ % 同上

$B = \text{reshape}(A, \text{siz})$ % 由 siz 决定变维的大小，元素个数与 A 中元素个数相同。

例 1-49 矩阵变维

```

>> a=[1:12];
>> b=reshape(a,2,6)
b =
      1      3      5      7      9     11
      2      4      6      8     10     12

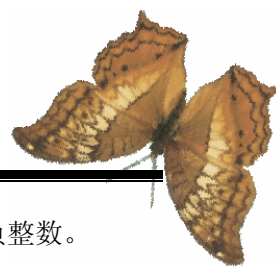
```

4. 矩阵的变向

(1) 矩阵旋转

函数

格式 $B = \text{rot90}(A)$ % 将矩阵 A 逆时针方向旋转 90°



$B = \text{rot90}(A, k)$ %将矩阵 A 逆时针方向旋转($k \times 90^\circ$), k 可取正负整数。

例 1-50

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> Y1=rot90(A),Y2=rot90(A,-1)
Y1 = %逆时针方向旋转
     3     6     9
     2     5     8
     1     4     7
Y2 = %顺时针方向旋转
     7     4     1
     8     5     2
     9     6     3
```

(2) 矩阵的左右翻转

函数 **fliplr**

格式 $B = \text{fliplr}(A)$ %将矩阵 A 左右翻转

(3) 矩阵的上下翻转

函数 **flipud**

格式 $B = \text{flipud}(A)$ %将矩阵 A 上下翻转

例 1-51

```
>> A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
>> B1=fliplr(A),B2=flipud(A)
B1 =
     3     2     1
     6     5     4
B2 =
     4     5     6
     1     2     3
```

(4) 按指定维数翻转矩阵

函数 **flipdim**

格式 $B = \text{flipdim}(A, \text{dim})$ % $\text{flipdim}(A, 1) = \text{flipud}(A)$, 并且 $\text{flipdim}(A, 2) = \text{fliplr}(A)$ 。

例 1-52

```
>> A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
>> B1=flipdim(A,1),B2=flipdim(A,2)
B1 =
     4     5     6
     1     2     3
B2 =
     3     2     1
     6     5     4
```



(5) 复制和平铺矩阵

函数 repmat

格式 `B = repmat(A,m,n)` %将矩阵 A 复制 $m \times n$ 块, 即 B 由 $m \times n$ 块 A 平铺而成。
`B = repmat(A,[m n])` %与上面一致
`B = repmat(A,[m n p...])` %B 由 $m \times n \times p \times \dots$ 个 A 块平铺而成
`repmat(A,m,n)` %当 A 是一个数 a 时, 该命令产生一个全由 a 组成的 $m \times n$ 矩阵。

例 1-53

```
>> A=[1 2;5 6]
A =
     1     2
     5     6
>> B=repmat(A,3,4)
B =
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
```

5. 矩阵的比较关系

矩阵的比较关系是针对于两个矩阵对应元素的, 所以在使用关系运算时, 首先应该保证两个矩阵的维数一致或其中一个矩阵为标量。关系运算是对两个矩阵的对应运算进行比较, 若关系满足, 则将结果矩阵中该位置元素置为 1, 否则置 0。

MATLAB 的各种比较关系运算有见表 1-1。

表 1-1

运算符	含义	运算符	含义
>	大于关系	<	大于关系
==	等于关系	>=	大于或等于关系
<=	小于或等于关系	~=	不等于关系

例 1-54

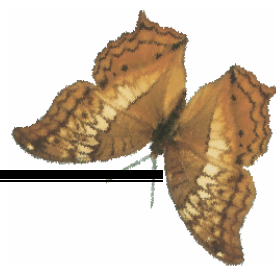
```
>> A=[1 2 3 4;5 6 7 8];B=[0 2 1 4;0 7 7 2];
>> C1=A==B, C2=A>=B, C3=A~=B
C1 =
     0     1     0     1
     0     0     1     0
C2 =
     1     1     1     1
     1     0     1     1
C3 =
     1     0     1     0
     1     1     0     1
```

6. 矩阵元素的数据变换

对于小数构成的矩阵 A 来说, 如果我们想对它取整数, 有以下几种方法:

(1) 按 $-\infty$ 方向取整**函数 floor**

格式 `floor(A)` %将 A 中元素按 $-\infty$ 方向取整, 即取不足整数。



(2) 按 $+\infty$ 方向取整

函数 **ceil**

格式 **ceil(A)** %将 A 中元素按 $+\infty$ 方向取整，即取过剩整数。

(3) 四舍五入取整

函数 **round**

格式 **round(A)** %将 A 中元素按最近的整数取整，即四舍五入取整。

(4) 按离 0 近的方向取整

函数 **fix**

格式 **fix(A)** %将 A 中元素按离 0 近的方向取整

例 1-55

```
>> A=-1.5+4*rand(3)
A =
    2.3005    0.4439    0.3259
   -0.5754    2.0652   -1.4260
    0.9274    1.5484    1.7856
>> B1=floor(A),B2=ceil(A),B3=round(A),B4=fix(A)
B1 =
     2     0     0
    -1     2    -2
     0     1     1
B2 =
     3     1     1
     0     3    -1
     1     2     2
B3 =
     2     0     0
    -1     2    -1
     1     2     2
B4 =
     2     0     0
     0     2    -1
     0     1     1
```

(5) 矩阵的有理数形式

函数 **rat**

格式 **[n,d]=rat(A)** %将 A 表示为两个整数矩阵相除，即 $A=n./d$ 。

例 1-56 对于上例中的 A

```
>> [n,d]=rat(A)
n =
    444     95    131
   -225    2059   -472
    166     48    1491
d =
    193    214    402
    391    997    331
    179     31    835
```

(6) 矩阵元素的余数

函数 **rem**

格式 **C = rem(A, x)** %表示 A 矩阵除以模数 x 后的余数。若 $x=0$ ，则定义 **rem(A,**



0)=NaN, 若 $x \neq 0$, 则整数部分由 $\text{fix}(A./x)$ 表示, 余数 $C=A-x.*\text{fix}(A./x)$ 。允许模 x 为小数。

7. 矩阵逻辑运算

设矩阵 A 和 B 都是 $m \times n$ 矩阵或其中之一为标量, 在 MATLAB 中定义了如下的逻辑运算:

(1) 矩阵的与运算

格式 $A \& B$ 或 $\text{and}(A, B)$

说明 A 与 B 对应元素进行与运算, 若两个数均非 0, 则结果元素的值为 1, 否则为 0。

(2) 或运算

格式 $A | B$ 或 $\text{or}(A, B)$

说明 A 与 B 对应元素进行或运算, 若两个数均为 0, 则结果元素的值为 0, 否则为 1。

(3) 非运算

格式 $\sim A$ 或 $\text{not}(A)$

说明 若 A 的元素为 0, 则结果元素为 1, 否则为 0。

(4) 异或运算

格式 $\text{xor}(A, B)$

说明 A 与 B 对应元素进行异或运算, 若相应的两个数中一个为 0, 一个非 0, 则结果为 1, 否则为 0。

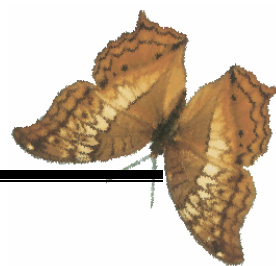
例 1-57

```
>> A=[0 2 3 4;1 3 5 0],B=[1 0 5 3;1 5 0 5]
A=
     0     2     3     4
     1     3     5     0
B=
     1     0     5     3
     1     5     0     5
>> C1=A&B,C2=A|B,C3=~A,C4=xor(A,B)
C1=
     0     0     1     1
     1     1     0     0
C2=
     1     1     1     1
     1     1     1     1
C3=
     1     0     0     0
     0     0     0     1
C4=
     1     1     0     0
     0     0     1     1
```

1.2.15 符号矩阵运算

1. 符号矩阵的四则运算

Matlab 6.x 抛弃了在 4.2 版中为符号矩阵设计的复杂函数形式, 把符号矩阵的四则运算简化为与数值矩阵完全相同的运算方式, 其运算符为: 加 (+)、减 (-)、乘 (×)、除 (/、\) 等或: 符号矩阵的和 (symadd)、差 (symsub)、乘 (symmul)。



```

例 1-58 >> A = sym('1/x, 1/(x+1); 1/(x+2), 1/(x+3)');
        >> B = sym('1/x, 1; x+2, 0');
        >> C=B-A
        >> D=a\b
则显示:
        C=
           x-1/x   1-1/(x+1)
        x+2-1/(x+2)   -1/(x+3)
        D=
        -6*x-2*x^3-7*x^2      1/2*x^3+x+3/2*x^2
        6+2*x^3+10*x^2+14*x   -2*x^2-3/2*x-1/2*x^3

```

2. 其他基本运算

符号矩阵的其他一些基本运算包括转置（'）、行列式（det）、逆（inv）、秩（rank）、幂（^）和指数（exp 和 expm）等都与数值矩阵相同

3. 将数值矩阵转化为符号矩阵

函数 sym

格式 B=sym(A) %将 A 转化为符号矩阵 B

例 1-59

```

>> A=[2/3,sqrt(2),0.222;1.4,1/0.23,log(3)]
A =
    0.6667    1.4142    0.2220
    1.4000    4.3478    1.0986
>> B=sym(A)
B =
    [ 2/3,      sqrt(2),      111/500]
    [ 7/5,      100/23,      4947709893870346*2^(-52)]

```

4. 符号矩阵的索引与修改

符号矩阵的索引与修改同数值矩阵的索引与修改完全相同，即用矩阵的坐标括号表达式实现。

例 1-60 对上例中的矩阵 B

```

>> B(2,3)      %矩阵的索引
ans =
    4947709893870346*2^(-52)
>> B(2,3)='log(7)' %矩阵的修改
B =
    [ 2/3, sqrt(2), 111/500]
    [ 7/5, 100/23, log(7)]

```

5. 符号矩阵的简化

符号工具箱中提供了符号矩阵因式分解、展开、合并、简化及通分等符号操作函数。

（1）因式分解

函数 factor

格式 factor(s) %符号表达式 s 的因式分解函数

说明 S 为符号矩阵或符号表达式，常用于多项式的因式分解。

例 1-61 将 x^9-1 分解因式

在 Matlab 命令窗口键入：

```

syms x
factor(x^9-1)

```



则显示: ans =

$$(x-1)*(x^2+x+1)*(x^6+x^3+1)$$

例 1-62 问“入”取何值时, 齐次方程组
$$\begin{cases} (1-\lambda)x_1 - 2x_2 + 4x_3 = 0 \\ 2x_1 + (3-\lambda)x_2 + x_3 = 0 \\ x_1 + x_2 + (1-\lambda)x_3 = 0 \end{cases}$$
 有非 0 解?

解: 在 Matlab 编辑器中建立 M 文件:

```
syms k
A=[1-k -2 4;2 3-k 1;1 1 1-k];
D=det(A)
factor(D)
```

其结果显示如下:

```
D =
-6*k+5*k^2-k^3
ans =
-k*(k-2)*(-3+k)
```

从而得到: 当 $k=0$ 、 $k=2$ 或 $k=3$ 时, 原方程组有非 0 解。

(2) 符号矩阵的展开

函数 **expand**

格式: `expand(s)` % 符号表达式 s 的展开函数

说明: s 为符号矩阵或表达式。常用在多项式的因式分解中, 也常用于三角函数, 指数函数和对数函数的展开中。

例 1-63 将 $(x+1)^3$ 、 $\sin(x+y)$ 展开

在 Matlab 编辑器中建立 M 文件:

```
syms x y
p=expand((x+1)^3)
q=expand(sin(x+y))
```

则结果显示为

```
p =
x^3+3*x^2+3*x+1
q =
sin(x)*cos(y)+cos(x)*sin(y)
```

(3) 同类项合并

函数 **Collect**

格式 `Collect(s,v)` % 将 s 中的变量 v 的同幂项系数合并

`Collect(s)` % s 是矩阵或表达式, 此命令对由命令 `findsym` 函数返回的默认变量进行同类项合并。

(4) 符号简化

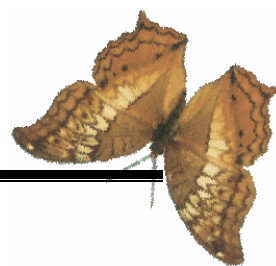
函数 **simple** 或 **simplify** % 寻找符号矩阵或符号表达式的最简型

格式 `simple(s)` % s 是矩阵或表达式

`[R,how]=simple(s)` % R 为返回的最简形, how 为简化过程中使用的主要方法。

说明 `Simple(s)` 将表达式 s 的长度化到最短。若还想让表达式更加精美, 可使用函数 `Pretty`。

格式 `Pretty(s)` % 使表达式 s 更加精美



例 1-64 计算行列式 $\begin{vmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \\ a^2 & b^2 & c^2 & d^2 \\ a^4 & b^4 & c^4 & d^4 \end{vmatrix}$ 的值。

在 Matlab 编辑器中建立 M 文件：

```
syms a b c d
A=[1 1 1 1;a b c d;a^2 b^2 c^2 d^2;a^4 b^4 c^4 d^4];
d1=det(A)
d2=simple(d1)      %化简表达式 d1
pretty(d2)         %让表达式 d2 符合人们的书写习惯
则显示结果如下：
d1 =
b*c^2*d^4-b*d^2*c^4-b^2*c*d^4+b^2*d*c^4+b^4*c*d^2-b^4*d*c^2-a*c^2*d^4+a*d^2*c^4+a*b
^2*d^4-a*b^2*c^4-a*b^4*d^2+a*b^4*c^2+a^2*c*d^4-a^2*d*c^4-a^2*b*d^4+a^2*b*c^4+a^2*b^4*d-a
^2*b^4*c-a^4*c*d^2+a^4*d*c^2+a^4*b*d^2-a^4*b*c^2-a^4*b^2*d+a^4*b^2*c
d2 =
(-d+c)*(b-d)*(b-c)*(-d+a)*(a-c)*(a-b)*(a+c+d+b)
(-d+c)(b-d)(b-c)(-d+a)(a-c)(a-b)(a+c+d+b)
```

1.2.16 矩阵元素个数的确定

函数 **numel**

格式 `n = numel(a)` % 计算矩阵 A 中元素的个数

例 1-65

```
>> A=[1 2 3 4;5 6 7 8];
>> n=numel(A)
n =
8
```

1.3 矩阵分解

1.3.1 Cholesky 分解

函数 **chol**

格式 `R = chol(X)` % 如果 X 为 n 阶对称正定矩阵，则存在一个实的非奇异上三角阵 R，满足 $R^*R = X$ ；若 X 非正定，则产生错误信息。

`[R,p] = chol(X)` % 不产生任何错误信息，若 X 为正定阵，则 $p=0$ ，R 与上相同；若 X 非正定，则 p 为正整数，R 是有序的上三角阵。

例 1-66

```
>> X=pascal(4)      %产生 4 阶 pascal 矩阵
X =
1     1     1     1
1     2     3     4
1     3     6    10
1     4    10    20
>> [R,p]=chol(X)
R =
```



```

1     1     1     1
0     1     2     3
0     0     1     3
0     0     0     1
p =
0

```

1.3.2 LU 分解

矩阵的三角分解又称 LU 分解，它的目的是将一个矩阵分解成一个下三角矩阵 L 和一个上三角矩阵 U 的乘积，即 $A=LU$ 。

函数 **lu**

格式 $[L,U]=lu(X)$ % U 为上三角阵， L 为下三角阵或其变换形式，满足 $LU=X$ 。
 $[L,U,P]=lu(X)$ % U 为上三角阵， L 为下三角阵， P 为单位矩阵的行变换矩阵，满足 $LU=PX$ 。

例 1-67

```

>> A=[1 2 3;4 5 6;7 8 9];
>> [L,U]=lu(A)
L =
    0.1429    1.0000         0
    0.5714    0.5000    1.0000
    1.0000         0         0
U =
    7.0000    8.0000    9.0000
         0    0.8571    1.7143
         0         0    0.0000
>> [L,U,P]=lu(A)
L =
    1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000
U =
    7.0000    8.0000    9.0000
         0    0.8571    1.7143
         0         0    0.0000
P =
         0         0         1
         1         0         0
         0         1         0

```

1.3.3 QR 分解

将矩阵 A 分解成一个正交矩阵与一个上三角矩阵的乘积。

函数 **qr**

格式 $[Q,R]=qr(A)$ %求得正交矩阵 Q 和上三角阵 R ， Q 和 R 满足 $A=QR$ 。
 $[Q,R,E]=qr(A)$ %求得正交矩阵 Q 和上三角阵 R ， E 为单位矩阵的变换形式， R 的对角线元素按大小降序排列，满足 $AE=QR$ 。
 $[Q,R]=qr(A,0)$ %产生矩阵 A 的“经济大小”分解
 $[Q,R,E]=qr(A,0)$ % E 的作用是使得 R 的对角线元素降序，且 $Q^*R=A(:, E)$ 。
 $R=qr(A)$ %稀疏矩阵 A 的分解，只产生一个上三角阵 R ，满足 $R^*R =$



A^*A , 这种方法计算 A^*A 时减少了内在数字信息的损耗。

`[C,R] = qr(A,b)` %用于稀疏最小二乘问题: $\text{minimize} \|Ax-b\|$ 的两步解: `[C,R] = qr(A,b)`, $x = R \backslash c$ 。

`R = qr(A,0)` %针对稀疏矩阵 A 的经济型分解

`[C,R] = qr(A,b,0)` %针对稀疏最小二乘问题的经济型分解

例 1-68

```
>> A=[ 1  2  3;4  5  6;7  8  9;10 11 12];
>> [Q,R] = qr(A)
Q =
   -0.0776   -0.8331    0.5444    0.0605
   -0.3105   -0.4512   -0.7709    0.3251
   -0.5433   -0.0694   -0.0913   -0.8317
   -0.7762    0.3124    0.3178    0.4461
R =
  -12.8841   -14.5916   -16.2992
         0    -1.0413    -2.0826
         0         0     0.0000
         0         0         0
```

函数 `qrdelete`

格式 `[Q,R] = qrdelete(Q,R,j)` %返回将矩阵 A 的第 j 列移去后的新矩阵的 qr 分解

例 1-69

```
>> A=[-149 -50 -154;537 180 546;-27 -9 -25];
>> [Q,R]=qr(A)
Q =
   -0.2671   -0.7088    0.6529
    0.9625   -0.1621    0.2176
   -0.0484    0.6865    0.7255
R =
   557.9418   187.0321   567.8424
         0    0.0741    3.4577
         0         0    0.1451
>> [Q,R]=qrdelete(Q,R,3)    %将 A 的第 3 列去掉后进行 qr 分解。
Q =
   -0.2671   -0.7088    0.6529
    0.9625   -0.1621    0.2176
   -0.0484    0.6865    0.7255
R =
   557.9418   187.0321
         0    0.0741
         0         0
```

函数 `qrinsert`

格式 `[Q,R] = qrinsert(Q,R,j,x)` %在矩阵 A 中第 j 列插入向量 x 后的新矩阵进行 qr 分解。若 j 大于 A 的列数, 表示在 A 的最后插入列 x。

例 1-70

```
>> A=[-149 -50 -154;537 180 546;-27 -9 -25];
>> x=[35 10 7]';
>> [Q,R]=qrinsert(Q,R,4,x)
Q =
   -0.2671   -0.7088    0.6529
```



```

    0.9625   -0.1621    0.2176
   -0.0484    0.6865    0.7255
R =
    557.9418   187.0321   567.8424   -0.0609
         0     0.0741    3.4577   -21.6229
         0         0     0.1451   30.1073

```

1.3.4 Schur 分解

函数 **schur**

格式 **T = schur(A)** %产生 schur 矩阵 T, 即 T 的主对角线元素为特征值的三角阵。

T = schur(A,flag) %若 A 有复特征根, 则 flag='complex', 否则 flag='real'。

[U,T] = schur(A,···) %返回正交矩阵 U 和 schur 矩阵 T, 满足 $A = U * T * U'$ 。

例 1-71

```

>> H = [-149  -50  -154; 537  180  546; -27  -9  -25];
>> [U,T]=schur(H)
U =
    0.3162   -0.6529    0.6882
   -0.9487   -0.2176    0.2294
    0.0000    0.7255    0.6882
T =
    1.0000   -7.1119 -815.8706
         0     2.0000  -55.0236
         0         0     3.0000

```

1.3.5 实 Schur 分解转化成复 Schur 分解

函数 **rsf2csf**

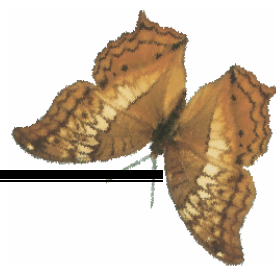
格式 **[U,T] = rsf2csf (U,T)** %将实舒尔形式转化成复舒尔形式

例 1-72

```

>> A=[1 1 1 3;1 2 1 1;1 1 3 1;-2 1 1 4];
>> [u,t]=schur (A)
u =
   -0.4916   -0.4900   -0.6331   -0.3428
   -0.4980    0.2403   -0.2325    0.8001
   -0.6751    0.4288    0.4230   -0.4260
   -0.2337   -0.7200    0.6052    0.2466
t =
    4.8121    1.1972   -2.2273   -1.0067
         0    1.9202   -3.0485   -1.8381
         0    0.7129    1.9202    0.2566
         0         0         0    1.3474
>> [U,T]=rsf2csf (u,t)
U =
   -0.4916          -0.2756 - 0.4411i    0.2133 + 0.5699i   -0.3428
   -0.4980          -0.1012 + 0.2163i   -0.1046 + 0.2093i    0.8001
   -0.6751          0.1842 + 0.3860i   -0.1867 - 0.3808i   -0.4260
   -0.2337          0.2635 - 0.6481i    0.3134 - 0.5448i    0.2466
T =
    4.8121          -0.9697 + 1.0778i   -0.5212 + 2.0051i   -1.0067
         0          1.9202 + 1.4742i    2.3355              0.1117 + 1.6547i
         0              0              1.9202 - 1.4742i    0.8002 + 0.2310i
         0              0              0              1.3474

```



1.3.6 特征值分解

函数 **eig**

格式 $d = \text{eig}(A)$ %求矩阵 A 的特征值 d ，以向量形式存放 d 。
 $d = \text{eig}(A,B)$ % A 、 B 为方阵，求广义特征值 d ，以向量形式存放 d 。
 $[V,D] = \text{eig}(A)$ %计算 A 的特征值对角阵 D 和特征向量 V ，使 $AV=VD$ 成立。
 $[V,D] = \text{eig}(A,\text{'nobalance'})$ %当矩阵 A 中有与截断误差数量级相差不远的值时，该指令可能更精确。 'nobalance' 起误差调节作用。
 $[V,D] = \text{eig}(A,B)$ %计算广义特征值向量阵 V 和广义特征值阵 D ，满足 $AV=BVD$ 。
 $[V,D] = \text{eig}(A,B,\text{flag})$ % 由 flag 指定算法计算特征值 D 和特征向量 V ， flag 的可能值为： 'chol' 表示对 B 使用 Cholesky 分解算法，这里 A 为对称 Hermitian 矩阵， B 为正定阵。 'qz' 表示使用 QZ 算法，这里 A 、 B 为非对称或非 Hermitian 矩阵。

说明 一般特征值问题是求解方程： $Ax = \lambda x$ 解的问题。广义特征值问题是求方程： $Ax = \lambda Bx$ 解的问题。

1.3.7 奇异值分解

函数 **svd**

格式 $s = \text{svd}(X)$ %返回矩阵 X 的奇异值向量
 $[U,S,V] = \text{svd}(X)$ %返回一个与 X 同大小的对角矩阵 S ，两个酉矩阵 U 和 V ，且满足 $U*S*V'$ 。若 A 为 $m \times n$ 阵，则 U 为 $m \times m$ 阵， V 为 $n \times n$ 阵。奇异值在 S 的对角线上，非负且按降序排列。
 $[U,S,V] = \text{svd}(X,0)$ %得到一个“有效大小”的分解，只计算出矩阵 U 的前 n 列，矩阵 S 的大小为 $n \times n$ 。

例 1-73

```
>> A=[1 2;3 4;5 6;7 8];
>> [U,S,V]=svd(A)
U =
   -0.1525   -0.8226   -0.3945   -0.3800
   -0.3499   -0.4214    0.2428    0.8007
   -0.5474   -0.0201    0.6979   -0.4614
   -0.7448    0.3812   -0.5462    0.0407
S =
   14.2691         0
         0    0.6268
         0         0
         0         0
V =
   -0.6414    0.7672
   -0.7672   -0.6414
>> [U,S,V]=svd(A,0)
U =
   -0.1525   -0.8226
   -0.3499   -0.4214
```



```

-0.5474 -0.0201
-0.7448 0.3812
S =
14.2691 0
0 0.6268
V =
-0.6414 0.7672
-0.7672 -0.6414

```

1.3.8 广义奇异值分解

函数 **gsvd**

格式 `[U,V,X,C,S] = gsvd(A,B)` % 返回酉矩阵 **U** 和 **V**、一个普通方阵 **X**、非负对角矩阵 **C** 和 **S**, 满足 $A = U * C * X'$, $B = V * S * X'$, $C' * C + S' * S = I$ (**I** 为单位矩阵); **A** 和 **B** 的列数必须相同, 行数可以不同。

`[U,V,X,C,S] = gsvd(A,B,0)` % 含义与前面相似

`sigma = gsvd(A,B)` % 返回广义奇异值 **sigma**

例 1-74

```
>> A=reshape(1:12,3,4) % 产生 3 行 4 列矩阵, 元素由 1, 2, ..., 12 构成。
```

```
A =
1 4 7 10
2 5 8 11
3 6 9 12
```

```
>> B=magic(4) % 产生 4 阶魔方阵
```

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

```
>> [U,V,X,C,S]=gsvd(A,B)
```

```
U =
0.4082 0.7071 0.5774
-0.8165 0.0000 0.5774
0.4082 -0.7071 0.5774

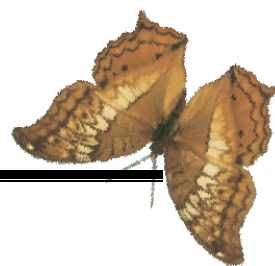
V =
0.2607 -0.7950 -0.5000 0.2236
-0.4029 0.3710 -0.5000 0.6708
-0.5452 -0.0530 -0.5000 -0.6708
0.6874 0.4770 -0.5000 -0.2236

X =
0 -9.4340 -17.0587 3.4641
1.8962 8.7980 -17.0587 8.6603
3.7924 8.1620 -17.0587 13.8564
-5.6885 -7.5260 -17.0587 19.0526

C =
0 0.0000 0 0
0 0 0.0829 0
0 0 0 1.0000

S =
1.0000 0 0 0
0 1.0000 0 0

```



```

0      0      0.9966      0
0      0      0      0.0000

```

1.3.9 特征值问题的 QZ 分解

函数 **qz**

格式 **[AA,BB,Q,Z,V] = qz(A,B)**

%A、B 为方阵，产生上三角阵 AA 和 BB，正交矩阵 Q、Z 或其列变换形式，V 为特征向量阵。且满足： $Q^*A^*Z=AA$ 和 $Q^*B^*Z=BB$ 。

[AA,BB,Q,Z,V] = qz(A,B,flag)

%产生由 flag 决定的分解结果，flag 取值为：
'complex': 表示复数分解（默认），取值为'real':
表示实数分解。

1.3.10 海森伯格形式的分解

如果矩阵 H 的第一子对角线下元素都是 0，则 H 为海森伯格(Hessenberg)矩阵。如果矩阵是对称矩阵，则它的海森伯格形式是对角三角阵。MATLAB 可以通过相似变换将矩阵变换成这种形式。

函数 **hess**

格式 **H = hess(A)** %返回矩阵 A 的海森伯格形式

[P,H] = hess(A) %P 为酉矩阵，满足： $A = PHP'$ 且 $P'P = \text{eye}(\text{size}(A))$ 。

例 1-75

```
>> A=[-149 -50 -154;537 180 546;-27 -9 -25];
```

```
>> [P,H]=hess(A)
```

```

P =
1.0000      0      0
      0 -0.9987  0.0502
      0  0.0502  0.9987

H =
-149.0000  42.2037 -156.3165
-537.6783 152.5511 -554.9272
      0    0.0728   2.4489

```

H 的第一子对角元素是 $H(3,1)=0$ 。

1.4 线性方程的组的求解

我们将线性方程的求解分为两类：一类是方程组求唯一解或求特解，另一类是方程组求无穷解即通解。可以通过系数矩阵的秩来判断：

若系数矩阵的秩 $r=n$ (n 为方程组中未知变量的个数)，则有唯一解；

若系数矩阵的秩 $r<n$ ，则可能有无穷解；

线性方程组的无穷解 = 对应齐次方程组的通解+非齐次方程组的一个特解；其特解的求法属于解的第一类问题，通解部分属第二类问题。

1.4.1 求线性方程组的唯一解或特解（第一类问题）

这类问题的求法分为两类：一类主要用于解低阶稠密矩阵 —— 直接法；另一类是解大



型稀疏矩阵 —— 迭代法。

1. 利用矩阵除法求线性方程组的特解（或一个解）

方程： $AX=b$

解法： $X=A\backslash b$

例 1-76 求方程组
$$\begin{cases} 5x_1 + 6x_2 & = 1 \\ x_1 + 5x_2 + 6x_3 & = 0 \\ x_2 + 5x_3 + 6x_4 & = 0 \\ x_3 + 5x_4 + 6x_5 & = 0 \\ x_4 + 5x_5 & = 1 \end{cases}$$
 的解。

解：

```
>>A=[5 6 0 0 0
      1 5 6 0 0
      0 1 5 6 0
      0 0 1 5 6
      0 0 0 1 5];
B=[1 0 0 0 1]';
R_A=rank(A) %求秩
X=A\B %求解
```

运行后结果如下

```
R_A =
      5
X =
      2.2662
     -1.7218
      1.0571
     -0.5940
      0.3188
```

这就是方程组的解。

或用函数 `rref` 求解：

```
>> C=[A,B] %由系数矩阵和常数列构成增广矩阵 C
>> R=rref(C) %将 C 化成行最简行
R =
      1.0000         0         0         0         0      2.2662
         0      1.0000         0         0         0     -1.7218
         0         0      1.0000         0         0      1.0571
         0         0         0      1.0000         0     -0.5940
         0         0         0         0      1.0000      0.3188
```

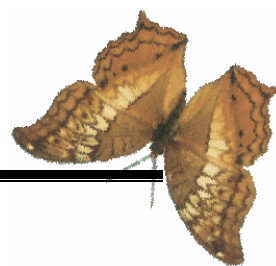
则 R 的最后一列元素就是所求之解。

例 1-77 求方程组
$$\begin{cases} x_1 + x_2 - 3x_3 - x_4 = 1 \\ 3x_1 - x_2 - 3x_3 + 4x_4 = 4 \\ x_1 + 5x_2 - 9x_3 - 8x_4 = 0 \end{cases}$$
 的一个特解。

解：

```
>>A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
>>B=[1 4 0]';
>>X=A\B %由于系数矩阵不满秩，该解法可能存在误差。
X=[0 0 -0.5333 0.6000]'（一个特解近似值）。
```

若用 `rref` 求解，则比较精确：



```
>> A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
      B=[1 4 0]';
>> C=[A,B]; %构成增广矩阵
>> R=rref(C)
R =
    1.0000         0   -1.5000    0.7500    1.2500
         0    1.0000   -1.5000   -1.7500   -0.2500
         0         0         0         0         0
```

由此得解向量 $X=[1.2500 \ -0.2500 \ 0 \ 0]'$ (一个特解)。

2. 利用矩阵的 LU、QR 和 cholesky 分解求方程组的解

(1) LU 分解:

LU 分解又称 Gauss 消去分解, 可把任意方阵分解为下三角矩阵的基本变换形式(行交换)和上三角矩阵的乘积。即 $A=LU$, L 为下三角阵, U 为上三角阵。

则: $A*X=b$ 变成 $L*U*X=b$

所以 $X=U \setminus (L \setminus b)$ 这样可以大大提高运算速度。

命令 $[L, U]=lu(A)$

例 1-78 求方程组 $\begin{cases} 4x_1 + 2x_2 - x_3 = 2 \\ 3x_1 - x_2 + 2x_3 = 10 \\ 11x_1 + 3x_2 = 8 \end{cases}$ 的一个特解。

解:

$$A = \begin{pmatrix} 4 & 2 & -1 \\ 3 & -1 & 2 \\ 11 & 3 & 0 \end{pmatrix} \quad b = [2, 10, 8]'$$

```
>> A=[4 2 -1;3 -1 2;11 3 0];
>> B=[2 10 8]';
>> D=det(A)
>> [L,U]=lu(A)
>> X=U \ (L \ B)
```

显示结果如下:

```
D =
    0

L =
    0.3636   -0.5000    1.0000
    0.2727    1.0000         0
    1.0000         0         0

U =
   11.0000    3.0000         0
         0   -1.8182    2.0000
         0         0    0.0000
```

Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 2.018587e-017.

> In D:\Matlab\pujun\lx0720.m at line 4

```
X =
    1.0e+016 *
   -0.4053
    1.4862
    1.3511
```

说明 结果中的警告是由于系数行列式为零产生的。可以通过 $A*X$ 验证其正确性。



(2) Cholesky 分解

若 A 为对称正定矩阵, 则 Cholesky 分解可将矩阵 A 分解成上三角矩阵和其转置的乘积,
即: $A = R' * R$ 其中 R 为上三角阵。

方程 $A * X = b$ 变成 $R' * R * X = b$

所以 $X = R \setminus (R' \setminus b)$

(3) QR 分解

对于任何长方矩阵 A , 都可以进行 QR 分解, 其中 Q 为正交矩阵, R 为上三角矩阵的初等变换形式, 即: $A = QR$

方程 $A * X = b$ 变形成 $QRX = b$

所以 $X = R \setminus (Q \setminus b)$

上例中 $[Q, R] = \text{qr}(A)$

$X = R \setminus (Q \setminus b)$

说明 这三种分解, 在求解大型方程组时很有用。其优点是运算速度快、可以节省磁盘空间、节省内存。

1.4.2 求线性齐次方程组的通解

在 Matlab 中, 函数 `null` 用来求解零空间, 即满足 $A * X = 0$ 的解空间, 实际上是求出解空间的一组基 (基础解系)。

格式 `z = null` % z 的列向量为方程组的正交规范基, 满足 $Z' * Z = I$ 。

`z = null(A, 'r')` % z 的列向量是方程 $AX=0$ 的有理基

例 1-79 求解方程组的通解:
$$\begin{cases} x_1 + 2x_2 + 2x_3 + x_4 = 0 \\ 2x_1 + x_2 - 2x_3 - 2x_4 = 0 \\ x_1 - x_2 - 4x_3 - 3x_4 = 0 \end{cases}$$

解:

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

```
>>format rat %指定有理式格式输出
```

```
>>B=null(A,'r') %求解空间的有理基
```

运行后显示结果如下:

```
B =
      2      5/3
     -2     -4/3
      1      0
      0      1
```

或通过行最简行得到基:

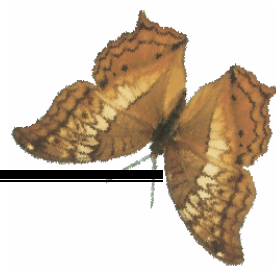
```
>> B=rref(A)
```

```
B =
    1.0000      0    -2.0000   -1.6667
      0    1.0000     2.0000    1.3333
      0      0      0      0
```

即可写出其基础解系 (与上面结果一致)。

写出通解:

```
syms k1 k2
```

```
X=k1*B(:,1)+k2*B(:,2)    %写出方程组的通解
pretty(X)                 %让通解表达式更加精美
```

运行后结果如下：

```
X =
[ 2*k1+5/3*k2]
[-2*k1-4/3*k2]
[          k1]
[          k2]
% 下面是其简化形式
[2k1 + 5/3k2 ]
[          ]
[-2k1 - 4/3k2]
[          ]
[          k1]
[          ]
[          k2]
[          ]
```

1.4.3 求非齐次线性方程组的通解

非齐次线性方程组需要先判断方程组是否有解，若有解，再去求通解。

因此，步骤为：

第一步：判断 $AX=b$ 是否有解，若有解则进行第二步

第二步：求 $AX=b$ 的一个特解

第三步：求 $AX=0$ 的通解

第四步： $AX=b$ 的通解= $AX=0$ 的通解+ $AX=b$ 的一个特解。

例 1-80 求解方程组
$$\begin{cases} x_1 - 2x_2 + 3x_3 - x_4 = 1 \\ 3x_1 - x_2 + 5x_3 - 3x_4 = 2 \\ 2x_1 + x_2 + 2x_3 - 2x_4 = 3 \end{cases}$$

解：在 Matlab 中建立 M 文件如下：

```
A=[1 -2 3 -1;3 -1 5 -3;2 1 2 -2];
b=[1 2 3]';
B=[A b];
n=4;
R_A=rank(A)
R_B=rank(B)
format rat
if R_A==R_B&R_A==n    %判断有唯一解
    X=A\b
elseif R_A==R_B&R_A<n    %判断有无穷解
    X=A\b    %求特解
    C=null(A,'r')    %求 AX=0 的基础解系
else X='equation no solve'    %判断无解
end
```

运行后结果显示：

```
R_A =
    2
R_B =
    3
X =
equation no solve
```



说明 该方程组无解

例 1-81 求解方程组的通解:
$$\begin{cases} x_1 + x_2 - 3x_3 - x_4 = 1 \\ 3x_1 - x_2 - 3x_3 + 4x_4 = 4 \\ x_1 + 5x_2 - 9x_3 - 8x_4 = 0 \end{cases}$$

解法一: 在 Matlab 编辑器中建立 M 文件如下:

```
A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
b=[1 4 0]';
B=[A b];
n=4;
R_A=rank(A)
R_B=rank(B)
format rat
if R_A==R_B&R_A==n
    X=A\b
elseif R_A==R_B&R_A<n
    X=A\b
    C=null(A,'r')
else X='Equation has no solves'
end
```

运行后结果显示为:

```
R_A =
    2
R_B =
    2
Warning: Rank deficient, rank = 2   tol = 8.8373e-015.
> In D:\Matlab\pujun\lx0723.m at line 11
X =
    0
    0
   -8/15
    3/5
C =
    3/2    -3/4
    3/2     7/4
     1         0
     0         1
```

所以原方程组的通解为
$$X = k_1 \begin{pmatrix} 3/2 \\ 3/2 \\ 1 \\ 0 \end{pmatrix} + k_2 \begin{pmatrix} -3/4 \\ 7/4 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -8/15 \\ 3/5 \end{pmatrix}$$

解法二: 用 rref 求解

```
A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
b=[1 4 0]';
B=[A b];
C=rref(B) %求增广矩阵的行最简形, 可得最简同解方程组。
```

运行后结果显示为:

```
C =
    1         0   -3/2    3/4    5/4
    0         1   -3/2   -7/4   -1/4
    0         0         0         0         0
```



对应齐次方程组的基础解系为: $\xi_1 = \begin{pmatrix} 3/2 \\ 3/2 \\ 1 \\ 0 \end{pmatrix}$, $\xi_2 = \begin{pmatrix} -3/4 \\ 7/4 \\ 0 \\ 1 \end{pmatrix}$ 非齐次方程组的特解为:

$\eta^* = \begin{pmatrix} 5/4 \\ -1/4 \\ 0 \\ 0 \end{pmatrix}$ 所以, 原方程组的通解为: $X = k_1 \xi_1 + k_2 \xi_2 + \eta^*$ 。

1.4.4 线性方程组的 LQ 解法

函数 **symmlq**

格式 $x = \text{symmlq}(A, b)$ %求线性方程组 $AX=b$ 的解 X 。A 必须为 n 阶对称方阵, b 为 n 元列向量。A 可以由 **afun** 定义并返回 $A*X$ 的函数。如果收敛, 将显示结果信息; 如果收敛失败, 将给出警告信息并显示相对残差 $\text{norm}(b-A*x)/\text{norm}(b)$ 和计算终止的迭代次数。

$\text{symmlq}(A, b, \text{tol})$ %指定误差 **tol**, 默认值是 $1e-6$ 。

$\text{symmlq}(A, b, \text{tol}, \text{maxit})$ %**maxit** 指定最大迭代次数

$\text{symmlq}(A, b, \text{tol}, \text{maxit}, M)$ %**M** 为用于对称正定矩阵的预处理因子

$\text{symmlq}(A, b, \text{tol}, \text{maxit}, M1, M2)$ %**M**=**M1**×**M2**

$\text{symmlq}(A, b, \text{tol}, \text{maxit}, M1, M2, x0)$ %**x0** 为初始估计值, 默认值为 0。

$[x, \text{flag}] = \text{symmlq}(A, b, \dots)$ %**flag** 的取值为: 0 表示在指定迭代次数之内按要求精度收敛; 1 表示在指定迭代次数内不收敛; 2 表示 **M** 为坏条件的预处理因子; 3 表示两次连续迭代完全相同; 4 表示标量参数太小或太大; 5 表示预处理因子不是对称正定的。

$[x, \text{flag}, \text{relres}] = \text{symmlq}(A, b, \dots)$ % **relres** 表示相对误差 $\text{norm}(b-A*x)/\text{norm}(b)$

$[x, \text{flag}, \text{relres}, \text{iter}] = \text{symmlq}(A, b, \dots)$ %**iter** 表示计算 x 的迭代次数

$[x, \text{flag}, \text{relres}, \text{iter}, \text{resvec}] = \text{symmlq}(A, b, \dots)$ %**resvec** 表示每次迭代的残差:
 $\text{norm}(b-A*x0)$

$[x, \text{flag}, \text{relres}, \text{iter}, \text{resvec}, \text{resveccg}] = \text{symmlq}(A, b, \dots)$ %**resveccg** 表示每次迭代共轭梯度残差的范数

1.4.5 双共轭梯度法解方程组

函数 **bicg**

格式 $x = \text{bicg}(A, b)$ %求线性方程组 $AX=b$ 的解 X 。A 必须为 n 阶方阵, b 为 n 元列向量。A 可以由 **afun** 定义并返回 $A*X$ 的函数。如果收敛, 将显示结果信息; 如果收敛失败, 将给出警告信息并显示相对残差 $\text{norm}(b-A*x)/\text{norm}(b)$ 和计算终止的迭代次数。

$\text{bicg}(A, b, \text{tol})$ %指定误差 **tol**, 默认值是 $1e-6$ 。



```

bicg(A,b,tol,maxit)           %maxit 指定最大迭代次数
bicg(A,b,tol,maxit,M)        %M 为用于对称正定矩阵的预处理因子
bicg(A,b,tol,maxit,M1,M2)    %M=M1×M2
bicg(A,b,tol,maxit,M1,M2,x0) %x0 为初始估计值，默认值为 0。
[x,flag] = bicg(A,b,...)     %flag 的取值为：0 表示在指定迭代次数之内按要求精度
                             收敛；1 表示在指定迭代次数内不收敛；2 表示 M 为坏条
                             件的预处理因子；3 表示两次连续迭代完全相同；4 表示
                             标量参数太小或太大。

[x,flag,relres] = bicg(A,b,...) % relres 表示相对误差  $\text{norm}(b-A*x)/\text{norm}(b)$ 
[x,flag,relres,iter] = bicg(A,b,...) %iter 表示计算 x 的迭代次数
[x,flag,relres,iter,resvec] = bicg(A,b,...) %resvec 表示每次迭代的残差：
                                            $\text{norm}(b-A*x_0)$ 

```

例 1-83 调用 MATLAB6.0 数据文件 west0479。

```

>> load west0479
>> A=west0479; %将数据取为系数矩阵 A。
>> b=sum(A,2); %将 A 的各行求和，构成一列向量。
>> X=A\b; %用 “\” 求  $AX=b$  的解。
>> norm(b-A*X)/norm(b) %计算解的相对误差。
ans =
    1.2454e-017
>> [x,flag,relres,iter,resvec] = bicg(A,b) %用 bicg 函数求解。
x = (全为 0，由于太长，不显示出来)
flag =
    1 %表示在默认迭代次数（20 次）内不收敛。
relres =
    1 %相对残差  $\text{relres} = \text{norm}(b-A*x)/\text{norm}(b) = \text{norm}(b)/\text{norm}(b) = 1$ 。
iter =
    0 %表明解法不当，使得初始估计值 0 向量比后来所有迭代值都好。
resvec = (略) %每次迭代的残差。
>> semilogy(0:20,resvec/norm(b),'-o') %作每次迭代的相对残差图形，结果如下图。
>> xlabel('iteration number') %x 轴为迭代次数。
>> ylabel('relative residual') %y 轴为相对残差。

```

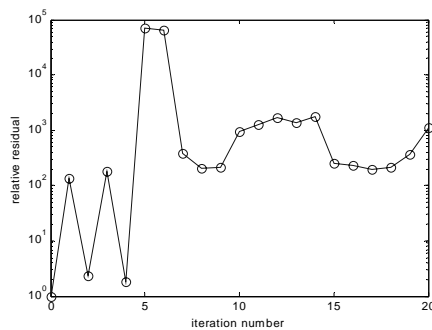
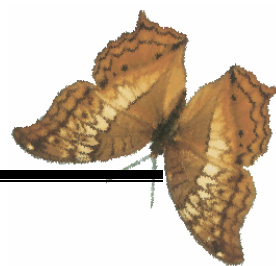


图 1-1 双共轭梯度法相对误差图

1.4.6 稳定双共轭梯度方法解方程组

函数 **bicgstab**





格式 `x=bicgstab(A,b)`
`bicgstab(A,b,tol)`
`bicgstab(A,b,tol,maxit)`
`bicgstab(A,b,tol,maxit,M)`
`bicgstab(A,b,tol,maxit,M1,M2)`
`bicgstab(A,b,tol,maxit,M1,M2,x0)`
`[x,flag]=bicgstab(A,b,...)`
`[x,flag,relres]=bicgstab(A,b,...)`
`[x,flag,relres,iter]=bicgstab(A,b,...)`
`[x,flag,relres,iter,resvec]=bicgstab(A,b,...)`

稳定双共轭梯度法解方程组，调用方式和返回的结果形式和命令 `bicg` 一样。

例 1-84

```
>>load west0479;
>>A=west0479;
>>b=sum(A,2);
>>[x,flag]=bicgstab(A,b)
```

显示结果是 `x` 的值全为 0，`flag=1`。表示在默认误差和默认迭代次数（20 次）下不收敛。

若改为：

```
>>[L1,U1]=luinc(A,1e-5);
>>[x1,flag1]=bicgstab(A,b,1e-6,20,L1,U1)
```

即指定误差，并用 `A` 的不完全 LU 分解因子 `L` 和 `U` 作为预处理因子 `M=L*U`，其结果是 `x1` 的值全为 0，`flag=2` 表示预处理因子为坏条件的预处理因子。

若改为

```
>>[L2,U2]=luinc(A,1e-6); %稀疏矩阵的不完全 LU 分解。
>>[x2,flag2,relres2,iter2,resvec2]=bicgstab(A,b,1e-15,10,L2,U2)
%指定最大迭代次数为 10 次，预处理因子 M=L*U。
>>semilogy(0:0.5:iter2,resvec2/norm(b),'-o') %每次迭代的相对残差图形，见图 1-2。
>>xlabel('iteration number')
>>ylabel('relative residual')
```

结果为

```
x2= (其值全为 1，略)
flag2 =
0 %表示收敛
relres2 =
2.8534e-016 %收敛时的相对误差
iter2 =
6 %计算终止时的迭代次数
resvec2 =
%每次迭代的残差
```

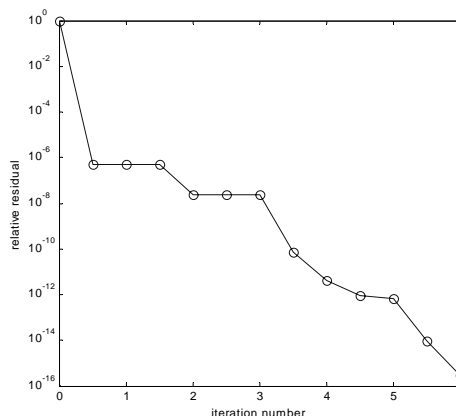


图 1-2 稳定双共轭梯度方法的相对误差图

1.4.7 复共轭梯度平方法解方程组

函数 `cgs`

格式 `x=cgs(A,b)`
`cgs(A,b,tol)`
`cgs(A,b,tol,maxit)`
`cgs(A,b,tol,maxit,M)`
`cgs(A,b,tol,maxit,M1,M2)`
`cgs(A,b,tol,maxit,M1,M2,x0)`
`[x,flag]=cgs(A,b,...)`
`[x,flag,relres]=cgs(A,b,...)`
`[x,flag,relres,iter]=cgs(A,b,...)`



```
[x,flag,relres,iter,resvec] = cgs(A,b,...)
```

调用方式和返回的结果形式与命令 `bicg` 一样。

1.4.8 共轭梯度的 LSQR 方法

函数 **lsqr**

格式 `x = lsqr(A,b)`

```
lsqr(A,b,tol)
```

```
lsqr(A,b,tol,maxit)
```

```
lsqr(A,b,tol,maxit,M)
```

```
lsqr(A,b,tol,maxit,M1,M2)
```

```
lsqr(A,b,tol,maxit,M1,M2,x0)
```

```
[x,flag] = lsqr(A,b,...)
```

```
[x,flag,relres] = lsqr(A,b,...)
```

```
[x,flag,relres,iter] = lsqr(A,b,...)
```

```
[x,flag,relres,iter,resvec] = lsqr(A,b,...)
```

调用方式和返回的结果形式与命令 `bicg` 一样。

例 1-85

```
>>n = 100;
>>on = ones(n,1);
>>A = spdiags([-2*on 4*on -on],[-1:1,n,n]); %产生一个对角矩阵
>>b = sum(A,2);
>>tol = 1e-8; %指定精度
>>maxit = 15; %指定最大迭代次数
>>M1 = spdiags([on/(-2) on],[-1:0,n,n]);
>>M2 = spdiags([4*on -on],0:1,n,n); %M1*M2=M, 即产生预处理因子
>>[x,flag,relres,iter,resvec] = lsqr(A,b,tol,maxit,M1,M2,[])
```

结果显示

`x` 的值全为 1

```
flag =
0 %表示收敛
```

```
relres =
3.5241e-009 %表示相对残差
```

```
iter =
12 %计算终止时的迭代次数
```

1.4.9 广义最小残差法

函数 **gmres**

格式 `x = gmres(A,b)`

```
gmres(A,b,restart)
```

```
gmres(A,b,restart,tol)
```

```
gmres(A,b,restart,tol,maxit)
```

```
gmres(A,b,restart,tol,maxit,M)
```

```
gmres(A,b,restart,tol,maxit,M1,M2)
```

```
gmres(A,b,restart,tol,maxit,M1,M2,x0)
```

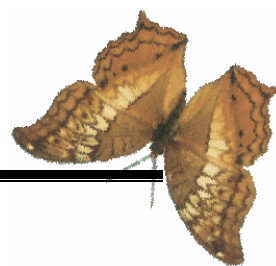
```
[x,flag] = gmres(A,b,...)
```

```
[x,flag,relres] = gmres(A,b,...)
```

```
[x,flag,relres,iter] = gmres(A,b,...)
```

```
[x,flag,relres,iter,resvec] = gmres(A,b,...)
```

除参数 `restart` (缺省时为系数方阵 `A` 的阶数 `n`) 可以给出外, 调用方式和返回的结果形式与命令 `bicg` 一样。

**例 1-86**

```
>>load west0479;
>>A = west0479;
>>b = sum(A,2);
>>[x,flag] = gmres(A,b,5)
```

结果显示 $\text{flag}=1$ ，表示在默认精度和默认迭代次数下不收敛。

若最后一行改为

```
>>[L1,U1] = luinc(A,1e-5);
>>[x1,flag1] = gmres(A,b,5,1e-6,5,L1,U1)
```

结果 $\text{flag1}=2$ ，说明该方法失败，原因是 $U1$ 的对角线上有 0 元素，构成坏条件的预处理因子。

若改为：

```
[L2,U2] = luinc(A,1e-6);
tol = 1e-15;
[x4,flag4,relres4,iter4,resvec4] = gmres(A,b,4,tol,5,L2,U2)
[x6,flag6,relres6,iter6,resvec6] = gmres(A,b,6,tol,3,L2,U2)
[x8,flag8,relres8,iter8,resvec8] = gmres(A,b,8,tol,3,L2,U2)
```

结果 $\text{flag4}=0$ ， $\text{flag6}=0$ ， $\text{flag8}=0$ 表示参数 restart 分别取为 4，6，8 时收敛。

1.4.10 最小残差法解方程组

函数 minres

格式 $x = \text{minres}(A,b)$
 $\text{minres}(A,b,\text{tol})$
 $\text{minres}(A,b,\text{tol},\text{maxit})$
 $\text{minres}(A,b,\text{tol},\text{maxit},M)$
 $\text{minres}(A,b,\text{tol},\text{maxit},M1,M2)$
 $\text{minres}(A,b,\text{tol},\text{maxit},M1,M2,x0)$
 $[x,\text{flag}] = \text{minres}(A,b,\cdots)$
 $[x,\text{flag},\text{relres}] = \text{minres}(A,b,\cdots)$
 $[x,\text{flag},\text{relres},\text{iter}] = \text{minres}(A,b,\cdots)$
 $[x,\text{flag},\text{relres},\text{iter},\text{resvec}] = \text{minres}(A,b,\cdots)$
 $[x,\text{flag},\text{relres},\text{iter},\text{resvec},\text{resvecg}] = \text{minres}(A,b,\cdots)$

这里 A 为对称矩阵，这种方法是寻找最小残差来求 x 。调用方式和返回的结果形式与命令 `bicg` 一样。

例 1-87

```
>>n = 100; on = ones(n,1);
>>A = spdiags([-2*on 4*on -2*on],[-1:1,n,n]);
>>b = sum(A,2);
>>tol = 1e-10;
>>maxit = 50;
>>M1 = spdiags(4*on,0,n,n);
>> [x,flag,relres,iter,resvec,resvecg] = minres(A,b,tol,maxit,M1,[],[])
```

结果显示：

```
flag =
    0
relres =
    4.6537e-014
iter =
    49
```



```
resvec = (略)  
resvecg = (略)
```

1.4.11 预处理共轭梯度方法

函数 **pcg**

格式 $x = \text{pcg}(A,b)$

```
pcg(A,b,tol)  
pcg(A,b,tol,maxit)  
pcg(A,b,tol,maxit,M)  
pcg(A,b,tol,maxit,M1,M2)  
pcg(A,b,tol,maxit,M1,M2,x0)  
pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)  
[x,flag] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)  
[x,flag,relres] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)  
[x,flag,relres,iter] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)  
[x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)
```

调用方式和返回的结果形式与命令 **bicg** 一样，这里 A 为对称正定矩阵。

1.4.12 准最小残差法解方程组

函数 **qmr**

格式 $x = \text{qmr}(A,b)$

```
qmr(A,b,tol)  
qmr(A,b,tol,maxit)  
qmr(A,b,tol,maxit,M)  
qmr(A,b,tol,maxit,M1,M2)  
qmr(A,b,tol,maxit,M1,M2,x0)  
qmr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)  
[x,flag] = qmr(A,b,...)  
[x,flag,relres] = qmr(A,b,...)  
[x,flag,relres,iter] = qmr(A,b,...)  
[x,flag,relres,iter,resvec] = qmr(A,b,...)
```

调用方式和返回的结果形式与命令 **bicg** 一样，这里 A 为方阵。

例 1-88

```
>>load west0479;  
>>A = west0479;  
>>b = sum(A,2);  
>>[x,flag] = qmr(A,b)
```

结果显示 **flag=1**，表示在缺省情况下不收敛

若改为

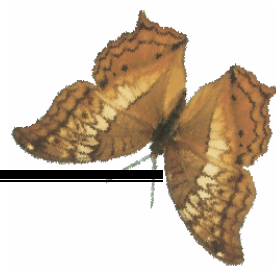
```
>>[L1,U1] = luinc(A,1e-5);  
>>[x1,flag1] = qmr(A,b,1e-6,20,L1,U1)
```

结果显示 **flag=2**，表示是坏条件的预处理因子，不收敛。

若改为

```
>>[L2,U2] = luinc(A,1e-6);  
>>[x2,flag2,relres2,iter2,resvec2] = qmr(A,b,1e-15,10,L2,U2)  
>>semilogy(0:iter2,resvec2/norm(b),'-o') % 每次迭代的相对残差图形  
>>xlabel('iteration number')  
>>ylabel('relative residual')
```

结果为



```

x= (全为 1, 略)
flag2 =
    0      %表示收敛
relres2 =      %表示相对残差
    2.8715e-016
iter2 =      %计算终止时的迭代次数
    8
resvec2 =      %每次迭代的残差
    1.0e+005 *
    7.0557
    7.1773
    3.4032
    1.7220
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000

```

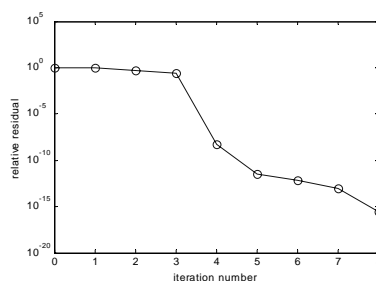


图 1-3 准最小残差法相对误差图

1.5 特征值与二次型

工程技术中的一些问题，如振动问题和稳定性问题，常归结为求一个方阵的特征值和特征向量。

1.5.1 特征值与特征向量的求法

设 A 为 n 阶方阵，如果数“ λ ”和 n 维列向量 x 使得关系式 $Ax = \lambda x$ 成立，则称 λ 为方阵 A 的特征值，非零向量 x 称为 A 对应于特征值“ λ ”的特征向量。

详见 1.3.5 和 1.3.6 节：特征值分解问题。

例 1-89 求矩阵 $A = \begin{pmatrix} -2 & 1 & 1 \\ 0 & 2 & 0 \\ -4 & 1 & 3 \end{pmatrix}$ 的特征值和特征向量

解：

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

```
>>[V,D]=eig(A)
```

结果显示：



```
V =
    -0.7071    -0.2425     0.3015
         0         0     0.9045
    -0.7071    -0.9701     0.3015
```

```
D =
    -1     0     0
         0     2     0
         0     0     2
```

即：特征值-1 对应特征向量 $(-0.7071 \ 0 \ -0.7071)^T$

特征值 2 对应特征向量 $(-0.2425 \ 0 \ -0.9701)^T$ 和 $(-0.3015 \ 0.9045 \ -0.3015)^T$

例 1-90 求矩阵 $A = \begin{pmatrix} -1 & 1 & 0 \\ -4 & 3 & 0 \\ 1 & 0 & 2 \end{pmatrix}$ 的特征值和特征向量。

解：

```
>>A=[-1 1 0;-4 3 0;1 0 2];
>>[V,D]=eig(A)
```

结果显示为

```
V =
         0         0.4082    -0.4082
         0         0.8165    -0.8165
        1.0000    -0.4082     0.4082

D =
     2     0     0
     0     1     0
     0     0     1
```

说明 当特征值为 1 (二重根)时, 对应特征向量都是 $k \ (0.4082 \ 0.8165 \ -0.4082)^T$, k 为任意常数。

1.5.2 提高特征值的计算精度

函数 **balance**

格式 $[T,B] = \text{balance}(A)$ %求相似变换矩阵 T 和平衡矩阵 B , 满足 $B = T^{-1}AT$ 。

$B = \text{balance}(A)$ %求平衡矩阵 B

1.5.3 复对角矩阵转化为实对角矩阵

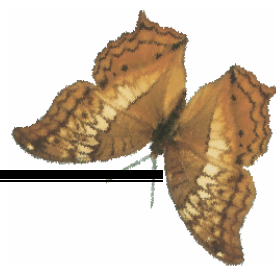
函数 **cdf2rdf**

格式 $[V,D] = \text{cdf2rdf}(v,d)$ %将复对角阵 d 变为实对角阵 D , 在对角线上, 用 2×2 实数块代替共轭复数对。

例 1-91

```
>>A=[1 2 3;0 4 5;0 -5 4];
>>[v,d]=eig(A)
v =
    1.0000         0         0
    -0.0191 - 0.4002i    -0.0191 + 0.4002i
         0    0 - 0.6479i         0 + 0.6479i
         0         0.6479         0.6479

d =
    1.0000         0         0
         0    4.0000 + 5.0000i         0
```



```

0          0          4.0000 - 5.0000i
>> [V,D]=cdf2rdf(v,d)
V =
    1.0000    -0.0191   -0.4002
         0         0   -0.6479
         0     0.6479         0
D =
    1.0000         0         0
         0     4.0000     5.0000
         0    -5.0000     4.0000

```

1.5.4 正交基

命令 `orth`

格式 `B=orth(A)` %将矩阵 A 正交规范化, B 的列与 A 的列具有相同的空间, B 的列向量是正交向量, 且满足: $B'*B = \text{eye}(\text{rank}(A))$ 。

例 1-92 将矩阵 $A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix}$ 正交规范化。

解:

```

>>A=[4 0 0;0 3 1;0 1 3];
>>B=orth(A)
>>Q=B'*B

```

则显示结果为

```

P =
    1.0000         0         0
         0     0.7071   -0.7071
         0     0.7071    0.7071
Q =
    1.0000         0         0
         0     1.0000         0
         0         0     1.0000

```

1.5.5 二次型

例 1-93 求一个正交变换 $X=PY$, 把二次型

$$f = 2x_1x_2 + 2x_1x_3 - 2x_1x_4 - 2x_2x_3 + 2x_2x_4 + 2x_3x_4 \text{ 化成标准形。}$$

解: 先写出二次型的实对称矩阵

$$A = \begin{pmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & -1 & 1 \\ 1 & -1 & 0 & 1 \\ -1 & 1 & 1 & 0 \end{pmatrix}$$

在 Matlab 编辑器中建立 M 文件如下:

```

A=[0 1 1 -1;1 0 -1 1;1 -1 0 1;-1 1 1 0];
[P,D]=schur(A)
syms y1 y2 y3 y4
y=[y1;y2;y3;y4];
X=vpa(P,2)*y %vpa 表示可变精度计算, 这里取 2 位精度
f=[y1 y2 y3 y4]*D*y

```



运行后结果显示如下：

```

P =
    780/989    780/3691    1/2   -390/1351
    780/3691    780/989   -1/2    390/1351
    780/1351   -780/1351   -1/2    390/1351
         0         0        1/2   1170/1351

D =
     1         0         0         0
     0         1         0         0
     0         0        -3         0
     0         0         0         1

X =
    [.79*y1+.21*y2+.50*y3-.29*y4]
    [.21*y1+.79*y2-.50*y3+.29*y4]
    [.56*y1-.56*y2-.50*y3+.29*y4]
    [          .50*y3+.85*y4]

f =
    y1^2+y2^2-3*y3^2+y4^2

```

即 $f = y_1^2 + y_2^2 - 3y_3^2 + y_4^2$

1.6 秩与线性相关性

1.6.1 矩阵和向量组的秩以及向量组的线性相关性

矩阵 A 的秩是矩阵 A 中最高阶非零子式的阶数；向量组的秩通常由该向量组构成的矩阵来计算。

函数 rank

格式 $k = \text{rank}(A)$ %返回矩阵 A 的行（或列）向量中线性无关个数
 $k = \text{rank}(A, \text{tol})$ %tol 为给定误差

例 1-94 求向量组 $\alpha_1 = (1 \ -2 \ 2 \ 3)$, $\alpha_2 = (-2 \ 4 \ -1 \ 3)$, $\alpha_3 = (-1 \ 2 \ 0 \ 3)$, $\alpha_4 = (0 \ 6 \ 2 \ 3)$, $\alpha_5 = (2 \ -6 \ 3 \ 4)$ 的秩, 并判断其线性相关性。

```

>>A=[1 -2 2 3;-2 4 -1 3;-1 2 0 3;0 6 2 3;2 -6 3 4];
>>k=rank(A)

```

结果为

```

k =
    3

```

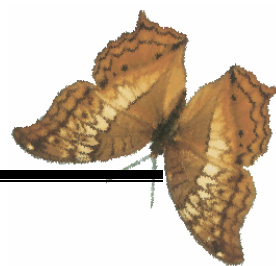
由于秩为 $3 <$ 向量个数, 因此向量组线性相关。

1.6.2 求行阶梯矩阵及向量组的基

行阶梯使用初等行变换, 矩阵的初等行变换有三条:

1. 交换两行 $r_i \leftrightarrow r_j$ (第 i 、第 j 两行交换)
2. 第 i 行的 K 倍 kr_i
3. 第 i 行的 K 倍加到第 j 行上去 $r_j + kr_i$

通过这三条变换可以将矩阵化成行最简形, 从而找出列向量组的一个最大无关组,



Matlab 将矩阵化成行最简形的命令是 `rref` 或 `rrefmovie`。

函数 `rref` 或 `rrefmovie`

格式 `R = rref(A)` %用高斯—约当消元法和行主元法求 A 的行最简行矩阵 R
`[R,jb] = rref(A)` % jb 是一个向量, 其含义为: $r = \text{length}(jb)$ 为 A 的秩; $A(:, jb)$ 为 A 的列向量基; jb 中元素表示基向量所在的列。
`[R,jb] = rref(A,tol)` % tol 为指定的精度
`rrefmovie(A)` %给出每一步化简的过程

例 1-95 求向量组 $a_1=(1,-2,2,3)$, $a_2=(-2,4,-1,3)$, $a_3=(-1,2,0,3)$, $a_4=(0,6,2,3)$, $a_5=(2,-6,3,4)$ 的一个最大无关组。

```
>> a1=[1 -2 2 3]';
>> a2=[-2 4 -1 3]';
>> a3=[-1 2 0 3]';
>> a4=[0 6 2 3]';
>> a5=[2 -6 3 4]';
A=[a1 a2 a3 a4 a5]
A =
     1     -2     -1     0     2
    -2     4      2     6    -6
     2     -1     0     2     3
     3     3      3     3     4
>> [R,jb]=rref(A)
R =
    1.0000         0    0.3333         0    1.7778
         0    1.0000    0.6667         0   -0.1111
         0         0         0    1.0000   -0.3333
         0         0         0         0         0
jb =
     1     2     4
>> A(:,jb)
ans =
     1     -2     0
    -2     4     6
     2     -1     2
     3     3     3
```

即: $a_1 \ a_2 \ a_4$ 为向量组的一个基。

1.7 稀疏矩阵技术

1.7.1 稀疏矩阵的创建

函数 `sparse`

格式 `S = sparse(A)` %将矩阵 A 转化为稀疏矩阵形式, 即由 A 的非零元素和下标构成稀疏矩阵 S 。若 A 本身为稀疏矩阵, 则返回 A 本身。
`S = sparse(m,n)` %生成一个 $m \times n$ 的所有元素都是 0 的稀疏矩阵
`S = sparse(i,j,s)` %生成一个由长度相同的向量 i , j 和 s 定义的稀疏矩阵 S , 其中 i , j 是整数向量, 定义稀疏矩阵的元素位置 (i,j) , s 是一个标



量或与 i, j 长度相同的向量，表示在 (i,j) 位置上的元素。

$S = \text{sparse}(i,j,s,m,n)$ %生成一个 $m \times n$ 的稀疏矩阵， (i,j) 对应位置元素为 s_i ， $m = \max(i)$ 且 $n = \max(j)$ 。

$S = \text{sparse}(i,j,s,m,n,nzmax)$ %生成一个 $m \times n$ 的含有 $nzmax$ 个非零元素的稀疏矩阵 S ， $nzmax$ 的值必须大于或者等于向量 i 和 j 的长度。

例 1-96

```
>> S=sparse(1:10,1:10,1:10)
```

S =

(1,1)	1
(2,2)	2
(3,3)	3
(4,4)	4
(5,5)	5
(6,6)	6
(7,7)	7
(8,8)	8
(9,9)	9
(10,10)	10

```
>> S=sparse(1:10,1:10,5)
```

S =

(1,1)	5
(2,2)	5
(3,3)	5
(4,4)	5
(5,5)	5
(6,6)	5
(7,7)	5
(8,8)	5
(9,9)	5
(10,10)	5

1.7.2 将稀疏矩阵转化为满矩阵

函数 **full**

格式 $A = \text{full}(S)$ % S 为稀疏矩阵， A 为满矩阵。

例 1-97

```
>> S=sparse(1:5,1:5,4:8)
```

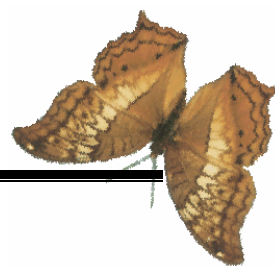
S =

(1,1)	4
(2,2)	5
(3,3)	6
(4,4)	7
(5,5)	8

```
>> A=full(S)
```

A =

4	0	0	0	0
0	5	0	0	0
0	0	6	0	0
0	0	0	7	0
0	0	0	0	8



1.7.3 稀疏矩阵非零元素的索引

函数 **find**

格式 $k = \text{find}(x)$ %按行检索 X 中非零元素的点，若没有非零元素，将返回空矩阵。

$[i,j] = \text{find}(X)$ %检索 X 中非零元素的行标 i 和列标 j

$[i,j,v] = \text{find}(X)$ %检索 X 中非零元素的行标 i 和列标 j 以及对应的元素值 v

例 1-98 上例中

```
>> [i,j,v]=find(S)
```

则显示为：

```
I=[ 1 2 3 4 5]'
```

```
j=[ 1 2 3 4 5]'
```

```
v=[4 5 6 7 8]'
```

1.7.4 外部数据转化为稀疏矩阵

函数 **spconvert**

格式 $S = \text{spconvert}(D)$ % D 是只有 3 列或 4 列的矩阵

说明：先运用 `load` 函数把外部数据（.mat 文件或.dat 文件）装载于 MATLAB 内存空间中的变量 T ； T 数组的行维为 `nnz` 或 `nnz+1`，列维为 3（对实数而言）或列维为 4（对复数而言）； T 数组的每一行（以 $[i,j,Sre,Sim]$ 形式）指定一个稀疏矩阵元素。

例 1-99

```
>> D=[1 2 3;2 5 4;3 4 6;3 6 7]
```

```
D =
```

```
1 2 3
2 5 4
3 4 6
3 6 7
```

```
>> S=spconvert(D)
```

```
S =
```

```
(1,2) 3
(3,4) 6
(2,5) 4
(3,6) 7
```

```
>> D=[1 2 3 4; 2 5 4 0; 3 4 6 9; 3 6 7 4];
```

```
D =
```

```
1 2 3 4
2 5 4 0
3 4 6 9
3 6 7 4
```

```
>> S=spconvert(D)
```

```
S =
```

```
(1,2) 3.0000 + 4.0000i
(3,4) 6.0000 + 9.0000i
(2,5) 4.0000
(3,6) 7.0000 + 4.0000i
```

1.7.5 基本稀疏矩阵

1. 带状（对角）稀疏矩阵

函数 **spdiags**



格式 `[B,d] = spdiags(A)` %从矩阵 **A** 中提取所有非零对角元素, 这些元素保存在矩阵 **B** 中, 向量 **d** 表示非零元素的对角线位置。

`B = spdiags(A,d)` %从 **A** 中提取由 **d** 指定的对角线元素, 并存放在 **B** 中。

`A = spdiags(B,d,A)` %用 **B** 中的列替换 **A** 中由 **d** 指定的对角线元素, 输出稀疏矩阵。

`A = spdiags(B,d,m,n)` %产生一个 $m \times n$ 稀疏矩阵 **A**, 其元素是 **B** 中的列元素放在由 **d** 指定的对角线位置上。

例 1-100

```
>>A=[11 0 13 0
      0 22 0 24
      0 0 33 0
      41 0 0 44
      0 52 0 0
      0 0 63 0
      0 0 0 74];
>>[B,d]=spdiags(A)
B =
    41    11     0
    52    22     0
    63    33    13
    74    44    24
d =
   -3      %表示 B 的第 1 列元素在 A 中主对角线下方第 3 条对角线上
    0      %表示 B 的第 2 列在 A 的主对角线上
    2      %表示 B 的第 3 列在 A 的主对角线上方第 2 条对角线上
```

例 1-101

```
>> B=[1 2 3 4
      5 6 7 8
      9 10 11 12
      13 14 15 16];
>> d=[-2 0 1 3];
>> A=spdiags(B,d,4,4);
>> full(A)
ans =
     2     7     0    16
     0     6    11     0
     1     0    10    15
     0     5     0    14
```

2. 单位稀疏矩阵**函数 speye**

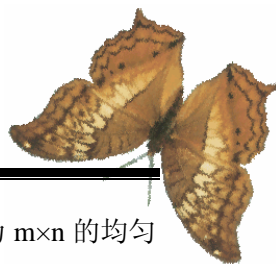
格式 `S = speye(m,n)` %生成 $m \times n$ 的单位稀疏矩阵

`S = speye(n)` %生成 $n \times n$ 的单位稀疏矩阵

3. 稀疏均匀分布随机矩阵**函数 sprand**

格式 `R = sprand(S)` %生成与 **S** 具有相同稀疏结构的均匀分布随机矩阵

`R = sprand(m,n,density)` %生成一个 $m \times n$ 的服从均匀分布的随机稀疏矩阵, 非零元素的分布密度是 **density**。



`R = sprand(m,n,density,rc)` %生成一个近似的条件数为 $1/rc$ 、大小为 $m \times n$ 的均匀分布的随机稀疏矩阵。

4. 稀疏正态分布随机矩阵

函数 **sprandn**

格式 `R = sprandn(S)` %生成与 S 具有相同稀疏结构的正态分布随机矩阵。

`R = sprandn(m,n,density)` %生成一个 $m \times n$ 的服从正态分布的随机稀疏矩阵，非零元素的分布密度是 `density`。

`R = sprandn(m,n,density,rc)` %生成一个近似的条件数为 $1/rc$ 、大小为 $m \times n$ 的均匀分布的随机稀疏矩阵。

5. 稀疏对称随机矩阵

函数 **sprandsym**

格式 `R = sprandsym(S)` %生成稀疏对称随机矩阵，其下三角和对角线与 S 具有相同的结构，其元素服从均值为 0、方差为 1 的标准正态分布。

`R = sprandsym(n,density)` %生成 $n \times n$ 的稀疏对称随机矩阵，矩阵元素服从正态分布，分布密度为 `density`。

`R = sprandsym(n,density,rc)` %生成近似条件数为 $1/rc$ 的稀疏对称随机矩阵

`R = sprandsym(n,density,rc,kind)` %生成一个正定矩阵，参数 `kind` 取值为 `kind=1` 表示矩阵由一正定对角矩阵经随机 Jacobi 旋转得到，其条件数正好为 $1/rc$ ；`kind=2` 表示矩阵为外积的换位和，其条件数近似等于 $1/rc$ ；`kind=3` 表示生成一个与矩阵 S 结构相同的稀疏随机矩阵，条件数近似为 $1/rc$ ，`density` 被忽略。

1.7.6 稀疏矩阵的运算

1. 稀疏矩阵非零元素的个数

函数 **nnz**

格式 `n = nnz(X)` %返回矩阵 X 中非零元素的个数

2. 稀疏矩阵的非零元素

函数 **nonzeros**

格式 `s = nonzeros(A)` %返回矩阵 A 中非零元素按列顺序构成的列向量

例 1-102

```
>>A=[ 2    7    0   16
      0    6   11    0
      1    0   10   15
      0    5    0   14];
```

```
>> s=nonzeros(A)
```

结果为：

```
s=[ 2  1  7  6  5 11 10 16 15 14]'
```

3. 稀疏矩阵非零元素的内存分配

函数 **nzmax**



格式 $n = \text{nzmax}(S)$ %返回非零元素分配的内存总数 n

4. 稀疏矩阵的存储空间

函数 **spalloc**

格式 $S = \text{spalloc}(m,n,\text{nzmax})$ %产生一个 $m \times n$ 阶只有 nzmax 个非零元素的稀疏矩阵，这样可以有效减少存储空间和提高运算速度。

5. 稀疏矩阵的非零元素应用

函数 **spfun**

格式 $f = \text{spfun}(\text{'function'}, S)$ %用 S 中非零元素对函数 'function' 求值，如果 'function' 不是对稀疏矩阵定义的，同样可以求值。

例 1-103 4 阶稀疏矩阵对角矩阵

```
S =
(1,1)      1
(2,2)      2
(3,3)      3
(4,4)      4
f = spfun('exp',S) %即指数 e 的非零元素方幂。
```

结果为

```
f =
(1,1)      2.7183
(2,2)      7.3891
(3,3)     20.0855
(4,4)     54.5982
```

6. 把稀疏矩阵的非零元素全换为 1

函数 **spones**

格式 $R = \text{spones}(S)$ %将稀疏矩阵 S 中的非零元素全换为 1

1.7.7 画稀疏矩阵非零元素的分布图形

函数 **spy**

格式 $\text{spy}(S)$ %画出稀疏矩阵 S 中非零元素的分布图形。 S 也可以是满矩阵。

$\text{spy}(S, \text{markersize})$ % markersize 为整数，指定点阵大小。

$\text{spy}(S, \text{'LineSpec'})$ % 'LineSpec' 指定绘图标记和颜色

$\text{spy}(S, \text{'LineSpec'}, \text{markersize})$ %参数与上面相同

例 1-104

```
>> load west0479
>> A=west0479;
>> spy(A,'ro',3)
```

结果如图 1-4 所示。

1.7.8 矩阵变换

1. 列近似最小度排序

函数 **colamd**

格式 $p = \text{colamd}(S)$ %返回稀疏矩阵 S 的列的

近似最小度排序向量 p

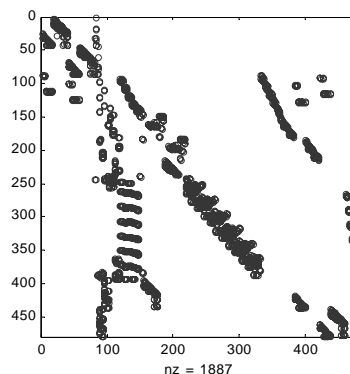
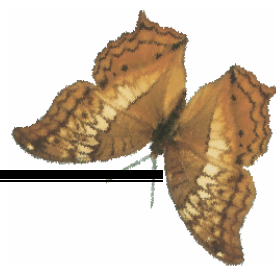


图 1-4 稀疏矩阵非零元素的分布图形



2. 列最小度排序

函数 **colmmd**

格式 $p = \text{colmmd}(S)$ %返回稀疏矩阵 S 的列的最小度排序向量 p , 按 p 排列后的矩阵为 $S(:, p)$ 。

例 1-105 比较稀疏矩阵 S 与排序后的矩阵 $S(:, p)$

```
>> load west0479;
>> S=west0479;
>> p=colmmd(S);
>> subplot(2,2,1),spy(S)
>> subplot(2,2,2),spy(S(:,p))
>> subplot(2,2,3),spy(lu(S))
>> subplot(2,2,4),spy(lu(S(:,p)))
```

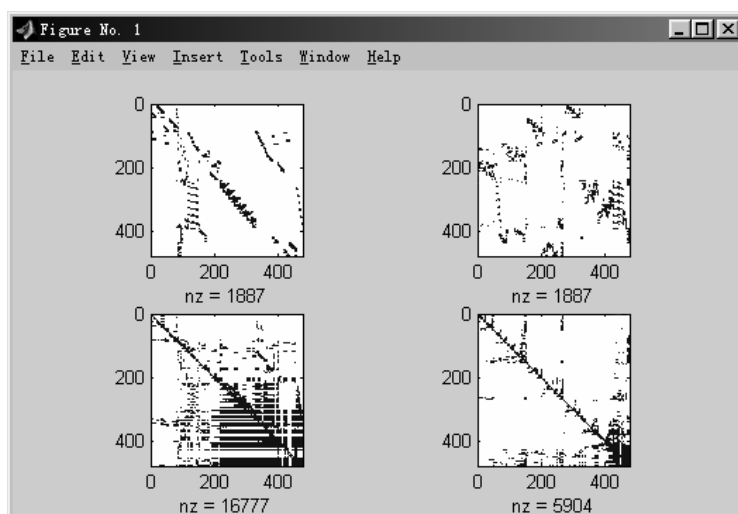


图 1-5 稀疏矩阵的排序图

3. 非零元素的列变换

函数 **colperm**

格式 $j = \text{colperm}(S)$ %返回一个稀疏矩阵 S 的列变换的向量。列按非 0 元素升序排列。有时这是 LU 分解前有用的变换: $\text{LU}(S(:, j))$ 。如果 S 是一个对称矩阵, 对行和列进行排序, 有利于 Cholesky 分解: $\text{chol}(S(j, j))$

4. Dulmage-Mendelsohn 分解

函数 **dmperm**

格式 $p = \text{dmperm}(A)$ %返回 A 的行排列向量 p , 这样, 如果 A 满列秩, 就使得 $A(p, :)$ 是具有非 0 对角线元素的方阵。

$[p, q, r] = \text{dmperm}(A)$ % A 为方阵, p 为行排列向量, q 为列排列向量, 使得 $A(p, q)$ 是上三角块形式, r 为索引向量。

$[p, q, r, s] = \text{dmperm}(A)$ % A 不是方阵, p, q, r 含义与前面相同, s 也是索引向量。



例 1-106

```
>> A=[11 0 13 0;41 0 0 44;0 22 0 24;0 0 63 0]
A =
    11     0    13     0
    41     0     0    44
     0    22     0    24
     0     0    63     0
>> [p,q,r]=dmperm(A)
p =
     3     2     1     4
q =
     2     4     1     3
r =
     1     2     3     4     5
>> A(p,q)
ans =
    22    24     0     0
     0    44    41     0
     0     0    11    13
     0     0     0    63
```

5. 整数的随机排列

函数 **randperm**

格式 `p = randperm(n)` %对正整数 1, 2, 3, ..., n 的随机排列, 可以用来创建随机变换矩阵。

例 1-107

```
>> p=randperm(6)
p =
     3     4     6     5     1     2
```

6. 稀疏对称近似最小度排列

函数 **symamd**

格式 `p = symamd(S)` %S 为对称正定矩阵, 返回排列向量 p。

7. 稀疏逆 Cuthill-McKee 排序

函数 **symrcm**

格式 `r = symrcm(S)` %返回 S 的对称逆 Cuthill-McKee 排序 r, 使 S 的非 0 元素集中在主对角线附近。

8. 稀疏对称最小度排列

函数 **symmmd**

格式 `p = symmmd(S)` %返回 S 的对称最小度排列向量 p, S 为对称正定矩阵。

例 1-108

```
>> B = bucky+4*speye(60);
>> r = symrcm(B);
>> p = symmmd(B);
>> R = B(r,r);
>> S = B(p,p);
>> subplot(2,2,1), spy(R), title('B(r,r)')
>> subplot(2,2,2), spy(S), title('B(s,s)')
>> subplot(2,2,3), spy(chol(R)), title('chol(B(r,r))')
>> subplot(2,2,4), spy(chol(S)), title('chol(B(s,s))')
```

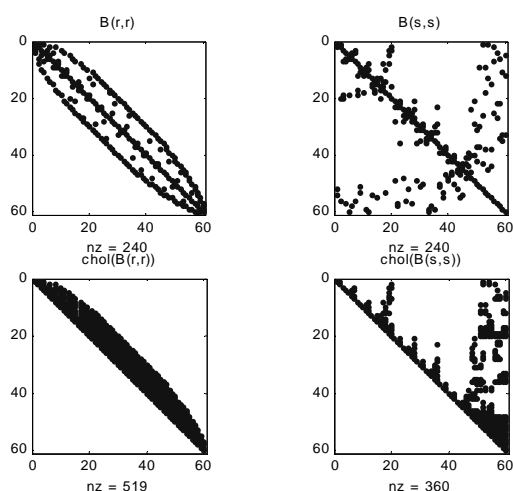
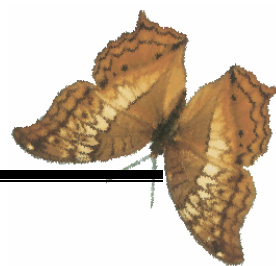


图 1-6 稀疏对称最小度排列图

1.7.9 稀疏矩阵的近似欧几里得范数和条件数

命令 矩阵 A 条件数的 1-范数估计值

函数 **condest**

格式 `c = condest(A)` % 计算方阵 A 的 1-范数中条件数的下界值 c
`[c,v] = condest(A)` % 方阵 A 的 1-范数中条件数的下界值 c 和向量 v, 使得 $\|Av\| = (\|A\|_1 \cdot \|v\|) / c$, 即:
 $\text{norm}(A*v,1) = \text{norm}(A,1)*\text{norm}(v,1)/c$.

命令 2-范数估计值

函数 **normest**

格式 `nrm = normest(S)` % 返回矩阵 S 的 2-范数的估计值, 相对误差为 10^{-6} 。
`nrm = normest(S,tol)` % tol 为指定的相对误差, 而不用默认误差 10^{-6} 。
`[nrm,count] = normest(...)` % count 为给出的计算范数迭代的次数

其条件数的计算与前面 1.2.12 相同。

1.7.10 稀疏矩阵的分解

命令 不完全的 LU 分解

函数 **luinc**

格式 `[L,U] = luinc(X,'0')` % X 为稀疏方阵; L 为下三角矩阵的置换形式; U 为上三角矩阵; '0' 是一种分解标准。
`[L,U,P] = luinc(X,'0')` % L 为下三角阵, 其主对角线元素为 1; U 为上三角阵, p 为单位矩阵的置换形式。
`[L,U] = luinc(X,options)` % options 取值为: droptol 表示指定的舍入误差; milu 表示改变分解以便于从上三角分解因子中抽取被去



掉的列元素。ugiag 为 1 表示用 droptol 值代替上三角因子中的对角线上的零元素，默认值为 0。thresh 为中心临界值。

[L,U] = luinc(X,droptol) %droptol 表示指定不完全分解的舍入误差
 [L,U,P] = luinc(X,options)
 [L,U,P] = luinc(X,droptol)

例 1-109

```
>> S=[11 0 13 0;41 0 0 44;0 22 0 24;0 0 63 0]
```

```
S =
    11     0    13     0
    41     0     0    44
     0    22     0    24
     0     0    63     0
```

```
>> S=sparse(S)
```

```
S =
(1,1)      11
(2,1)      41
(3,2)      22
(1,3)      13
(4,3)      63
(2,4)      44
(3,4)      24
```

```
>> luinc(S,'0')
```

```
ans =
(1,1)      41.0000
(4,1)       0.2683
(2,2)      22.0000
(3,3)      63.0000
(4,3)       0.2063
(1,4)      44.0000
(2,4)      24.0000
```

```
>> [L,U,p]=luinc(S,'0')
```

```
L =
(1,1)      1.0000
(4,1)       0.2683
(2,2)      1.0000
(3,3)      1.0000
(4,3)       0.2063
(4,4)      1.0000
```

```
U =
(1,1)      41
(2,2)      22
(3,3)      63
(1,4)      44
(2,4)      24
```

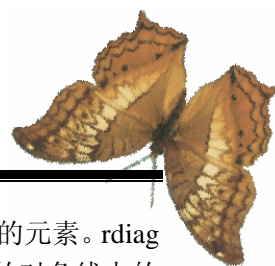
```
p =
(4,1)      1
(1,2)      1
(2,3)      1
(3,4)      1
```

命令 稀疏矩阵的不完全 Cholesky 分解

函数 cholinc

格式 R = (X,droptol) %稀疏矩阵 X 的不完全 Cholesky 分解,droptol 为指定误差。

R = cholinc(X,options) %options 取值: droptol 表示舍入误差; michol 表示如



果 `ichol=1`，则从对角线上抽取出被去掉的元素。`rdiag` 表示用 `droptol` 值代替上三角分解因子中的对角线上的零元素。

`R = cholinc(X,'0')` %'0'是一种分解标准
`[R,p] = cholinc(X,'0')` %不产生任何出错信息，如果 `R` 存在，则 `p=0`；如果 `R` 不存在，则 `p` 为正整数。
`R = cholinc(X,'inf')` %进行 Cholesky 无穷分解

1.7.11 稀疏矩阵的特征值分解

函数 **eigs**

格式 `d = eigs(A)` %求稀疏矩阵 `A` 的 6 个最大特征值 `d`，`d` 以向量形式存放。
`d = eigs(A,B)` %求稀疏矩阵的广义特征值问题。满足 $AV=BVD$ ，其中 `D` 为特征值对角阵，`V` 为特征向量矩阵，`B` 必须是对称正定阵或 Hermitian 正定阵。
`d = eigs(A,k)` %返回 `k` 个最大特征值
`d = eigs(A,B,k)` %返回 `k` 个最大特征值
`d = eigs(A,k,sigma)` %`sigma` 取值：'lm' 表示最大数量的特征值；'sm' 最小数量特征值；对实对称问题：'la'表示最大特征值；'sa'为最小特征值；对非对称和复数问题：'lr' 表示最大实部；'sr' 表示最小实部；'li' 表示最大虚部；'si'表示最小虚部。
`d = eigs(A,B,k,sigma)` %同上
`d = eigs(A,k,sigma,options)` % `options` 为指定参数：参见 `eigs` 帮助文件。
`d = eigs(A,B,k,sigma,options)` %同上。以下的参数 `k`、`sigma`、`options` 相同。
`d = eigs(Afun,n)` %用函数 `Afun` 代替 `A`，`n` 为 `A` 的阶数，`D` 为特征值。
`d = eigs(Afun,n,B)`
`d = eigs(Afun,n,k)`
`d = eigs(Afun,n,B,k)`
`d = eigs(Afun,n,k,sigma)`
`d = eigs(Afun,n,B,k,sigma)`
`d = eigs(Afun,n,k,sigma,options)`
`d = eigs(Afun,n,B,k,sigma,options)`
`[V,D] = eigs(A,...)` %`D` 为 6 个最大特征值对角阵，`V` 的列向量为对应特征向量。
`[V,D] = eigs(Afun,n,...)`
`[V,D,flag] = eigs(A,...)` %`flag` 表示特征值的收敛性，若 `flag=0`，则所有特征值都收敛，否则，不是所有都收敛。
`[V,D,flag] = eigs(Afun,n,...)`

1.7.12 稀疏矩阵的线性方程组

参见 1.4 节的方程组求解。



第2章 数值计算与数据分析

2.1 基本数学函数

2.1.1 三角函数与双曲函数

函数 **sin、sinh**

功能 正弦函数与双曲正弦函数

格式 **Y = sin(X)** % 计算参量 X (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的正弦值 Y, 所有分量的角度单位为弧度。

Y = sinh(X) % 计算参量 X 的双曲正弦值 Y

注意: $\sin(\pi)$ 并不是零, 而是与浮点精度有关的无穷小量 eps , 因为 π 仅仅是精确值 π 浮点近似的表示值而已; 对于复数 $Z = x + iy$, 函数的定义为: $\sin(x + iy) = \sin(x)\cos(y) + i\cos(x)\sin(y)$, $\sin(z) = \frac{e^{iz} - e^{-iz}}{2}$, $\sin(z) = \frac{e^z - e^{-z}}{2}$

例 2-1

`x = -pi:0.01:pi; plot(x, sin(x))`

`x = -5:0.01:5; plot(x, sinh(x))`

图形结果为图 2-1。

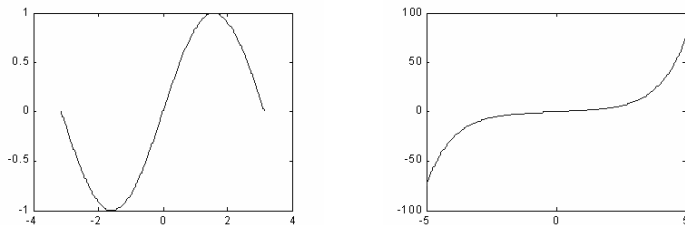


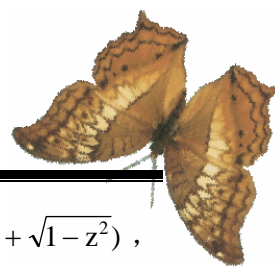
图 2-1 正弦函数与双曲正弦函数图

函数 **asin、asinh**

功能 反正弦函数与反双曲正弦函数

格式 **Y = asin(X)** % 返回参量 X (可以是向量、矩阵) 中每一个元素的反正弦函数值 Y。若 X 中有的分量处于 $[-1, 1]$ 之间, 则 $Y = \text{asin}(X)$ 对应的分量处于 $[-\pi/2, \pi/2]$ 之间, 若 X 中有分量在区间 $[-1, 1]$ 之外, 则 $Y = \text{asin}(X)$ 对应的分量为复数。

Y = asinh(X) % 返回参量 X 中每一个元素的反双曲正弦函数值 Y



说明 反正弦函数与反双曲正弦函数的定义为： $\text{asin } z = -i \cdot \ln(i \cdot z + \sqrt{1 - z^2})$ ， $\text{asinh } z = \ln(z + \sqrt{1 + z^2})$

例 2-2

```
x = -1:0.01:1; plot(x,asin(x))
x = -5:0.01:5; plot(x,asinh(x))
```

图形结果为图 2-2。

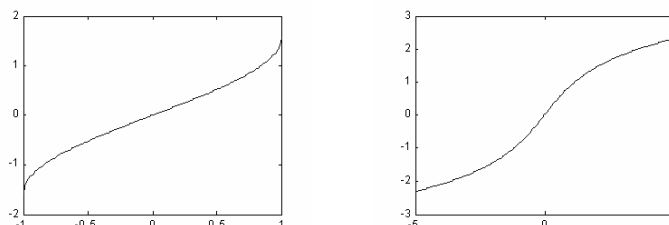


图 2-2 反正弦函数与反双曲正弦函数图

函数 cos、cosh

功能 余弦函数与双曲余弦函数

格式 $Y = \cos(X)$ % 计算参量 X (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的余弦值 Y , 所有角度分量的单位为弧度。我们要指出的是, $\cos(\pi/2)$ 并不是精确的零, 而是与浮点精度有关的无穷小量 eps , 因为 π 仅仅是精确值 π 浮点近似的表示值而已。

$Y = \sinh(X)$ % 计算参量 X 的双曲余弦值 Y

说明 若 X 为复数 $z = x + iy$, 则函数定义为: $\cos(x + iy) = \cos(x) \cdot \cos(y) + i \cdot \sin(x) \cdot \sin(y)$,

$$\cos z = \frac{e^{iz} + e^{-iz}}{2}, \quad \cosh z = \frac{e^z + e^{-z}}{2}$$

例 2-3

```
x = -pi:0.01:pi; plot(x,cos(x))
x = -5:0.01:5; plot(x,cosh(x))
```

图形结果为图 2-3。

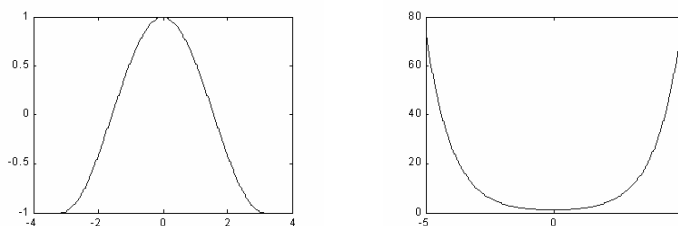


图 2-3 余弦函数与双曲余弦函数图

函数 acos、acosh

功能 反余弦函数与反双曲余弦函数



格式 $Y = \text{acos}(X)$ %返回参量 X (可以是向量、矩阵) 中每一个元素的反余弦函数值 Y 。若 X 中有的分量处于 $[-1,1]$ 之间, 则 $Y = \text{acos}(X)$ 对应的分量处于 $[0, \pi]$ 之间, 若 X 中有分量在区间 $[-1,1]$ 之外, 则 $Y = \text{acos}(X)$ 对应的分量为复数。

$Y = \text{asinh}(X)$ %返回参量 X 中每一个元素的反双曲余弦函数 Y

说明 反余弦函数与反双曲余弦函数定义为: $\text{acos } z = -i \cdot \ln(i \cdot z + i \cdot \sqrt{1-z^2})$,
 $\text{acosh } z = \ln(z + \sqrt{z^2 - 1})$

例 2-4

```
x = -1:0.01:1; plot(x,acos(x))
x = -5:0.01:5; plot(x,acosh(x))
```

图形结果为图 2-4。

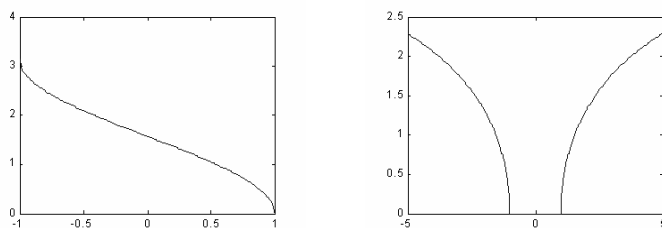


图 2-4 反余弦函数与反双曲余弦函数图

函数 \tan 、 \tanh

功能 正切函数与双曲正切函数

格式 $Y = \tan(X)$ %计算参量 X (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的正切值 Y , 所有角度分量的单位为弧度。我们要指出的是, $\tan(\pi/2)$ 并不是精确的零, 而是与浮点精度有关的无穷小量 eps , 因为 π 仅仅是精确值 π 浮点近似的表示值而已。

$Y = \tanh(X)$ %返回参量 X 中每一个元素的双曲正切函数值 Y

例 2-5

```
x = (-pi/2)+0.01:0.01:(pi/2)-0.01; % 稍微缩小定义域
plot(x,tan(x))
x = -5:0.01:5; plot(x,tanh(x))
```

图形结果为图 2-5。

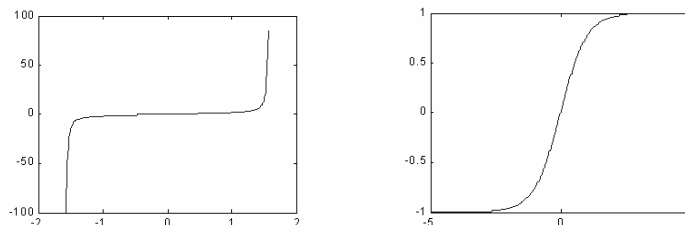
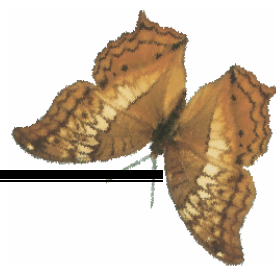


图 2-5 正切函数与双曲正切函数图

**函数 atan、atanh****功能** 反正切函数与反双曲正切函数

格式 $Y = \text{atan}(X)$ %返回参量 X (可以是向量、矩阵) 中每一个元素的反正切函数值 Y 。若 X 中有的分量为实数, 则 $Y = \text{atan}(X)$ 对应的分量处于 $[-\pi/2, \pi/2]$ 之间。

$Y = \text{atanh}(X)$ %返回参量 X 中每一个元素的反双曲正切函数值 Y 。

说明 反正切函数与反双曲正切函数定义为: $a \tan z = \frac{i}{2} \ln \frac{i+z}{i-z}$, $a \tanh z = \frac{1}{2} \ln \frac{1+z}{1-z}$

例 2-6

```
x = -20:0.01:20; plot(x,atan(x))
x = -0.99:0.01:0.99; plot(x,atanh(x))
```

图形结果为图 2-6。

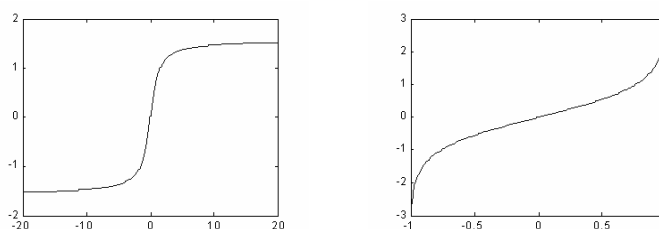


图 2-6 反正切函数与反双曲正切函数图

函数 cot、coth**功能** 余切函数与双曲余切函数

格式 $Y = \text{cot}(X)$ %计算参量 X (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的余切值 Y , 所有角度分量的单位为弧度。

$Y = \text{coth}(X)$ %返回参量 X 中每一个元素的双曲余切函数值 Y

例 2-7

```
x1 = -pi+0.01:0.01:-0.01; % 去掉奇点 x = 0
x2 = 0.01:0.01:pi-0.01; % 做法同上
plot(x1,cot(x1),x2,cot(x2))
plot(x1,coth(x1),x2,coth(x2))
```

图形结果为图 2-7。

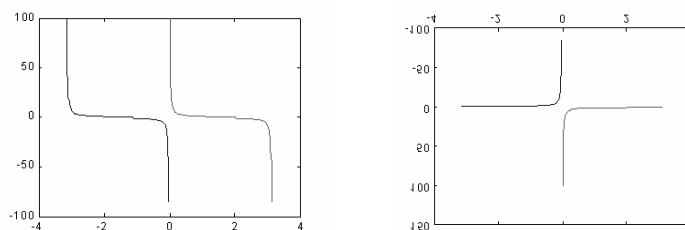


图 2-7 余切函数与双曲余切函数图

**函数** **acot、acoth****功能** 反余切函数与反双曲余切函数**格式** $Y = \text{acot}(X)$ %返回参量 X (可以是向量、矩阵) 中每一个元素的反余切函数 Y $Y = \text{acoth}(X)$ %返回参量 X 中每一个元素的反双曲余切函数值 Y **例 2-8**

```
x1 = -2*pi/pi/30:-0.1; x2 = 0.1:pi/30:2*pi; % 去掉奇异点 x = 0
plot(x1,acot(x1),x2,acot(x2))
x1 = -30:0.1:-1.1; x2 = 1.1:0.1:30;
plot(x1,acoth(x1),x2,acoth(x2))
```

图形结果为图 2-8。

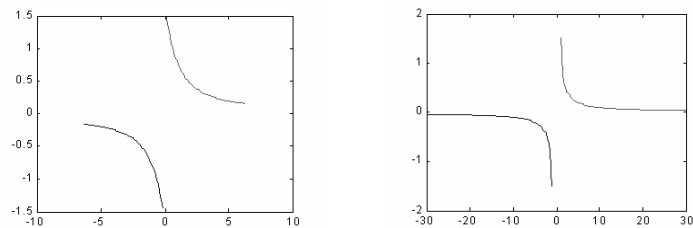


图 2-8 反余切函数与反双曲余切函数图

函数 **sec、sech****功能** 正割函数与双曲正割函数**格式** $Y = \text{sec}(X)$ %计算参量 X (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的正割函数值 Y , 所有角度分量的单位为弧度。我们要指出的是, $\text{sec}(\pi/2)$ 并不是无穷大, 而是与浮点精度有关的无穷小量 eps 的倒数, 因为 π 仅仅是精确值 π 浮点近似的表示值而已。 $Y = \text{sech}(X)$ %返回参量 X 中每一个元素的双曲正割函数值 Y **例 2-9**

```
x1 = -pi/2+0.01:0.01:pi/2-0.01; % 去掉奇异点 x = pi/2
x2 = pi/2+0.01:0.01:(3*pi/2)-0.01;
plot(x1,sec(x1),x2,sec(x2))
x = -2*pi:0.01:2*pi;
plot(x,sech(x))
```

图形结果为图 2-9。

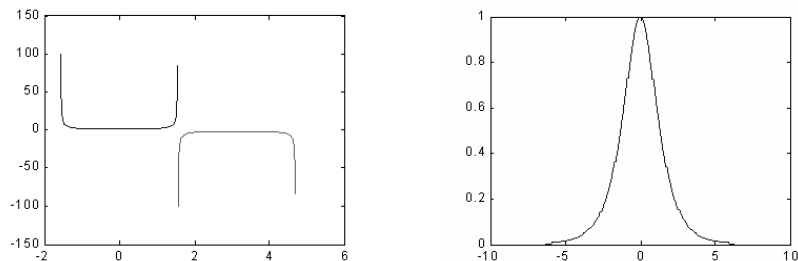
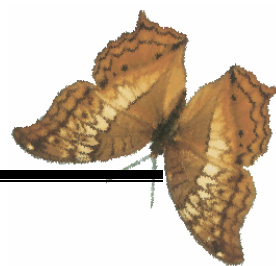


图 2-9 正割函数与双曲正割函数图



函数 `asec`、`asech`

功能 反正割函数与反双曲正割函数

格式 `Y = asec(X)` %返回参量 `X` (可以是向量、矩阵) 中每一个元素的反正割函数值
`Y`

`Y = asech(X)` %返回参量 `X` 中每一个元素的反双曲正割函数值 `Y`

例 2-10

```
x1 = -5:0.01:-1; x2 = 1:0.01:5;
plot(x1,asec(x1),x2,asec(x2))
x = 0.01:0.001:1; plot(x,asech(x))
```

图形结果为图 2-10。

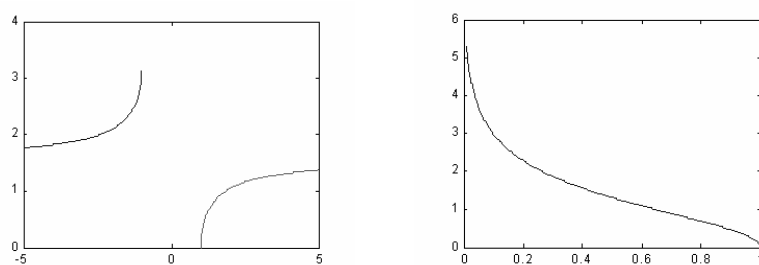


图 2-10 反正割函数与反双曲正割函数图

函数 `csc`、`csch`

功能 余割函数与双曲余割函数

格式 `Y = csc(X)` %计算参量 `X` (可以是向量、矩阵, 元素可以是复数) 中每一个角度分量的余割函数值 `Y`, 所有角度分量的单位为弧度。

`Y = csch(X)` %返回参量 `X` 中每一个元素的双曲余割函数值 `Y`

例 2-11

```
x1 = -pi+0.01:0.01:-0.01; x2 = 0.01:0.01:pi-0.01; % 去掉奇异点 x=0
plot(x1,csc(x1),x2,csc(x2))
plot(x1,csch(x1),x2,csch(x2))
```

图形结果为图 2-11。

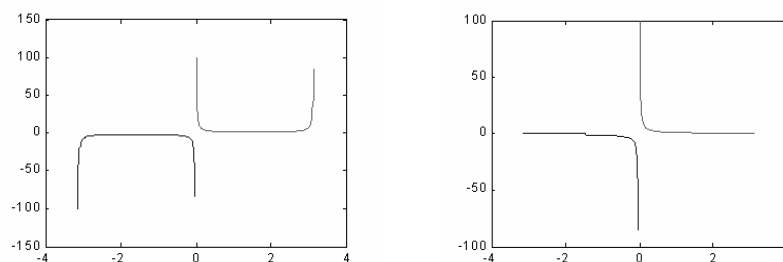


图 2-11 余割函数与双曲余割函数图

函数 `acsc`、`acsch`



功能 反余割函数与反双曲余割函数。

格式 $Y = \operatorname{asec}(X)$ %返回参量 X (可以是向量、矩阵) 中每一个元素的反余割函数值 Y

$Y = \operatorname{asech}(X)$ %返回参量 X 中每一个元素的反双曲余割函数值 Y

例 2-12

```
x1 = -10:0.01:-1.01; x2 = 1.01:0.01:10; % 去掉奇异点 x = 1
plot(x1,acsc(x1),x2,acsc(x2))
x1 = -20:0.01:-1; x2 = 1:0.01:20;
plot(x1,acsch(x1),x2,acsch(x2))
```

图形结果为图 2-12。

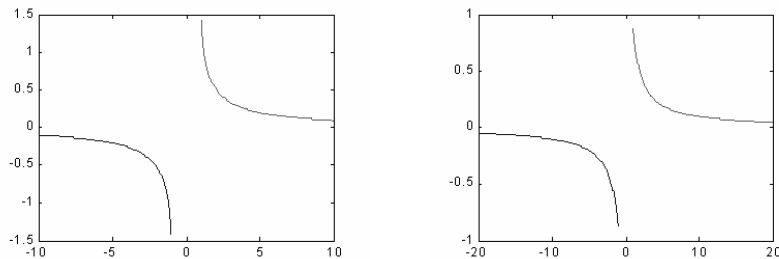


图 2-12 反余割函数与反双曲余割函数图

函数 atan2

功能 四象限的反正切函数

格式 $P = \operatorname{atan2}(Y,X)$ %返回一与参量 X 和 Y 同型的、与 X 和 Y 元素的实数部分对应的、元素对元素的四象限的反正切函数阵列 P , 其中 X 和 Y 的虚数部分将忽略。阵列 P 中的元素分布在闭区间 $[-\pi, \pi]$ 上。特定的象限将取决于 $\operatorname{sign}(Y)$ 与 $\operatorname{sign}(X)$ 。

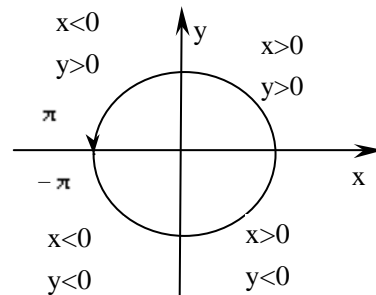
例 2-13

```
z=1+2i;
r = abs(z);
theta = atan2(imag(z),real(z))
z = r * exp(i * theta)
feather(z);hold on
t=0:0.1:2*pi;
x=1+sqrt(5)*cos(t);
y=sqrt(5)*sin(t);
plot(x,y);
axis equal; hold off
```

计算结果为:

```
theta =
    1.1071
z =
    1.0000 + 2.0000i
```

图形结果为图 2-13。



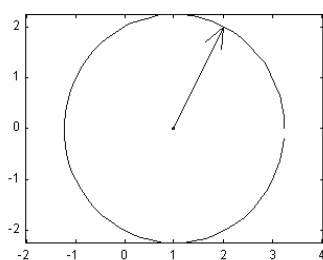
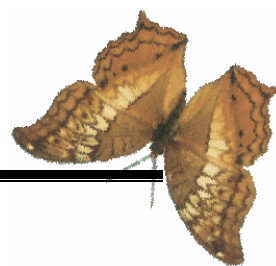


图 2-13 四象限的反正切函数图

2.1.2 其他常用函数

函数 **fix**

功能 朝零方向取整

格式 $B = \text{fix}(A)$ %对 A 的每一个元素朝零的方向取整数部分, 返回与 A 同维的数组。
对于复数参量 A , 则返回一复数, 其分量的实数与虚数部分分别取原复数的、朝零方向的整数部分。

例 2-14

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
```

```
>>B = fix(A)
```

计算结果为:

B =

Columns 1 through 4

-1.0000

0

3.0000

5.0000

Columns 5 through 6

7.0000

2.0000 + 3.0000i

函数 **round**

功能 朝最近的方向取整。

格式 $Y = \text{round}(X)$ %对 X 的每一个元素朝最近的方向取整数部分, 返回与 X 同维的数组。对于复数参量 X , 则返回一复数, 其分量的实数与虚数部分分别取原复数的、朝最近方向的整数部分。

例 2-15

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
```

```
>>Y = round(A)
```

计算结果为:

Y =

Columns 1 through 4

-2.0000

0

3.0000

6.0000

Columns 5 through 6

7.0000

2.0000 + 4.0000i

函数 **floor**

功能 朝负无穷大方向取整

格式 $B = \text{floor}(A)$ %对 A 的每一个元素朝负无穷大的方向取整数部分, 返回与 A 同维的数组。对于复数参量 A , 则返回一复数, 其分量的实数与虚



数部分分别取原复数的、朝负无穷大方向的整数部分。

例 2-16

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>F = floor(A)
```

计算结果为:

```
F =
Columns 1 through 4
-2.0000    -1.0000     3.0000     5.0000
Columns 5 through 6
 7.0000    2.0000 + 3.0000i
```

函数 **rem**

功能 求作除法后的剩余数

格式 **R = rem(X,Y)** % 返回结果 $X - \text{fix}(X./Y) \cdot Y$, 其中 X 、 Y 应为正数。若 X 、 Y 为浮点数, 由于计算机对浮点数的表示的不精确性, 则结果将可能是不可意料的。 $\text{fix}(X./Y)$ 为商数 $X./Y$ 朝零方向取的整数部分。若 X 与 Y 为同符号的, 则 $\text{rem}(X,Y)$ 返回的结果与 $\text{mod}(X,Y)$ 相同, 不然, 若 X 为正数, 则 $\text{rem}(-X,Y) = \text{mod}(-X,Y) - Y$ 。该命令返回的结果在区间 $[0, \text{sign}(X) \cdot \text{abs}(Y)]$, 若 Y 中有零分量, 则相应地返回 NaN。

例 2-17

```
>>X = [12 23 34 45];
>>Y = [3 7 2 6];
>>R = rem(X,Y)
计算结果为:
R =
     0     2     0     3
```

函数 **ceil**

功能 朝正无穷大方向取整

格式 **B = floor(A)** % 对 A 的每一个元素朝正无穷大的方向取整数部分, 返回与 A 同维的数组。对于复数参量 A , 则返回一复数, 其分量的实数与虚数部分分别取原复数的、朝正无穷大方向的整数部分。

例 2-18

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>B = ceil(A)
```

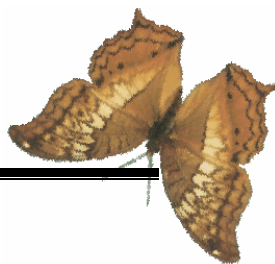
计算结果为:

```
B =
Columns 1 through 4
-1.0000     0     4.0000     6.0000
Columns 5 through 6
 7.0000    3.0000 + 4.0000i
```

函数 **exp**

功能 以 e 为底数的指数函数

格式 **Y = exp(X)** % 对参量 X 的每一分量, 求以 e 为底数的指数函数 Y 。 X 中的分量可以为复数。对于复数分量如, $z = x + i \cdot y$, 则相应地计算: $e^z = e^x \cdot (\cos(y) + i \cdot \sin(y))$ 。



例 2-19

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>Y = exp(A)
```

计算结果为:

```
Y =
    1.0e+003 *
    Columns 1 through 4
    0.0001    0.0008    0.0231    0.2704
    Columns 5 through 6
    1.0966   -0.0099 - 0.0049i
```

函数 **expm**

功能 求矩阵的以 e 为底数的指数函数

格式 **Y = expm(X)** % 计算以 e 为底数、x 的每一个元素为指数的指数函数值。若矩阵 x 有小于等于零的特征值，则返回复数的结果。

说明 该函数为一内建函数，它有三种计算算法：

- (1) 使用文件 **expm1.m** 中的用比例法与二次幂算法得到的 Pad 近似值；
- (2) 使用 Taylor 级数近似展开式计算，这种计算在文件 **expm2.m** 中。但这种一般计算方法是不可取的，通常计算是缓慢且不精确的；
- (3) 在文件 **expm3.m** 中，先将矩阵对角线化，再把函数计算出相应的特征向量，最后转换过来。但当输入的矩阵没有与矩阵阶数相同的特征向量个数时，就会出现错误。

例 2-20

```
>>A=hilb(4);
>>Y = expm(A)
```

计算结果为:

```
Y =
    3.2506    1.2068    0.8355    0.6417
    1.2068    1.7403    0.5417    0.4288
    0.8355    0.5417    1.4100    0.3318
    0.6417    0.4288    0.3318    1.2729
```

函数 **log**

功能 自然对数，即以 e 为底数的对数。

格式 **Y = log(X)** % 对参量 X 中的每一个元素计算自然对数。其中 X 中的元素可以是复数与负数，但由此可能得到意想不到的结果。若 $z = x + i*y$ ，则 $\log(z) = \log(\text{abs}(z)) + i*\text{atan2}(y,x)$

例 2-21 下面的语句可以得到无理数 π 的近似值：

```
>>Pi = abs(log(-1))
```

计算结果为:

```
Pi =
    3.1416
```

函数 **log10**

功能 常用对数，即以 10 为底数的对数。

格式 **Y = log10(X)** % 计算 X 中的每一个元素的常用对数，若 X 中出现复数，则可能得到意想不到的结果。

例 2-22



```
>>L1 = log10(realmax) % 由此可得特殊变量 realmax 的近似值
>>L2 = log10(eps) % 由此可得特殊变量 eps 的近似值
>>M = magic(4);
>>L3 = log10(M)
```

计算结果为:

```
L1 =
    308.2547
L2 =
   -15.6536
L3 =
    1.2041    0.3010    0.4771    1.1139
    0.6990    1.0414    1.0000    0.9031
    0.9542    0.8451    0.7782    1.0792
    0.6021    1.1461    1.1761         0
```

函数 sort

功能 把输入参量中的元素按从小到大的方向重新排列

格式 `B = sort(A)` %沿着输入参量 A 的不同维的方向、从小到大重新排列 A 中的元素。A 可以是字符串的、实数的、复数的单元数组。对于 A 中完全相同的元素,则按它们在 A 中的先后位置排列在一块;若 A 为复数的,则按元素幅值的从小到大排列,若有幅值相同的复数元素,则再按它们在区间 $[-\pi, \pi]$ 的幅角从小到大排列;若 A 中有元素为 NaN,则将它们排到最后。若 A 为向量,则返回从小到大的向量,若 A 为二维矩阵,则按列的方向进行排列;若 A 为多维数组,sort(A) 把沿着第一非单元素集的元素象向量一样进行处理。

`B = sort(A,dim)` %沿着矩阵 A (向量的、矩阵的或多维的)中指定维数 dim 方向重新排列 A 中的元素。

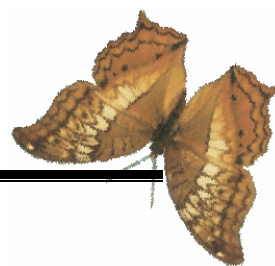
`[B,INDEX] = sort(A,...)` %输出参量 B 的结果如同上面的情形,输出 INDEX 是一等于 size(A)的数组,它的每一列是与 A 中列向量的元素相对应的置换向量。若 A 中有重复出现的相同的值,则返回保存原来相对位置的索引。

例 2-23

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>[B1,INDEX] = sort(A)
>>M = magic(4);
>>B2 = sort(M)
```

计算结果为:

```
B1 =
Columns 1 through 4
   -0.2000   -1.9000    3.1416    2.4000 + 3.6000i
Columns 5 through 6
    5.6000    7.0000
INDEX =
     2     1     3     6     4     5
B2 =
     4     2     3     1
     5     7     6     8
     9    11    10    12
```



16 14 15 13

函数 abs

功能 数值的绝对值与复数的幅值

格式 $Y = \text{abs}(X)$ %返回参量 X 的每一个分量的绝对值；若 X 为复数的，则返回每一分量的幅值： $\text{abs}(X) = \sqrt{\text{real}(X)^2 + \text{imag}(X)^2}$ 。

例 2-24

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>Y = abs(A)
```

计算结果为：

```
Y =
    1.9000    0.2000    3.1416    5.6000    7.0000    4.3267
```

函数 conj

功能 复数的共轭值

格式 $ZC = \text{conj}(Z)$ %返回参量 Z 的每一个分量的共轭复数：
 $\text{conj}(Z) = \text{real}(Z) - i * \text{imag}(Z)$

函数 imag

功能 复数的虚数部分

格式 $Y = \text{imag}(Z)$ %返回输入参量 Z 的每一个分量的虚数部分。

例 2-25

```
>>imag(2+3i)
```

计算结果为：

```
ans =
     3
```

函数 real

功能 复数的实数部分。

格式 $Y = \text{real}(Z)$ %返回输入参量 Z 的每一个分量的实数部分。

例 2-26

```
>>real(2+3i)
```

计算结果为：

```
ans =
     2
```

函数 angle

功能 复数的相角

格式 $P = \text{angle}(Z)$ %返回输入参量 Z 的每一复数元素的、单位为弧度的相角，其值在区间 $[-\pi, \pi]$ 上。

说明 $\text{angle}(z) = \text{imag}(\log(z)) = \text{atan2}(\text{imag}(z), \text{real}(z))$

例 2-27

```
>>Z = [1-i, 2+i, 3-i, 4+i;
>>1+2i, 2-2i, 3+2i, 4-2i;
>>1-3i, 2+3i, 3-3i, 4+3i;
>>1+4i, 2-4i, 3+4i, 4-4i];
>>P = angle(Z)
```

计算结果为：

```
P =
```



```

-0.7854    0.4636   -0.3218    0.2450
 1.1071   -0.7854    0.5880   -0.4636
-1.2490    0.9828   -0.7854    0.6435
 1.3258   -1.1071    0.9273   -0.7854

```

函数 complex**功能** 用实数与虚数部分创建复数

格式 `c = complex(a,b)` %用两个实数 `a`, `b` 创建复数 `c=a+bi`。输出参量 `c` 与 `a`、`b` 同型（同为向量、矩阵、或多维阵列）。该命令比下列形式的复数输入更有用：`a + i*b` 或 `a + j*b` 因为 `i` 和 `j` 可能被用做其他的变量（不等于 `sqrt(-1)`），或者 `a` 和 `b` 不是双精度的。

`c = complex(a)` %输入参量 `a` 作为输出复数 `c` 的实部，其虚部为 0：`c = a+0*i`。

例 2-28

```

>>a = uint8([1;2;3;4]);
>>b = uint8([4;3;2;1]);
>>c = complex(a,b)

```

计算结果为：

```

c =
1.0000 + 4.0000i
2.0000 + 3.0000i
3.0000 + 2.0000i
4.0000 + 1.0000i

```

函数 mod**功能** 模数（带符号的除法余数）

用法 `M = mod(X,Y)` %输入参量 `X`、`Y` 应为整数，此时返回余数 `X - Y.*floor(X./Y)`，若 `Y` ≠ 0，或者是 `X`。若运算数 `x` 与 `y` 有相同的符号，则 `mod(X,Y)` 等于 `rem(X,Y)`。总之，对于整数 `x,y`，有：`mod(-x,y) = rem(-x,y)+y`。若输入为实数或复数，由于浮点数在计算机上的不精确表示，该操作将导致不可预测的结果。

例 2-29

```

>>M1 = mod(13,5)
>>M2 = mod([1:5],3)
>>M3 = mod(magic(3),3)

```

计算结果为：

```

M1 =
3
M2 =
1    2    0    1    2
M3 =
2    1    0
0    2    1
1    0    2

```

函数 nchoosek**功能** 二项式系数或所有的组合数。该命令只有对 `n<15` 时有用。

函数 `C = nchoosek(n,k)` %参量 `n,k` 为非负整数，返回 `n! / ((n-k)! k!)`，即一次从 `n` 个物体中取出 `k` 个的组合数。



$C = \text{nchoosek}(v,k)$ % 参量 v 为 n 维向量, 返回一矩阵, 其行向量的分量为一次性从 v 个物体中取 k 个物体的组合数。矩阵 C 包含 $C_k^n = n! / ((n-k)! k!)$ 行与 k 列。

例 2-30

```
>>C = nchoosek(2:2:10,4)
```

计算结果为:

```
C =
     2     4     6     8
     2     4     6    10
     2     4     8    10
     2     6     8    10
     4     6     8    10
```

函数 rand

功能 生成元素均匀分布于(0,1)上的数值与阵列

用法 $Y = \text{rand}(n)$ % 返回 $n \times n$ 阶的方阵 Y , 其元素均匀分布于区间(0,1)。若 n 不是一标量, 在显示一出错信息。

$Y = \text{rand}(m,n)$ 、 $Y = \text{rand}([m \ n])$ % 返回阶数为 $m \times n$ 的, 元素均匀分布于区间(0,1)上矩阵 Y 。

$Y = \text{rand}(m,n,p,\dots)$ 、 $Y = \text{rand}([m \ n \ p \ \dots])$ % 生成阶数 $m \times n \times p \times \dots$ 的, 元素服从均匀分布的多维随机阵列 Y 。

$Y = \text{rand}(\text{size}(A))$ % 生成一与阵列 A 同型的随机均匀阵列 Y

rand % 该命令在每次单独使用时, 都返回一随机数(服从均匀分布)。

$s = \text{rand}('state')$ % 返回一有 35 元素的列向量 s , 其中包含均匀分布生成器的当前状态。该改变生成器的当前的状态, 见表 2-1。

表 2-1

命 令	含 义
$\text{Rand}('state',s)$	设置状态为 s
$\text{Rand}('state',0)$	设置生成器为初始状态
$\text{Rand}('state',k)$	设置生成器第 k 个状态(k 为整数)
$\text{Rand}('state',\text{sum}(100 \times \text{clock}))$	设置生成器在每次使用时的状态都不同 (因为 clock 每次都不同)

例:

```
>>R1 = rand(4,5)
```

```
>>a = 10; b = 50;
```

```
>>R2 = a + (b-a) * rand(5) % 生成元素均匀分布于(10,50)上的矩阵
```

计算结果可能为:

```
R1 =
    0.6655    0.0563    0.2656    0.5371    0.6797
    0.3278    0.4402    0.9293    0.5457    0.6129
    0.6325    0.4412    0.9343    0.9394    0.3940
    0.5395    0.6501    0.5648    0.7084    0.2206

R2 =
   33.6835   19.8216   36.9436   49.6289   46.4679
   18.5164   34.2597   15.3663   31.0549   49.0377
   19.0026   37.1006   33.6046   39.5361   13.9336
   12.4641   12.9804   35.5420   23.2916   46.8304
   28.5238   48.7418   49.0843   13.0512   10.9265
```

函数 randn



功能 生成元素服从正态分布 ($N(0,1)$) 的数值与阵列

格式 $Y = \text{randn}(n)$ % 返回 $n \times n$ 阶的方阵 Y , 其元素服从正态分布 $N(0,1)$ 。若 n 不是一标量, 则显示一出错信息。

$Y = \text{randn}(m,n)$ 、 $Y = \text{randn}([m \ n])$ % 返回阶数为 $m \times n$ 的, 元素均匀分布于区间 $(0,1)$ 上矩阵 Y 。

$Y = \text{randn}(m,n,p,\dots)$ 、 $Y = \text{randn}([m \ n \ p \ \dots])$ % 生成阶数 $m \times n \times p \dots$ 的, 元素服从正态分布的多维随机阵列 Y 。

$Y = \text{randn}(\text{size}(A))$ % 生成一与阵列 A 同型的随机正态阵列 Y

randn % 该命令在每次单独使用时, 都返回一随机数 (服从正态分布)。

$s = \text{randn}('state')$ % 返回一有 2 元素的向量 s , 其中包含正态分布生成器的当前状态。该改变生成器的当前状态, 见表 2-2。

表 2-2

命 令	含 义
$\text{randn}('state',s)$	设置状态为 s
$\text{randn}('state',0)$	设置生成器为初始状态
$\text{rand}('state',k)$	设置生成器第 k 个状态(k 为整数)
$\text{rand}('state',\text{sum}(100 \times \text{clock}))$	设置生成器在每次使用时的状态都不同 (因为 clock 每次都不同)

例:

```
>>R1 = rand(4,5)
>>R2 = 0.6 + sqrt(0.1) * randn(5)
```

计算结果可能为:

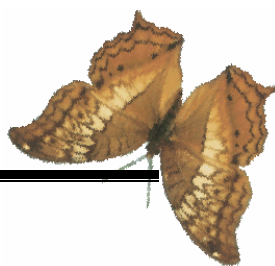
```
R1 =
    0.2778    0.2681    0.5552    0.5167    0.8821
    0.2745    0.3710    0.1916    0.3385    0.5823
    0.9124    0.5129    0.4164    0.2993    0.0550
    0.4125    0.2697    0.1508    0.9370    0.5878

R2 =
    0.4632    0.9766    0.5410    0.6360    0.6931
    0.0733    0.9760    0.8295    0.9373    0.1775
    0.6396    0.5881    0.4140    0.6187    0.8259
    0.6910    0.7035    1.2904    0.5698    1.1134
    0.2375    0.6552    0.5569    0.3368    0.3812
```

2.2 插值、拟合与查表

插值法是实用的数值方法, 是函数逼近的重要方法。在生产和科学实验中, 自变量 x 与因变量 y 的函数 $y = f(x)$ 的关系式有时不能直接写出表达式, 而只能得到函数在若干个点的函数值或导数值。当要求知道观测点之外的函数值时, 需要估计函数值在该点的值。

如何根据观测点的值, 构造一个比较简单的函数 $y = \phi(x)$, 使函数在观测点的值等于已知的数值或导数值。用简单函数 $y = \phi(x)$ 在点 x 处的值来估计未知函数 $y = f(x)$ 在 x 点的值。寻找这样的函数 $\phi(x)$, 办法是很多的。 $\phi(x)$ 可以是一个代数多项式, 或是三角多项式, 也可以是有理分式; $\phi(x)$ 可以是任意光滑 (任意阶导数连续) 的函数或是分段函数。函数类的不同, 自然地有不同的逼近效果。在许多应用中, 通常要用一个解析函数 (一、二元函数) 来描述观测数据。



根据测量数据的类型：

1. 测量值是准确的，没有误差。
2. 测量值与真实值有误差。

这时对应地有两种处理观测数据方法：

1. 插值或曲线拟合。
2. 回归分析（假定数据测量是精确时，一般用插值法，否则用曲线拟合）。

MATLAB 中提供了众多的数据处理命令。有插值命令，有拟合命令，有查表命令。

2.2.1 插值命令

命令 1 `interp1`

功能 一维数据插值（表格查找）。该命令对数据点之间计算内插值。它找出一元函数 $f(x)$ 在中间点的数值。其中函数 $f(x)$ 由所给数据决定。各个参量之间的关系示意图为图 2-14。

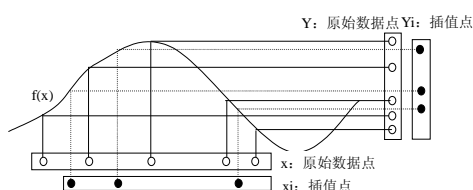


图 2-14 数据点与插值点关系示意图

格式 `yi = interp1(x,Y,xi)`

% 返回插值向量 y_i ，每一元素对应于参量 x_i ，同时由向量 x 与 Y 的内插值决定。参量 x 指定数据 Y 的点。若 Y 为一矩阵，则按 Y 的每列计算。 y_i 是阶数为 $\text{length}(x_i) \times \text{size}(Y, 2)$ 的输出矩阵。

`yi = interp1(Y,xi)`

% 假定 $x=1:N$ ，其中 N 为向量 Y 的长度，或者为矩阵 Y 的行数。

`yi = interp1(x,Y,xi,method)` % 用指定的算法计算插值：

'nearest': 最近邻点插值，直接完成计算；

'linear': 线性插值（缺省方式），直接完成计算；

'spline': 三次样条函数插值。对于该方法，命令 `interp1` 调用函数 `spline`、`ppval`、`mkpp`、`umkpp`。这些命令生成一系列用于分段多项式操作的函数。命令 `spline` 用它们执行三次样条函数插值；

'pchip': 分段三次 Hermite 插值。对于该方法，命令 `interp1` 调用函数 `pchip`，用于对向量 x 与 y 执行分段三次内插值。该方法保留单调性与数据的外形；

'cubic': 与 'pchip' 操作相同；

'v5cubic': 在 MATLAB 5.0 中的三次插值。

对于超出 x 范围的 x_i 的分量，使用方法 'nearest'、'linear'、'v5cubic' 的插值算法，相应地将返回 NaN。对其他的方法，`interp1` 将对超出的分量执行外插值算法。



`yi = interp1(x,Y,xi,method,'extrap')` %对于超出 x 范围的 xi 中的分量将执行特殊的外插值法 `extrap`。

`yi = interp1(x,Y,xi,method,extrapval)` %确定超出 x 范围的 xi 中的分量的外插值 `extrapval`，其值通常取 NaN 或 0。

例 2-31

```
>>x = 0:10; y = x.*sin(x);
>>xx = 0:.25:10; yy = interp1(x,y,xx);
>>plot(x,y,'kd',xx,yy)
```

插值图形为图 2-15。

例 2-32

```
>> year = 1900:10:2010;
>> product = [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505
249.633 256.344 267.893];
>>p1995 = interp1(year,product,1995)
>>x = 1900:1:2010;
>>y = interp1(year,product,x,'pchip');
>>plot(year,product,'o',x,y)
```

插值结果为：

```
p1995 =
252.9885
```

插值图形为图 2-16。

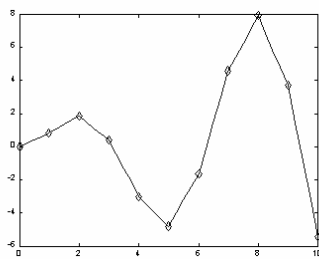


图 2-15 一元函数插值图形

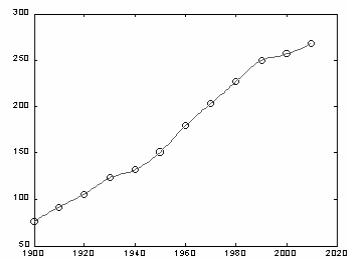


图 2-16 离散数据的一维插值图

命令 2 interp2

功能 二维数据内插值（表格查找）

格式 `ZI = interp2(X,Y,Z,XI,YI)` %返回矩阵 `ZI`，其元素包含对应于参量 `XI` 与 `YI`（可以是向量、或同型矩阵）的元素，即 $Z_i(i,j) \leftarrow [X_i(i,j), y_i(i,j)]$ 。用户可以输入行向量和列向量 `Xi` 与 `Yi`，此时，输出向量 `Zi` 与矩阵 `meshgrid(xi,yi)` 是同型的。同时取决于由输入矩阵 `X`、`Y` 与 `Z` 确定的二维函数 $Z=f(X,Y)$ 。参量 `X` 与 `Y` 必须是单调的，且相同的划分格式，就像由命令 `meshgrid` 生成的一样。若 `Xi` 与 `Yi` 中有在 `X` 与 `Y` 范围之外的点，则相应地返回 `nan`（Not a Number）。

`ZI = interp2(Z,XI,YI)` %缺省地， $X=1:n$ 、 $Y=1:m$ ，其中 $[m,n]=\text{size}(Z)$ 。再按



第一种情形进行计算。

`ZI = interp2(Z,n)`

%作 n 次递归计算，在 Z 的每两个元素之间插入它们的二维插值，这样， Z 的阶数将不断增加。`interp2(Z)` 等价于 `interp2(z,1)`。

`ZI = interp2(X,Y,Z,XI,YI,method)` %用指定的算法 `method` 计算二维插值：

'linear': 双线性插值算法（缺省算法）；

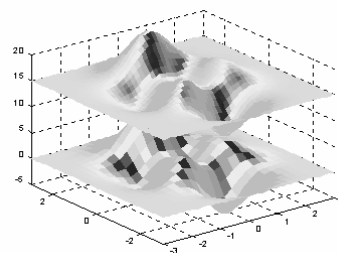
'nearest': 最临近插值；

'spline': 三次样条插值；

'cubic': 双三次插值。

例 2-33:

```
>>[X,Y] = meshgrid(-3:.25:3);
>>Z = peaks(X,Y);
>>[XI,YI] = meshgrid(-3:.125:3);
>>ZZ = interp2(X,Y,Z,XI,YI);
>>surfl(X,Y,Z);hold on;
>>surfl(XI,YI,ZZ+15)
>>axis([-3 3 -3 3 -5 20]);shading flat
>>hold off
```



插值图形为图 2-17。

图 2-17 二维插值图

例 2-34

```
>>years = 1950:10:1990;
>>service = 10:10:30;
>>wage = [150.697 199.592 187.625
179.323 195.072 250.287
203.212 179.092 322.767
226.505 153.706 426.730
249.633 120.281 598.243];
>>w = interp2(service,years,wage,15,1975)
```

插值结果为：

```
w =
190.6288
```

命令 3 **interp3**

功能 三维数据插值（查表）

格式 `VI = interp3(X,Y,Z,V,XI,YI,ZI)` %找出由参量 X,Y,Z 决定的三元函数 $V=V(X,Y,Z)$ 在点 (XI,YI,ZI) 的值。参量 XI,YI,ZI 是同型阵列或向量。若向量参量 XI,YI,ZI 是不同长度，不同方向（行或列）的向量，这时输出参量 VI 与 $Y1,Y2,Y3$ 为同型矩阵。其中 $Y1,Y2,Y3$ 为用命令 `meshgrid(XI,YI,ZI)` 生成的同型阵列。若插值点 (XI,YI,ZI) 中有位于点 (X,Y,Z) 之外的点，则相应地返回特殊变量值 `NaN`。

`VI = interp3(V,XI,YI,ZI)` %缺省地， $X=1:N$ ， $Y=1:M$ ， $Z=1:P$ ，其中， $[M,N,P]=size(V)$ ，再按上面的情形计算。

`VI = interp3(V,n)` %作 n 次递归计算，在 V 的每两个元素之间插入它们的三维插值。这样， V 的阶数将不断增加。`interp3(V)` 等价于 `interp3(V,1)`。



`VI = interp3(...,method)` %用指定的算法 `method` 作插值计算:

‘linear’: 线性插值 (缺省算法);

‘cubic’: 三次插值;

‘spline’: 三次样条插值;

‘nearest’: 最邻近插值。

说明 在所有的算法中, 都要求 `X,Y,Z` 是单调且有相同的格点形式。当 `X,Y,Z` 是等距且单调时, 用算法 ‘linear’, ‘cubic’, ‘nearest’, 可得到快速插值。

例 2-35

```
>>[x,y,z,v] = flow(20);
>>[xx,yy,zz] = meshgrid(1:25:10, -3:25:3, -3:25:3);
>>vv = interp3(x,y,z,v,xx,yy,zz);
>>slice(xx,yy,zz,vv,[6 9.5],[1 2],[-2 .2]); shading interp;colormap cool
```

插值图形为图 2-18。

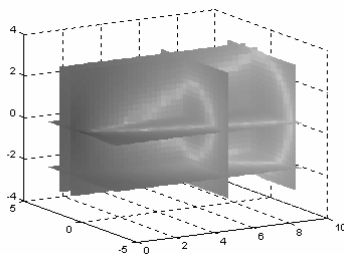


图 2-18 三维插值图

命令 4 `interpft`

功能 用快速 Fourier 算法作一维插值

格式 `y = interpft(x,n)` %返回包含周期函数 `x` 在重采样的 `n` 个等距的点的插值 `y`。若 `length(x)=m`, 且 `x` 有采样间隔 `dx`, 则新的 `y` 的采样间隔 `dy=dx*m/n`。注意的是必须 `n ≥ m`。若 `x` 为一矩阵, 则按 `x` 的列进行计算。返回的矩阵 `y` 有与 `x` 相同的列数, 但有 `n` 行。

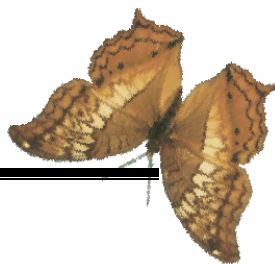
`y = interpft(x,n,dim)` %沿着指定的方向 `dim` 进行计算

命令 5 `griddata`

功能 数据格点

格式 `ZI = griddata(x,y,z,XI,YI)` %用二元函数 `z=f(x,y)` 的曲面拟合有不规则的数据向量 `x,y,z`。 `griddata` 将返回曲面 `z` 在点 `(XI,YI)` 处的插值。曲面总是经过这些数据点 `(x,y,z)` 的。输入参量 `(XI,YI)` 通常是规则的格点 (像用命令 `meshgrid` 生成的一样)。 `XI` 可以是一行向量, 这时 `XI` 指定一有常数列向量的矩阵。类似地, `YI` 可以是一列向量, 它指定一有常数行向量的矩阵。

`[XI,YI,ZI] = griddata(x,y,z,xi,yi)` %返回的矩阵 `ZI` 含义同上, 同时, 返回的矩阵 `XI,YI` 是由行向量 `xi` 与列向量 `yi` 用命令



meshgrid 生成的。

`[...] = griddata(...,method)` %用指定的算法 method 计算:

‘linear’: 基于三角形的线性插值 (缺省算法);

‘cubic’: 基于三角形的三次插值;

‘nearest’: 最邻近插值法;

‘v4’: MATLAB 4 中的 griddata 算法。

命令 6 spline

功能 三次样条数据插值

格式 `yy = spline(x,y,xx)` %对于给定的离散的测量数据 x,y (称为断点), 要寻找一个三项多项式 $y = p(x)$, 以逼近每对数据 (x,y) 点间的曲线。过两点 (x_i, y_i) 和 (x_{i+1}, y_{i+1}) 只能确定一条直线, 而通过一点的三次多项式曲线有无穷多条。为使通过中间断点的三次多项式曲线具有唯一性, 要增加两个条件 (因为三次多项式有 4 个系数):

1. 三次多项式在点 (x_i, y_i) 处有: $p'_i(x_i) = p''_i(x_i)$;
 2. 三次多项式在点 (x_{i+1}, y_{i+1}) 处有: $p'_i(x_{i+1}) = p''_i(x_{i+1})$;
 3. $p(x)$ 在点 (x_i, y_i) 处的斜率是连续的 (为了使三次多项式具有良好的解析性, 加上的条件);
 4. $p(x)$ 在点 (x_i, y_i) 处的曲率是连续的;
- 对于第一个和最后一个多项式, 人为地规定如下条件:

$$\textcircled{1}. p''_1(x) = p''_n(x)$$

$$\textcircled{2}. p'''_n(x) = p'''_{n-1}(x)$$

上述两个条件称为非结点(not-a-knot)条件。综合上述内容, 可知对数据拟合的三次样条函数 $p(x)$ 是一个分段的三次多项式:

$$p(x) = \begin{cases} p_1(x) & x_1 \leq x \leq x_2 \\ p_2(x) & x_2 \leq x \leq x_3 \\ \dots\dots & \dots\dots \\ p_n(x) & x_n \leq x \leq x_{n+1} \end{cases}, \text{ 其中每段 } p_i(x) \text{ 都是三次多项式。}$$

该命令用三次样条插值计算出由向量 x 与 y 确定的一元函数 $y=f(x)$ 在点 xx 处的值。若参量 y 是一矩阵, 则以 y 的每一列和 x 配对, 再分别计算由它们确定的函数在点 xx 处的值。则 yy 是一阶数为 $\text{length}(xx) \times \text{size}(y,2)$ 的矩阵。

`pp = spline(x,y)` %返回由向量 x 与 y 确定的分段样条多项式的系数矩阵 pp , 它可用于命令 `ppval`、`unmkpp` 的计算。

例 2-36

对离散地分布在 $y=\exp(x)\sin(x)$ 函数曲线上的数据点进行样条插值计算:

```
>>x = [0 2 4 5 8 12 12.8 17.2 19.9 20]; y = exp(x).*sin(x);
>>xx = 0:.25:20;
>>yy = spline(x,y,xx);
>>plot(x,y,'o',xx,yy)
```



插值图形结果为图 2-19。

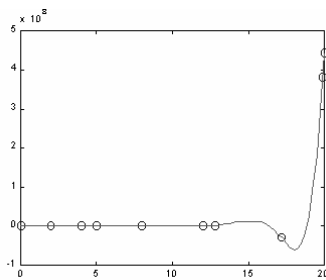


图 2-19 三次样条插值

命令 7 **interp**

功能 n 维数据插值（查表）

格式 $VI = \text{interp}(X1, X2, \dots, Xn, V, Y1, Y2, \dots, Yn)$ % 返回由参量 $X1, X2, \dots, Xn, V$ 确定的 n 元函数 $V = V(X1, X2, \dots, Xn)$ 在点 $(Y1, Y2, \dots, Yn)$ 处的插值。参量 $Y1, Y2, \dots, Yn$ 是同型的矩阵或向量。若 $Y1, Y2, \dots, Yn$ 是向量，则可以是不同长度，不同方向（行或列）的向量。它们将通过命令 **ndgrid** 生成同型的矩阵，再作计算。若点 $(Y1, Y2, \dots, Yn)$ 中有位于点 $(X1, X2, \dots, Xn)$ 之外的点，则相应地返回特殊变量 NaN。

$VI = \text{interp}(V, Y1, Y2, \dots, Yn)$ % 缺省地， $X1 = 1:\text{size}(V, 1)$, $X2 = 1:\text{size}(V, 2)$, ..., $Xn = 1:\text{size}(V, n)$ ，再按上面的情形计算。

$VI = \text{interp}(V, \text{ntimes})$ % 作 ntimes 次递归计算，在 V 的每两个元素之间插入它们的 n 维插值。这样， V 的阶数将不断增加。 $\text{interp}(V)$ 等价于 $\text{interp}(V, 1)$ 。

$VI = \text{interp}(\dots, \text{method})$ % 用指定的算法 method 计算：

‘linear’：线性插值（缺省算法）；

‘cubic’：三次插值；

‘spline’：三次样条插值法；

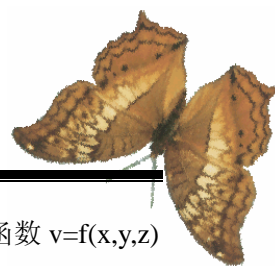
‘nearest’：最邻近插值算法。

命令 8 **meshgrid**

功能 生成用于画三维图形的矩阵数据。

格式 $[X, Y] = \text{meshgrid}(x, y)$ 将由向量 x, y （可以是不同方向的）指定的区域 $[\min(x), \max(x), \min(y), \max(y)]$ 用直线 $x = x(i), y = y(j)$ （ $i = 1, 2, \dots, \text{length}(x)$, $j = 1, 2, \dots, \text{length}(y)$ ）进行划分。这样，得到了 $\text{length}(x) * \text{length}(y)$ 个点，这些点的横坐标用矩阵 X 表示， X 的每个行向量与向量 x 相同；这些点的纵坐标用矩阵 Y 表示， Y 的每个列向量与向量 y 相同。其中 X, Y 可用于计算二元函数 $z = f(x, y)$ 与三维图形中 xy 平面矩形定义域的划分或曲面作图。

$[X, Y] = \text{meshgrid}(x)$ % 等价于 $[X, Y] = \text{meshgrid}(x, x)$ 。



`[X,Y,Z] = meshgrid(x,y,z)` %生成三维阵列 `X,Y,Z`, 用于计算三元函数 `v=f(x,y,z)` 或三维容积图。

例 2-37

`[X,Y] = meshgrid(1:3,10:14)`

计算结果为:

`X =`

1	2	3
1	2	3
1	2	3
1	2	3
1	2	3

`Y =`

10	10	10
11	11	11
12	12	12
13	13	13
14	14	14

命令 9 `ndgrid`

功能 生成用于多维函数计算或多维插值用的阵列

格式 `[X1,X2,...,Xn] = ndgrid(x1,x2,...,xn)` %把通过向量 `x1,x2,x3,...,xn` 指定的区域转换为数组 `x1,x2,x3,...,xn`。这样, 得到了 `length(x1)*length(x2)*...*length(xn)` 个点, 这些点的第一维坐标用矩阵 `X1` 表示, `X1` 的每个第一维向量与向量 `x1` 相同; 这些点的第二维坐标用矩阵 `X2` 表示, `X2` 的每个第二维向量与向量 `x2` 相同; 如此等等。其中 `X1,X2,...,Xn` 可用于计算多元函数 `y=f(x1,x2,...,xn)` 以及多维插值命令用到的阵列。

`[X1,X2,...,Xn] = ndgrid(x)` %等价于 `[X1,X2,...,Xn] = ndgrid(x,x,...,x)`

2.2.2 查表命令

命令 1 `table1`

功能 一维查表

格式 `Y = table1(TAB,X0)` %返回用表格矩阵 `TAB` 中的行线性插值元素, 对 `X0` (`TAB` 的第一列查找 `X0`) 进行线性插值得到的结果 `Y`。矩阵 `TAB` 是第一列包含关键值, 而其他列包含数据的矩阵。 `X0` 中的每一元素将相应地返回一线性插值行向量。矩阵 `TAB` 的第一列必须是单调的。

例 2-38

`>>tab = [(1:4)' hilb(4)]`
`>>y = table1(tab,[1 2.3 3.6 4])`

查表结果为:

`tab =`

1.0000	1.0000	0.5000	0.3333	0.2500
2.0000	0.5000	0.3333	0.2500	0.2000
3.0000	0.3333	0.2500	0.2000	0.1667
4.0000	0.2500	0.2000	0.1667	0.1429

Warning: TABLE1 is obsolete and will be removed in future versions. Use INTERP1 or INTERP1Q



```

instead.
> In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at line 31
y =
    1.0000    0.5000    0.3333    0.2500
    0.4500    0.3083    0.2350    0.1900
    0.2833    0.2200    0.1800    0.1524
    0.2500    0.2000    0.1667    0.1429

```

由上面结果可知，table1 是一将要废弃的命令。

命令 2 table2

功能 二维查表

格式 $Z = \text{table1}(\text{TAB}, X0, Y0)$ %返回用表格矩阵 TAB 中的行与列交叉线性插值元素，对 X0 (TAB 的第一列查找 X0) 进行线性插值，对 Y0 (TAB 的第一行查找 Y0) 进行线性插值，对上述两个数值进行交叉线性插值，得到的结果为 Z。矩阵 TAB 是第一列与第一行列都包含关键值，而其他的元素包含数据的矩阵。TAB(1,1)的关键值将被忽略。[X0,Y0]中的每点将相应地返回一线性插值。矩阵 TAB 的第一行与第一列必须是单调的。

例 2-39

```

>>tab = [NaN 1:4; (1:4)' magic(4)]
>>y = table2(tab,[2 3 3.7],[1.3 2.3 4])

```

查表的结果为：

```

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

```

Warning: TABLE2 is obsolete and will be removed in future versions. Use INTERP2 instead.

```
> In D:\MATLABR12\toolbox\matlab\polyfun\table2.m at line 24
```

Warning: TABLE1 is obsolete and will be removed in future versions. Use INTERP1 or INTERP1Q instead.

```
> In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at line 31
```

```
In D:\MATLABR12\toolbox\matlab\polyfun\table2.m at line 29
```

Warning: TABLE1 is obsolete and will be removed in future versions. Use INTERP1 or INTERP1Q instead.

```
> In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at line 31
```

```
In D:\MATLABR12\toolbox\matlab\polyfun\table2.m at line 31
```

```

y =
    6.8000    10.7000     8.0000
    8.4000     6.7000    12.0000
    7.4200    12.0200     4.3000

```

由上面的结果可知，table2 是将要废弃的命令。

2.3 数值积分

2.3.1 一元函数的数值积分

函数 1 quad、quadl、quad8



格式	<code>q = quad(fun,a,b)</code>	%近似地从 a 到 b 计算函数 fun 的数值积分, 误差为 10^{-6} 。
		若给 fun 输入向量 x, 应返回向量 y, 即 fun 是一单值函数。
	<code>q = quad(fun,a,b,tol)</code>	%用指定的绝对误差 tol 代替缺省误差。tol 越大, 函数计算的次数越少, 速度越快, 但结果精度变小。
	<code>q = quad(fun,a,b,tol,trace,p1,p2,...)</code>	%将可选参数 p1,p2,... 等传递给函数 fun(x,p1,p2,...), 再作数值积分。若 tol=[] 或 trace=[], 则用缺省值进行计算。
	<code>[q,n] = quad(fun,a,b,...)</code>	%同时返回函数计算的次数 n
	<code>... = quadl(fun,a,b,...)</code>	%用高精度进行计算, 效率可能比 quad 更好。
	<code>... = quad8(fun,a,b,...)</code>	%该命令是将废弃的命令, 用 quadl 代替。

```
>>fun = inline('3*x.^2./(x.^3-2*x.^2+3)');
>>Q1 = quad(fun,0,2)
>>Q2 = quadl(fun,0,2)
```

Q1 = 3.7224
Q2 = 3.7224

格式	$T = \text{trapz}(Y)$	%用等距梯形法近似计算 Y 的积分。若 Y 是一向量, 则 $\text{trapz}(Y)$ 为 Y 的积分; 若 Y 是一矩阵, 则 $\text{trapz}(Y)$ 为 Y 的每一列的积分; 若 Y 是一多维阵列, 则 $\text{trapz}(Y)$ 沿着 Y 的第一个非单元集的方向进行计算。
	$T = \text{trapz}(X,Y)$	%用梯形法计算 Y 在 X 点上的积分。若 X 为一列向量, Y 为矩阵, 且 $\text{size}(Y,1) = \text{length}(X)$, 则 $\text{trapz}(X,Y)$ 通过 Y 的第一个非单元集方向进行计算。
	$T = \text{trapz}(\dots, \text{dim})$	%沿着 dim 指定的方向对 Y 进行积分。若参量中包含 X , 则应有 $\text{length}(X) = \text{size}(Y, \text{dim})$ 。

```
>>X = -1:1:1;  
>>Y = 1./(1+25*X.^2);  
>>T = trapz(X,Y)
```

$$T = 0.5492$$

功能 有理分式近似。虽然所有的浮点数值都是有理数，有时用简单的有理数字（分子与分母都是较小的整数）近似地表示它们是有必要的。函数 `rat` 将试图做到这一点。对于有连续出现的小数的数值，将会用有理式近似表示它们。函数 `rats` 调用函数 `rat`，且返回字符



串。

格式	<code>[N,D] = rat(X)</code>	%对于缺省的误差 $1.e-6*\text{norm}(X(:,1))$, 返回阵列 N 与 D, 使 $N./D$ 近似为 X。
	<code>[N,D] = rat(X,tol)</code>	%在指定的误差 tol 范围内, 返回阵列 N 与 D, 使 $N./D$ 近似为 X。
	<code>rat(X)、rat(X···)</code>	%在没有输出参量时, 简单地显示 x 的连续分数。
	<code>S = rats(X,strlen)</code>	%返回一包含简单形式的、X 中每一元素的有理近似字符串 S, 若对于分配的空间中不能显示某一元素, 则用星号表示。该元素与 X 中其他元素进行比较而言较小, 但并非是可以忽略。参量 strlen 为函数 rats 中返回的字符串元素的长度。缺省值为 <code>strlen=13</code> , 这允许在 78 个空格中有 6 个元素。
	<code>S = rats(X)</code>	%返回与用 MATLAB 命令 <code>format rat</code> 显示 X 相同的结果给 S。

例 2-42

```
>>s = 1-1/2+1/3-1/4+1/5-1/6+1/7
>>format rat
>>S1 = rats(s)
>>S2 = rat(s)
>>[n,d] = rat(s)
>>PI1 = rats(pi)
>>PI2 = rat(pi)
```

计算结果为:

```
s =
    0.7595
S1 =
    319/420
S2 =
    1 + 1/(-4 + 1/(-6 + 1/(-3 + 1/(-5))))
n =
    319
d =
    420
PI1 =
    355/113
PI2 =
    3 + 1/(7 + 1/(16))
```

2.3.2 二元函数重积分的数值计算

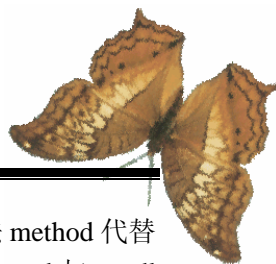
函数 1 dblquad

功能 矩形区域上的二重积分的数值计算

格式 `q = dblquad(fun,xmin,xmax,ymin,ymax)` %调用函数 `quad` 在区域 `[xmin,xmax,ymin,ymax]` 上计算二元函数 $z=f(x,y)$ 的二重积分。输入向量 `x`, 标量 `y`, 则 $f(x,y)$ 必须返回一用于积分的向量。

`q = dblquad(fun,xmin,xmax,ymin,ymax,tol)` %用指定的精度 `tol` 代替缺省精度 10^{-6} , 再进行计算。





`q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)` %用指定的算法 `method` 代替缺省算法 `quad`。`method` 的取值有 `@quadl` 或用户指定的、与命令 `quad` 与 `quadl` 有相同调用次序的函数句柄。

`q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method,p1,p2,...)` %将可选参数 `p1,p2,...` 等传递给函数 `fun(x,y,p1,p2,...)`。若 `tol=[]`, `method=[]`, 则使用缺省精度和算法 `quad`。

例 2-43

```
>>fun = inline('y./sin(x)+x.*exp(y)');
>>Q = dblquad(fun,1,3,5,7)
```

计算结果为:

```
Q =
3.8319e+003
```

函数 2 quad2dggen

功能 任意区域上二元函数的数值积分

格式 `q = quad2dggen(fun,xlower,xupper,ymin,ymax)` %在由 `[xlower,xupper, ymin,ymax]` 指定的区域上计算二元函数 $z=f(x,y)$ 的二重积分。

`q = dblquad(fun,xlower,xupper,ymin,ymax,tol)` %用指定的精度 `tol` 代替缺省精度 10^{-6} , 再进行计算。

`q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)` %用指定的算法 `method` 代替缺省算法。`method` 的取值有缺省算法或用户指定的、与缺省命令有相同调用次序的函数句柄。

`q=dblquad(fun,xlower,xupper,ymin,ymax,tol,method,p1,p2,...)` %将可选参数 `p1,p2,...` 等传递给函数 `fun(x,y,p1,p2,...)`。若 `tol=[]`, `method=[]`, 则使用缺省精度和算法。

例 2-44

计算单位圆域上的积分: $I = \iint_{x^2+y^2 \leq 1} e^{-\frac{x^2}{2}} \sin(x^2 + y) dx dy$

先把二重积分转化为二次积分的形式: $I = \int_{-1}^1 dy \int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} e^{-\frac{x^2}{2}} \sin(x^2 + y) dx$

```
f = inline('exp(-x.^2/2).*sin(x.^2+y)','x','y');
xlower = inline('-sqrt(1-y.^2)','y'); xupper = inline('sqrt(1-y.^2)','y');
Q = quad2dggen(fun,xlower,xupper,-1,1,1e-4)
```

计算结果为:

```
Q =
0.5368603818
```

2.4 常微分方程数值解

函数 `ode45`、`ode23`、`ode113`、`ode15s`、`ode23s`、`ode23t`、`ode23tb`



功能 常微分方程 (ODE) 组初值问题的数值解

参数说明:

solver 为命令 ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb 之一。

Odefun 为显式常微分方程 $y' = f(t, y)$, 或为包含一混合矩阵的方程 $M(t, y) * y' = f(t, y)$ 。命令 ode23 只能求解常数混合矩阵的问题; 命令 ode23t 与 ode15s 可以求解奇异矩阵的问题。

Tspan 积分区间 (即求解区间) 的向量 $tspan = [t_0, t_f]$ 。要获得问题在其他指定时间点 t_0, t_1, t_2, \dots 上的解, 则令 $tspan = [t_0, t_1, t_2, \dots, t_f]$ (要求是单调的)。

Y0 包含初始条件的向量。

Options 用命令 odeset 设置的可选积分参数。

p1, p2, ... 传递给函数 odefun 的可选参数。

格式 $[T, Y] = \text{solver}(\text{odefun}, tspan, y_0)$ % 在区间 $tspan = [t_0, t_f]$ 上, 从 t_0 到 t_f , 用初始条件 y_0 求解显式微分方程 $y' = f(t, y)$ 。对于标量 t 与列向量 y , 函数 $f = \text{odefun}(t, y)$ 必须返回一 $f(t, y)$ 的列向量 f 。解矩阵 Y 中的每一行对应于返回的时间列向量 T 中的一个时间点。要获得问题在其他指定时间点 t_0, t_1, t_2, \dots 上的解, 则令 $tspan = [t_0, t_1, t_2, \dots, t_f]$ (要求是单调的)。

$[T, Y] = \text{solver}(\text{odefun}, tspan, y_0, \text{options})$ % 用参数 **options** (用命令 odeset 生成) 设置的属性 (代替了缺省的积分参数), 再进行操作。常用的属性包括相对误差值 **RelTol** (缺省值为 $1e-3$) 与绝对误差向量 **AbsTol** (缺省值为每一元素为 $1e-6$)。

$[T, Y] = \text{solver}(\text{odefun}, tspan, y_0, \text{options}, p_1, p_2, p_3, \dots)$ 将参数 p_1, p_2, p_3, \dots 等传递给函数 **odefun**, 再进行计算。若没有参数设置, 则令 **options** = []。

1. 求解具体 ODE 的基本过程:

(1) 根据问题所属学科中的规律、定律、公式, 用微分方程与初始条件进行描述。

$$F(y, y', y'', \dots, y^{(n)}, t) = 0$$

$$y(0) = y_0, y'(0) = y_1, \dots, y^{(n-1)}(0) = y_{n-1}$$

而 $y = [y; y(1); y(2); \dots, y(m-1)]$, n 与 m 可以不等

(2) 运用数学中的变量替换: $y_n = y^{(n-1)}, y_{n-1} = y^{(n-2)}, \dots, y_2 = y_1 = y$, 把高阶 (大于 2 阶) 的方

程 (组) 写成一阶微分方程组:

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} = \begin{bmatrix} f_1(t, y) \\ f_2(t, y) \\ \vdots \\ f_n(t, y) \end{bmatrix}, \quad y_0 = \begin{bmatrix} y_1(0) \\ y_2(0) \\ \vdots \\ y_n(0) \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

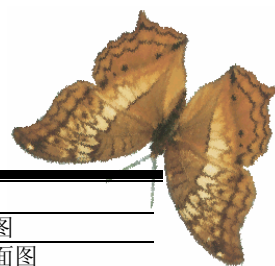
(3) 根据 (1) 与 (2) 的结果, 编写能计算导数的 M-函数文件 **odefile**。

(4) 将文件 **odefile** 与初始条件传递给求解器 **Solver** 中的一个, 运行后就可得到 ODE 的、在指定时间区间上的解列向量 y (其中包含 y 及不同阶的导数)。

2. 求解器 Solver 与方程组的关系表见表 2-3。

表 2-3

函数指令		含 义	函 数		含 义
求解器 Solver	ode23	普通 2-3 阶法解 ODE	odefile		包含 ODE 的文件
	ode23s	低阶法解刚性 ODE	选项	odeset	创建、更改 Solver 选项
	ode23t	解适度刚性 ODE		odeget	读取 Solver 的设置值



ode23tb	低阶法解刚性 ODE	输出	odeplot	ODE 的时间序列图
ode45	普通 4-5 阶法解 ODE		odephas2	ODE 的二维相平面图
ode15s	变阶法解刚性 ODE		odephas3	ODE 的三维相平面图
ode113	普通变阶法解 ODE		odeprint	在命令窗口输出结果

3. 因为没有一种算法可以有效地解决所有的 ODE 问题, 为此, MATLAB 提供了多种求解器 Solver, 对于不同的 ODE 问题, 采用不同的 Solver。

表 2-4 不同求解器 Solver 的特点

求解器 Solver	ODE 类型	特点	说明
ode45	非刚性	一步算法; 4, 5 阶 Runge-Kutta 方程; 累计截断误差达 $(\Delta x)^3$	大部分场合的首选算法
ode23	非刚性	一步算法; 2, 3 阶 Runge-Kutta 方程; 累计截断误差达 $(\Delta x)^3$	使用于精度较低的情形
ode113	非刚性	多步法; Adams 算法; 高低精度均可到 $10^{-3} \sim 10^{-6}$	计算时间比 ode45 短
ode23t	适度刚性	采用梯形算法	适度刚性情形
ode15s	刚性	多步法; Gear's 反向数值微分; 精度中等	若 ode45 失效时, 可尝试使用
ode23s	刚性	一步法; 2 阶 Rosebrock 算法; 低精度	当精度较低时, 计算时间比 ode15s 短
ode23tb	刚性	梯形算法; 低精度	当精度较低时, 计算时间比 ode15s 短

4. 在计算过程中, 用户可以对求解指令 solver 中的具体执行参数进行设置 (如绝对误差、相对误差、步长等)。

表 2-5 Solver 中 options 的属性

属性名	取值	含义
AbsTol	有效值: 正实数或向量 缺省值: $1e-6$	绝对误差对应于解向量中的所有元素; 向量则分别对应于解向量中的每一分量
RelTol	有效值: 正实数 缺省值: $1e-3$	相对误差对应于解向量中的所有元素。在每步(第 k 步)计算过程中, 误差估计为: $e(k) \leq \max(\text{RelTol} * \text{abs}(y(k)), \text{AbsTol}(k))$
NormControl	有效值: on、off 缺省值: off	为 'on' 时, 控制解向量范数的相对误差, 使每步计算中, 满足: $\text{norm}(e) \leq \max(\text{RelTol} * \text{norm}(y), \text{AbsTol})$
Events	有效值: on、off	为 'on' 时, 返回相应的事件记录
OutputFcn	有效值: odeplot、odephas2、odephas3、odeprint 缺省值: odeplot	若无输出参量, 则 solver 将执行下面操作之一: 画出解向量中各元素随时间的变化; 画出解向量中前两个分量构成的相平面图; 画出解向量中前三个分量构成的三维相空间图; 随计算过程, 显示解向量
OutputSel	有效值: 正整数向量 缺省值: []	若不使用缺省设置, 则 OutputFcn 所表现的是那些正整数指定的解向量中的分量的曲线或数据。若为缺省值时, 则缺省地按上面情形进行操作
Refine	有效值: 正整数 $k > 1$ 缺省值: $k = 1$	若 $k > 1$, 则增加每个积分步中的数据点记录, 使解曲线更加的光滑
Jacobian	有效值: on、off 缺省值: off	若为 'on' 时, 返回相应的 ode 函数的 Jacobi 矩阵
Jpattern	有效值: on、off 缺省值: off	为 'on' 时, 返回相应的 ode 函数的稀疏 Jacobi 矩阵
Mass	有效值: none、M、M(t)、M(t,y) 缺省值: none	M: 不随时间变化的常数矩阵 M(t): 随时间变化的矩阵 M(t,y): 随时间、地点变化的矩阵
MaxStep	有效值: 正实数 缺省值: $\text{tspan}/10$	最大积分步长



例 2-45 求解描述振荡器的经典的 Ver der Pol 微分方程 $\frac{d^2 y}{dt^2} - \mu(1 - y^2) \frac{dy}{dt} + 1 = 0$

$$y(0)=1, y'(0)=0$$

令 $x_1=y, x_2=dy/dx$, 则

$$dx_1/dt = x_2$$

$$dx_2/dt = \mu(1-x_2^2)-x_1$$

编写函数文件 verderpol.m:

```
function xprime = verderpol(t,x)
global MU
xprime = [x(2);MU*(1-x(1)^2)*x(2)-x(1)];
```

再在命令窗口中执行:

```
>>global MU
>>MU = 7;
>>Y0=[1;0]
>>[t,x] = ode45('verderpol',0,40,Y0);
>>x1=x(:,1);x2=x(:,2);
>>plot(t,x1,t,x2)
```

图形结果为图 2-20。

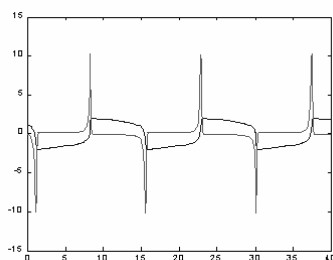


图 2-20 Ver der Pol 微分方程图

2.5 偏微分方程的数值解

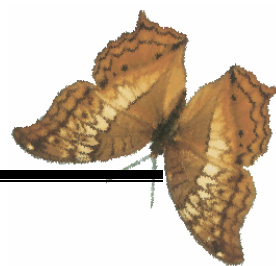
MATLAB 提供了一个专门用于求解偏微分方程的工具箱—PDE Toolbox (Partial Difference Equation)。在本章, 我们仅提供一些最简单、经典的偏微分方程, 如: 椭圆型、双曲型、抛物型等少数的偏微分方程, 并给出求解方法。用户可以从了解其解题基本方法, 从而解决相类似的问题。

Matlab 能解决的偏微分类型

$$-\nabla \cdot (c \nabla u) + au = f \quad \text{其中 } u = u(x,y), (x,y) \in G$$

$$\nabla(c \nabla u) = \frac{\partial}{\partial x} \left(c \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(c \frac{\partial u}{\partial y} \right), \quad f \in L_2(G)$$

$$c = c(x,y) \in C^1(\partial G), \quad a \geq 0, \quad a \in C^0(\partial G)$$



2.5.1 单的 Poisson 方程

Poisson 方程是特殊的椭圆型方程:
$$\begin{cases} -\nabla^2 u = 1, & G = \{(x,y) | x^2 + y^2 \leq 1\} \\ u|_{\partial G} = 0 \end{cases}$$

即 $c = 1, a = 0, f = -1$

Poisson 的解析解为: $u = \frac{1-x^2-y^2}{4}$ 。在下面计算中, 用求得的数值解与精确解进行比较, 看误差如何。

方程求解

(1) 问题输入:

```
c = 1; a = 0; f = 1; % 方程的输入。给 c, a, f 赋值即可
g = 'circleg' % 区域 G, 内部已经定义为 circleg
b = 'circleb1' % u 在区域 G 的边界上的条件, 内部已经定义好
```

(2) 对单位圆进行网格化, 对求解区域 G 作剖分, 且作的是三角剖分:

```
[p,e,t] = initmesh(g,hmax', 1)
```

(3) 迭代求解:

```
error = []; err = 1;
while err > 0.001,
[p,e,t] = refinemesh('circleg',p,e,t);
u = assempde('circleb1',p,e,t,1,0,1);
exact = -(p(1,:) ^2 + p(2,:) ^2 - 1) / 4;
err = norm(u - exact, 'inf');
error = [error, err];
end
```

(4) 误差显示与区域 G 内的剖分点数:

```
Error: 1.292265e-002. Number of nodes: 25
Error: 4.079923e-003. Number of nodes: 81
Error: 1.221020e-003. Number of nodes: 289
Error: 3.547924e-004. Number of nodes: 1089
```

(5) 结果显示:

```
subplot(2,2,1), pdemesh(p,e,t) % 结果显示
title('数值解')
subplot(2,2,2), pdesurf(p,t,u) % 精确解显示
title('精确解')
subplot(2,2,3), pdesurf(p,t,u-exact) % 与精确解的误差
title('计算误差')
```

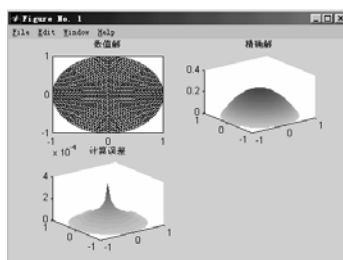


图 2-21 Poisson 方程图



2.5.2 双曲型偏微分方程

1. Matlab 能求解的类型: $d \frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c \nabla u) + au = f$

其中 $u = u(x, y, z)$, $(x, y, z) \in G$, $d = d(x, y, z) \in C^0(G)$, $a \geq 0$, $a \in C^0(\partial G)$, $f \in L_2(G)$

$$2. \text{形传递问题: } \begin{cases} \frac{\partial^2 u}{\partial t^2} - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0 \\ u|_{t=0} = 0 \\ \frac{\partial u}{\partial t}|_{t=0} = 0 \end{cases}, \quad G = \{(x, y, z) | 0 \leq x, y, z \leq 1\}$$

即: $c=1; a=0; f=0; d=1$

3. 方程求解

(1) 问题的输入:

```
c = 1; a = 0; f = 0; d = 1; % 输入方程的系数
g = 'squareg' % 输入方形区域 G, 内部已经定义好
b = 'squareb3' % 输入边界条件, 即初始条件
```

(2) 对单位矩形 G 进行网格化:

```
[p,e,t] = initmesh('squareg')
```

(3) 定解条件和求解时间点:

```
x = p(1,:); y = p(2,:);
u0 = atan(cos(pi/2*x));
ut0 = 3*sin(pi*x).*exp(sin(pi/2*y));
n = 31;
tlist = linspace(0,5,n);
```

(4) 求解:

```
uu = hyperbolic(u0, ut0, tlist, b, p, e, t, c, a, f, d);
```

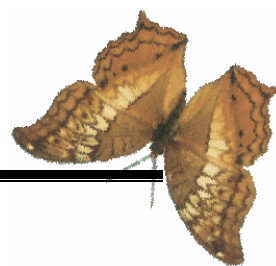
结果显示: 计算过程中的时间点和信息

```
Time: 0.166667
Time: 0.333333
Time: 4.33333
.....
Time: 4.66667
Time: 4.83333
Time: 5
428 successful steps
62 failed attempts
982 function evaluations
1 partial derivatives
142 LU decompositions
981 solutions of linear systems
```

(5) 动画显示:

```
delta=-1:0.1:1;
[uxy,tn,a2,a3]=tri2grid(p,t,uu(:,1),delta,delta);
gp=[tn;a2;a3];
umax=max(max(uu));
umin=min(min(uu));
```





```
newplot;M=moviein(n);
for i=1:n,
    pdeplot(p,e,t,'xydata',uu(:,i),'zdata',uu(:,i),...
        'mesh','off','xygrid','on','gridparam',gp,...
        'colorbar','off','zstyle','continuous');
    axis([-1 1 -1 1 umin umax]);
    caxis([umin umax]);
    M(:,i)=getframe;
end
movie(M,5)
```

图 2-22 是动画过程中的一个状态。

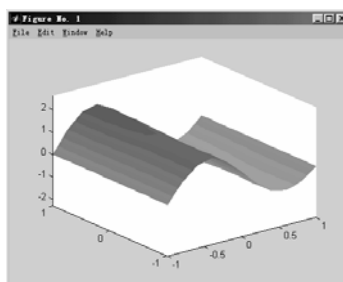


图 2-22 波动方程动画中的一个状态

2.5.3 抛物型偏微分方程

1. Matlab 能求解的类型: $d \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u) + au = f$

其中 $u = u(x, y, z)$, $(x, y, z) \in G$, $d = d(x, y, z) \in C^0(G)$, $a \geq 0$, $a \in C^0(\partial G)$, $f \in L_2(G)$

2. 热传导方程:
$$\begin{cases} \frac{\partial u}{\partial t} - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0, & G = \{(x, y, z) | 0 \leq x, y, z \leq 1\} \\ u|_{\partial G} = 0 \end{cases}$$

即: $c = 1; a = 0; f = 1; d = 1;$

3. 问题计算

(1) 问题的输入:

```
c = 1; a = 0; f = 1; d = 1; % 输入方程的系数
g = 'squareg'; % 输入方形区域 G
b = 'squareb1'; % 输入边界条件
```

(2) 对单位矩形的网格化:

```
[p,e,t] = initmesh(g);
```

(3) 定解条件和求解的时间点:

```
u0 = zeros(size(p, 2), 1);
ix = find(sqrt(p(1, :).^2 + p(2, :).^2) < 0.4);
u0(ix) = ones(size(ix));
nframes = 20;
tlist = linspace(0, 0.1, nframes) % 在时间[0, 0.1]内 20 个点上计算, 生成 20 帧
```

(4) 求解方程:



```
u1 = parabolic(u0, tlist, b, p, e, t, c, a, f, d)
```

计算结果如下:

```
Time: 0.00526316
```

```
Time: 0.0105263
```

```
.....
```

```
Time: 0.0947368
```

```
Time: 0.1
```

```
75 successful steps
```

```
1 failed attempts
```

```
154 function evaluations
```

```
1 partial derivatives
```

```
17 LU decompositions
```

```
153 solutions of linear systems
```

(5) 动画显示:

```
x = linspace(-1,1,31); y = x;
```

```
newplot;
```

```
Mv = moviein(nframes);
```

```
umax=max(max(u1));
```

```
umin=min(min(u1));
```

```
for j=1:nframes
```

```
    u=tri2grid(p,t,u1(:,j),tn,a2,a3);i=find(isnan(u));u(i)=zeros(size(i));...
```

```
    surf(x,y,u);caxis([umin umax]);colormap(cool),...
```

```
    axis([-1 1 -1 1 0 1]);...
```

```
    Mv(:,j) = getframe;...
```

```
end
```

```
movie(Mv,10)
```

图 2-23 是动画过程中的瞬间状态。

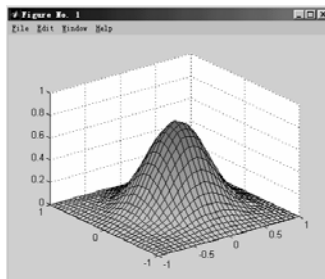
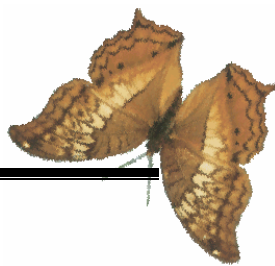


图 2-23 热传导方程动画瞬间状态图



第3章 符号运算

3.1 算术符号操作

命令 $+, -, *, ./, \backslash, \wedge, \wedge, ', '$
功能 符号矩阵的算术操作

用法如下:

$A+B$ 、 $A-B$ 符号阵列的加法与减法。

若 A 与 B 为同型阵列时, $A+B$ 、 $A-B$ 分别对对应分量进行加减; 若 A 与 B 中至少有一个为标量, 则把标量扩大为与另外一个同型的阵列, 再按对应的分量进行加减。

$A*B$ 符号矩阵乘法。

$A*B$ 为线性代数中定义的矩阵乘法。按乘法定义要求必须有矩阵 A 的列数等于矩阵 B 的行数。即: 若 $A_{n \times k} * B_{k \times m} = (a_{ij})_{n \times k} * (b_{ij})_{k \times m} = C_{n \times m} = (c_{ij})_{n \times m}$, 则

$c_{ij} = \sum_{s=1}^k a_{is} * b_{sj}$, $i=1,2,\dots,n$; $j=1,2,\dots,m$ 。或者至少有一个为标量时, 方可进行

乘法操作, 否则将返回一出错信息。

$A.*B$ 符号数组的乘法。

$A.*B$ 为按参量 A 与 B 对应的分量进行相乘。 A 与 B 必须为同型阵列, 或至少有一个为标量。即: $A_{n \times m}.*B_{n \times m} = (a_{ij})_{n \times m}.*(b_{ij})_{n \times m} = C_{n \times m} = (c_{ij})_{n \times m}$, 则 $c_{ij} = a_{ij} * b_{ij}$, $i=1,2,\dots,n$; $j=1,2,\dots,m$ 。

$A \setminus B$ 矩阵的左除法。

$X = A \setminus B$ 为符号线性方程组 $A*X=B$ 的解。我们指出的是, $A \setminus B$ 近似地等于 $\text{inv}(A)*B$ 。若 X 不存在或者不唯一, 则产生一警告信息。矩阵 A 可以是矩形矩阵 (即非正方形矩阵), 但此时要求方程组必须是相容的。

$A \setminus B$ 数组的左除法。

$A \setminus B$ 为按对应的分量进行相除。若 A 与 B 为同型阵列时, $A_{n \times m} \setminus B_{n \times m} = (a_{ij})_{n \times m} \setminus (b_{ij})_{n \times m} = C_{n \times m} = (c_{ij})_{n \times m}$, 则 $c_{ij} = a_{ij} \setminus b_{ij}$, $i=1,2,\dots,n$; $j=1,2,\dots,m$ 。若 A 与 B 中至少有一个为标量, 则把标量扩大为与另外一个同型的阵列, 再按对应的分量进行操作。

A/B 矩阵的右除法。

$X = B/A$ 为符号线性方程组 $X*A=B$ 的解。我们指出的是, B/A 粗略地等于 $B*\text{inv}(A)$ 。若 X 不存在或者不唯一, 则产生一警告信息。矩阵 A 可以是矩形矩阵 (即非正方形矩阵), 但此时要求方程组必须是相容的。

$A./B$ 数组的右除法。

$A./B$ 为按对应的分量进行相除。若 A 与 B 为同型阵列时,



$A_{n \times m} / B_{n \times m} = (a_{ij})_{n \times m} / (b_{ij})_{n \times m} = C_{n \times m} = (c_{ij})_{n \times m}$, 则 $c_{ij} = a_{ij} / b_{ij}$, $i=1,2,\dots,n$; $j=1,2,\dots,m$ 。
若 A 与 B 中至少有一个为标量, 则把标量扩大为与另外一个同型的阵列, 再按对应的分量进行操作。

A^B 矩阵的方幂。

计算矩阵 A 的整数 B 次方幂。若 A 为标量而 B 为方阵, A^B 用方阵 B 的特征值与特征向量计算数值。若 A 与 B 同时为矩阵, 则返回一错误信息。

$A.^B$ 数组的方幂。

$A.^B$ 为按 A 与 B 对应的分量进行方幂计算。若 A 与 B 为同型阵列时, $A_{n \times m} .^ B_{n \times m} = (a_{ij})_{n \times m} .^ (b_{ij})_{n \times m} = C_{n \times m} = (c_{ij})_{n \times m}$, 则 $c_{ij} = a_{ij} ^ b_{ij}$, $i=1,2,\dots,n$; $j=1,2,\dots,m$ 。
若 A 与 B 中至少有一个为标量, 则把标量扩大为与另外一个同型的阵列, 再按对应的分量进行操作。

A' 矩阵的 Hermition 转置。

若 A 为复数矩阵, 则 A' 为复数矩阵的共轭转置。即, 若 $A = (a_{ij}) = (x_{ij} + i * y_{ij})$, 则 $A' = (a'_{ji}) = (\overline{a_{ij}}) = (x_{ij} - i * y_{ij})$ 。

$A.'$ 数组转置。

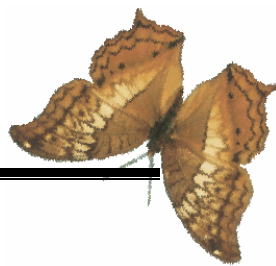
$A.'$ 为真正的矩阵转置, 其没有进行共轭转置。

例 3-1

```
>>syms a b c d e f g h;
>>A = [a b; c d];
>>B = [e f; g h];
>>C1 = A.*B
>>C2 = A.^B
>>C3 = A*B/A
>>C4 = A.*A-A^2
>>syms a11 a12 a21 a22 b1 b2;
>>A = [a11 a12; a21 a22];
>>B = [b1 b2];
>>X = B/A; % 求解符号线性方程组 X*A=B 的解
>>x1 = X(1)
>>x2 = X(2)
```

计算结果为:

```
C1 =
[ a*e, b*f]
[ c*g, d*h]
C2 =
[ a^e, b^f]
[ c^g, d^h]
C3 =
[ -(a*c*f+c*b*h-a*e*d-b*d*g)/(a*d-b*c), (a*b*h-b^2*g+a^2*f-b*a*e)/(a*d-b*c)]
[ -(c*e*d+c*d*h+c^2*f-d^2*g)/(a*d-b*c), (a*d*h+a*c*f-b*c*e-b*d*g)/(a*d-b*c)]
C4 =
[ -b*c, b^2-a*b-b*d]
[ c^2-a*c-d*c, -b*c]
x1 =
(-a22*b1+b2*a21)/(a12*a21-a11*a22)
x2 =
-(a12*b1+a11*b2)/(a12*a21-a11*a22)
```



3.2 基本运算

命令 1 合并同类项

函数 `collect`

格式 `R = collect(S)` %对于多项式 S 中的每一函数, `collect(S)`按缺省变量 x 的次数合并系数。

`R = collect(S,v)` %对指定的变量 v 计算, 操作同上。

例 3-2

```
>>syms x y;
>>R1 = collect((exp(x)+x)*(x+2))
>>R2 = collect((x+y)*(x^2+y^2+1), y)
>>R3 = collect([(x+1)*(y+1),x+y])
```

计算结果为:

```
R1 =
x^2+(exp(x)+2)*x+2*exp(x)
R2 =
y^3+x*y^2+(x^2+1)*y+x*(x^2+1)
R3 =
[(y+1)*x+y+1, x+y]
```

命令 2 列空间的基

函数 `colspace`

格式 `B = colspace(A)` %返回矩阵 B , 其列向量形成由矩阵 A 的列向量形成的空间的坐标基, 其中 A 可以是符号或数值矩阵。而 `size(colspace(A),2)`等于 `rank(A)`。即由 A 生成的空间维数等于 A 的秩。

例 3-3

```
>>syms a b c
>>A = sym([1,a;2,b;3,c])
>>B = colspace(A)
```

计算结果为:

```
A =
[ 1, a]
[ 2, b]
[ 3, c]
B =
[ 1, 0]
[ 0, 1]
[ -(3*b-2*c)/(-b+2*a), (-c+3*a)/(-b+2*a)]
```

命令 3 复合函数计算

函数 `compose`

格式 `compose(f,g)` %返回复合函数 $f[g(y)]$, 其中 $f=f(x)$, $g=g(y)$ 。其中符号 x 为函数 f 中由命令 `findsym(f)` 确定的符号变量, 符号 y 为函数 g 中由命令 `findsym(g)` 确定的符号变量。

`compose(f,g,z)` %返回复合函数 $f[g(z)]$, 其中 $f=f(x)$, $g=g(y)$, 符号 x 、 y 为函数 f 、 g 中由命令 `findsym` 确定的符号变量。



`compose(f,g,x,z)` %返回复合函数 $f[g(z)]$, 而令变量 x 为函数 f 中的自变量 $f=f(x)$ 。
令 $x=g(z)$, 再将 $x=g(z)$ 代入函数 f 中。

`compose(f,g,x,y,z)` %返回复合函数 $f[g(z)]$ 。而令变量 x 为函数 f 中的自变量 $f=f(x)$, 而令变量 y 为函数 g 中的自变量 $g=g(y)$ 。令 $x=g(y)$, 再将 $x=g(y)$ 代入函数 $f=f(x)$ 中, 得 $f[g(y)]$, 最后用指定的变量 z 代替变量 y , 得 $f[g(z)]$ 。

例 3-4

```
>>syms x y z t u v;
>>f = 1/(1 + x^2*y); h = x^t; g = sin(y); p = sqrt(-y/u);
>>C1 = compose(f,g) % 令 x=g=sin(y), 再替换 f 中的变量 x=findsym(f)。
>>C2 = compose(f,g,t) % 令 x=g=sin(t), 再替换 f 中的变量 x=findsym(f)。
>>C3 = compose(h,g,x,z) % 令 x=g=sin(z), 再替换 h 中的变量 x。
>>C4 = compose(h,g,t,z) % 令 t=g=sin(z), 再替换 h 中的变量 t。
>>C5 = compose(h,p,x,y,z) % 令 x=p(y)=sqrt(-y/u), 替换 h 中的变量 x, 再将 y 换成 z。
>>C6 = compose(h,p,t,u,z) % 令 t=p(u)=sqrt(-y/u), 替换 h 中的变量 t, 再将 u 换成 z。
```

计算结果为:

```
C1 =
1/(1+sin(y)^2*y)
C2 =
1/(1+sin(t)^2*y)
C3 =
sin(z)^t
C4 =
x^sin(z)
C5 =
((-z/u)^(1/2))^t
C6 =
```

```
x^((-y/z)^(1/2))
```

命令 4 符号复数的共轭

函数 **conj**

格式 `conj(X)` %返回符号复数 X 的共轭复数

例 3-5

$X=\text{real}(X)+i*\text{imag}(X)$, 则 $\text{conj}(X)=\text{real}(X)-i*\text{imag}(X)$

命令 5 符号复数的实数部分

函数 **real**

格式 `real(Z)` %返回符号复数 z 的实数部分

命令 6 符号复数的虚数部分

函数 **imag**

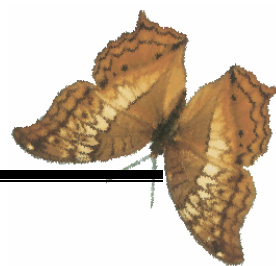
格式 `imag(Z)` %返回符号复数 z 的虚数部分

命令 7 余弦函数的整函数

格式 `Y = cosint(X)` %计算余弦函数在点 X 处的整函数值。其中 X 可以是数值矩阵,

或符号矩阵。余弦函数的整函数定义为: $Y_i = \gamma + \ln X_i + \int_0^{X_i} \frac{\cos t - 1}{t} dt$, 其中 γ 为

Euler 常数, $\gamma=0.57721566490153286060651209\cdots$ $i=1,2,\dots,\text{size}(X)$ 。Euler 常数可以通过命令 `vpa('eulergamma')` 获得。



例 3-6

```
>>cosint(7.2)
>>cosint([0:0.1:1])
>>syms x;
>>f=cosint(x);
>>diff(x)
```

计算结果为:

```
ans =
    0.0960
ans =
Columns 1 through 7
    Inf    -1.7279    -1.0422    -0.6492    -0.3788    -0.1778    -0.0223
Columns 8 through 11
    0.1005    0.1983    0.2761    0.3374
ans =
    1
```

命令 8 设置变量的精度

函数 **digits**

格式 **digits(d)** %设置当前的可变算术精度的位数为整数 d 位

d = digits %返回当前的可变算术精度位数给 d

digits %显示当前可变算术精度的位数

说明 设置有意义的十进制数值的、在 Maple 软件中用于做可变算术精度(命令为: **vpa**)计算的数字位数。其缺省值为 32 位数字。

例 3-7

```
>>z = 1.0e-16 % z 为一很小的数
>>x = 1.0e+2 % x 为较大的数
>>digits(14)
>>y1 = vpa(x*z+1) % 大数 1 “吃掉” 小数 x*y
>>digits(15)
>>y2 = vpa(x*z+1) % 防止 “去掉” 小数 x*y
```

计算结果为:

```
z =
    1.0000e-016
x =
    100
y1 =
    1.000000000000000
y2 =
    1.000000000000001
```

命令 9 将符号转换为 MATLAB 的数值形式

函数 **double**

格式 **R = double(S)** %将符号对象 S 转换为数值对象 R。若 S 为符号常数或表达式常数, **double** 返回 S 的双精度浮点数值表示形式; 若 S 为每一元素是符号常数或表达式常数的符号矩阵, **double** 返回 S 每一元素的双精度浮点数值表示的数值矩阵 R。

例 3-8

```
>>gold_ratio = double(sym('(sqrt(5)-1)/2')) % 计算黄金分割率。
```



```
>>T = sym(hilb(4))
```

```
>>R = double(T)
```

计算结果为:

```
gold_ratio =  
0.6180  
  
T =  
[ 1, 1/2, 1/3, 1/4]  
[ 1/2, 1/3, 1/4, 1/5]  
[ 1/3, 1/4, 1/5, 1/6]  
[ 1/4, 1/5, 1/6, 1/7]  
  
R =  
1.0000    0.5000    0.3333    0.2500  
0.5000    0.3333    0.2500    0.2000  
0.3333    0.2500    0.2000    0.1667  
0.2500    0.2000    0.1667    0.1429
```

命令 10 符号表达式的展开

函数 **expand**

格式 **R = expand(S)** %对符号表达式 S 中每个因式的乘积进行展开计算。该命令通常用于计算多项式函数、三角函数、指数函数与对数函数等表达式的展开式。

例 3-9

```
>>syms x y a b c t  
>>E1 = expand((x-2)*(x-4)*(y-t))  
>>E2 = expand(cos(x+y))  
>>E3 = expand(exp((a+b)^3))  
>>E4 = expand(log(a*b/sqrt(c)))  
>>E5 = expand([sin(2*t), cos(2*t)])
```

计算结果为:

```
E1 =  
x^2*y-x^2*t-6*x*y+6*x*t+8*y-8*t  
E2 =  
cos(x)*cos(y)-sin(x)*sin(y)  
E3 =  
exp(a^3)*exp(a^2*b)^3*exp(a*b^2)^3*exp(b^3)  
E4 =  
log(a*b/c^(1/2))  
E5 =  
[ 2*sin(t)*cos(t), 2*cos(t)^2-1]
```

命令 11 符号因式分解

函数 **factor**

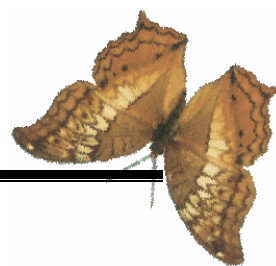
格式 **factor(X)** %参量 x 可以是正整数、符号表达式阵列或符号整数阵列。若 X 为一正整数，则 factor(X)返回 X 的质数分解式。若 x 为多项式或整数矩阵，则 factor(X)分解矩阵的每一元素。若整数阵列中有一元素位数超过 16 位，用户必须用命令 sym 生成该元素。

例 3-10

```
>>syms a b x y  
>>F1 = factor(x^4-y^4)  
>>F2 = factor([a^2-b^2, x^3+y^3])  
>>F3 = factor(sym('12345678901234567890'))
```

计算结果为:





```
F1 =
      (x-y)*(x+y)*(x^2+y^2)
F2 =
      [(a-b)*(a+b), (x+y)*(x^2-x*y+y^2)]
F3 =
      (2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)
```

命令 12 符号表达式的分子与分母

函数 **numden**

格式 **[N,D] = numden(A)**

说明 将符号或数值矩阵 **A** 中的每一元素转换成整系数多项式的有理式形式，其中分子与分母是相对互素的。输出的参量 **N** 为分子的符号矩阵，输出的参量 **D** 为分母的符号矩阵。

例 3-11

```
>>syms x y a b c d;
>>[n1,d1] = numden(sym(sin(4/5)))
>>[n2,d2] = numden(x/y + y/x)
>>A = [a, 1/b;1/c d];
>>[n3,d3] = numden(A)
```

计算结果为：

```
n1 =
      6461369247334093
d1 =
      9007199254740992
n2 =
      x^2+y^2
d2 =
      y*x
n3 =
      [ a, 1]
      [ 1, d]
d3 =
      [ 1, b]
      [ c, 1]
```

命令 13 搜索符号表达式的最简形式

函数 **simple**

格式 **r = simple(S)** %该命令试图找出符号表达式 **S** 的代数上的简单形式，显示任意的能使表达式 **S** 长度变短的表达式，且返回其中最短的一个。若 **S** 为一矩阵，则结果为整个矩阵的最短形式，而非是每一个元素的最简形式。若没有输出参量 **r**，则该命令将显示所有可能使用的算法与表达式，同时返回最短的一个。

[r,how] = simple(S) %没有显示中间的化简结果，但返回能找到的最短的一个。

输出参量 **r** 为一符号，**how** 为一字符串，用于表示算法。

例 3-12

```
>>syms x
>>R1 = simple(cos(x)^4+sin(x)^4)
>>R2 = simple(2*cos(x)^2-sin(x)^2)
>>R3 = simple(cos(x)^2-sin(x)^2)
```



```
>>R4 = simple(cos(x)+(-sin(x)^2)^(1/2))
>>R5 = simple(cos(x)+i*sin(x))
>>R6 = simple((x+1)*x*(x-1))
>>R7 = simple(x^3+3*x^2+3*x+1)
>> [R8,how] = simple(cos(3*acos(x)))
```

计算的结果为：

```
R1 =
    1/4*cos(4*x)+3/4
R2 =
    3*cos(x)^2-1
R3 =
    cos(2*x)
R4 =
    cos(x)+i*sin(x)
R5 =
    exp(i*x)
R6 =
    x^3-x
R7 =
    (x+1)^3
R8 =
    4*x^3-3*x
how =
```

~~expand~~

命令 14 符号表达式的化简

函数 **simplify**

格式 **R = simplify(S)**

说明 使用 Maple 软件中的化简规则，将化简符号矩阵 S 中每一元素。

例 3-13

```
>>syms x a b c
>>R1 = simplify(sin(x)^4 + cos(x)^4)
>>R2 = simplify(exp(c*log(sqrt(a+b))))
>>S = [(x^2+5*x+6)/(x+2),sqrt(16)];
>>R3 = simplify(S)
```

计算结果为：

```
R1 =
    2*cos(x)^4+1-2*cos(x)^2
R2 =
    (a+b)^(1/2*c)
R3 =
    [ x+3, 4]
```

命令 15 符号矩阵的维数

函数 **size**

格式 **d = size(A)**

%若 A 为 m*n 阶的符号矩阵，则输出结果 d=[m, n]。

[m,n] = size(A)

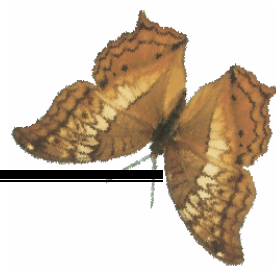
%分别返回矩阵 A 的行数于 m，列数于 n。

d = size(A, n)

%返回由标量 n 指定的 A 的方向的维数：n=1 为行方向，n=2 为列方向。

例 3-14

```
>>syms a b c d
```

```
>>A=[a b c; a b d; d c b; c b a];
>>d=size(A)
>>r=size(A, 2)
```

计算结果为:

```
d =
     4     3
r =
     3
```

命令 16 代数方程的符号解析解

函数 **solve**

格式 **g = solve(eq)** %输入参量 **eq** 可以是符号表达式或字符串。若 **eq** 是一符号表达式 $x^2-2*x-1$ 或一没有等号的字符串 $'x^2-2*x-1'$, 则 **solve(eq)** 对方程 **eq** 中的缺省变量(由命令 **findsym(eq)** 确定的变量)求解方程 **eq=0**。若输出参量 **g** 为单一变量, 则对于有多重解的非线性方程, **g** 为一行向量。

g = solve(eq,var) %对符号表达式或没有等号的字符串 **eq** 中指定的变量 **var** 求解方程 **eq(var)=0**。

g = solve(eq1,eq2,...,eqn) %输入参量 **eq1,eq2,...,eqn** 可以是符号表达式或字符串。该命令对方程组 **eq1,eq2,...,eqn** 中由命令 **findsym** 确定的 **n** 个变量如 **x1,x2,...,xn** 求解。若 **g** 为一单个变量, 则 **g** 为一包含 **n** 个解的结构; 若 **g** 为有 **n** 个变量的向量, 则分别返回结果给相应的变量。

g = solve(eq1,eq2,...,eqn,var1,var2,...,varn) %对方程组 **eq1,eq2,...,eqn** 中指定的 **n** 个变量如 **var1,var2,...,varn** 求解。

注意: 对于单个的方程或方程组, 若不存在符号解, 则返回方程(组)的数值解。

例 3-15

```
>>solve('a*x^2 + b*x + c')
>>solve('a*x^2 + b*x + c','b')
>>solve('x + y = 1','x - 11*y = 5')
>>A = solve('a*u^2 + v^2','u - v = 1','a^2 - 5*a + 6')
```

计算结果为:

```
ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
ans =
-(a*x^2+c)/x
ans =
x: [1x1 sym]
y: [1x1 sym]
A =
a: [4x1 sym]
u: [4x1 sym]
v: [4x1 sym]
```

命令 17 以共同的子表达式形式重写一符号表达式

函数 **subexpr**

格式 **[Y,SIGMA] = subexpr(X,SIGMA)**



```
[Y,SIGMA] = subexpr(X,'SIGMA')
```

说明 找出符号表达式 X 中相同的子表达式，再结合命令 `pretty(X)` 将 X 中相同的、比较复杂的子字符串用符号 `%1,%2,...` 代替。而用命令 `pretty(Y)` 将 X 中相同的、比较复杂的子字符串用符号 `SIGMA` 代替。

例 3-16

```
>>t = solve('a*x^3+b*x^2+c*x+d = 0');
>> [r,s] = subexpr(t,'s');
>>pretty(t)
>>pretty(r)
```

计算结果为：(略)

命令 18 特征多项式

函数 `poly`

格式 `p = poly(A)` 或 `p = poly(A, v)`

说明 若 A 为一数值阵列，则返回矩阵 A 的特征多项式的系数，且有：命令 `poly(sym(A))` 近似等于 `poly2sym(poly(A))`。其近似程度取决于舍入误差的大小。若 A 为一符号矩阵，则返回矩阵 A 的变量为 x 的特征多项式。若带上参量 v ，则返回变量为 v 的特征多项式。

例 3-17

```
>>A = hilb(4);
>>p = poly(A)
>>q = poly(sym(A))
>>s = poly(sym(A),z)
```

计算结果为：

```
p =
    1.0000    -1.6762     0.2652    -0.0017     0.0000
q =
x^4-176/105*x^3+3341/12600*x^2-41/23625*x+1/6048000
s =
176/105*z^3+3341/12600*z^2-41/23625*z+1/6048000+z^4
```

命令 19 将多项式系数向量转化为带符号变量的多项式

函数 `poly2sym`

格式 `r = poly2sym(c)` 和 `r = poly2sym(c, v)`

说明 将系数在数值向量 c 中的多项式转化成相应的带符号变量的多项式（按次数的降幂排列）。缺省的符号变量为 x ；

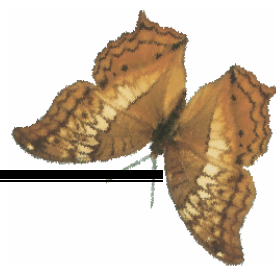
若带上参量 v ，则符号变量用 v 显示。`poly2sym` 使用命令 `sym` 的缺省转换模式（有理形式）将数值型系数转换为符号常数。该模式将数值转换成接近的整数比值的表达式，否则用 2 的幂指数表示。若 x 有一数值值，且命令 `sym` 能将 c 的元素精确表示，则 `eval(poly2sym(c))` 的结果与 `polyval(c,x)` 相同。

例 3-18

```
>>r1 = poly2sym([1 2 3 4])
>>r2 = poly2sym([.694228, sqrt(2), sin(pi/3)])
>>r3 = poly2sym([1 0 1 -1 2], y)
```

计算结果为：

```
r1 =
x^3+2*x^2+3*x+4
```



```
r2 =
6253049924220329/9007199254740992*x^2+x*2^(1/2)+1/2*3^(1/2)
r3 =
y^4+y^2-y+2
```

命令 20 将复杂的符号表达式显示成我们习惯的数学书写形式

函数 **pretty**

格式 **pretty(S)** %用缺省的线型宽度 79 显示符号矩阵 s 中每一元素

pretty(S,n) %用指定的线型宽度 n 显示

例 3-19

```
>>A = sym(pascal(3));
>>B = eig(A)
>>pretty(B,50) % 多看几次结果，会发现该命令显示的特点
>>syms x
>>y=log(x)/sqrt(x);
>>dy = diff(y)
>>pretty(dy)
```

计算结果为：

```
B =
[      1]
[ 4+15^(1/2)]
[ 4 -15^(1/2)]
[      1]
[      ]
[      1/2]
[4 + 15  ]
[      ]
[      1/2]
[4 - 15  ]
dy =
1/x^(3/2)-1/2*log(x)/x^(3/2)
1      log(x~)
---- - 1/2  -----
3/2      3/2
```

命令 21 从一符号表达式中或矩阵中找出符号变量

函数 **findsym**

格式 **r = findsym(S)** %以字母表的顺序返回表达式 S 中的所有符号变量（注：符号变量为由字母（除了 i 与 j）与数字构成的、字母打头的字符串）。若 S 中没有任何的符号变量，则 **findsym** 返回一空字符串。

r = findsym(S,n) %返回字母表中接近 x 的 n 个符号变量

例 3-20

```
>>syms a x y z t alpha beta
>>1 = findsym(sin(pi*t*alpha+beta))
>>S2 = findsym(x+i*y-j*z+eps-nan)
>>S3 = findsym(a+y,pi)
```

计算结果为：

```
S1 =
pi, alpha, beta, t
S2 =
```



NaN, x, y, z
S3 =

a, y

命令 22 函数的反函数

函数 finverse

格式 $g = \text{finverse}(f)$ %返回函数 f 的反函数。其中 f 为单值的一元数学函数, 如 $f=f(x)$ 。
若 f 的反函数存在, 设为 g , 则有 $g[f(x)] = x$ 。

$g = \text{finverse}(f,u)$ %若符号函数 f 中有几个符号变量时, 对指定的符号自变量 v 计算其反函数。若其反函数存在, 设为 g , 则有 $g[f(v)] = v$ 。

例 3-21

```
>>syms x p q u v;
>>V1 = finverse(1/((x^2+p)*(x^2+q)))
>>V2 = finverse(exp(u-2*v),u)
```

计算结果为:

Warning: finverse(1/(x^2+p)/(x^2+q)) is not unique.

> In D:\MATLABR12\toolbox\symbolic\@sym\finverse.m at line 43

V1 =

$1/2/x^{1/2}*(x*(-x*q-x*p+(x^2*q^2-2*x^2*q*p+x^2*p^2+4*x)^{1/2}))^{1/2}$

V2 =

$2*v+\log(u)$

命令 23 嵌套形式的多项式的表达式

函数 horner

格式 $R = \text{horner}(P)$ %若 P 为一符号多项式的矩阵, 该命令将矩阵的每一元素转换成嵌套形式的表达式 R 。

例 3-22

```
>>syms x y
>>H1 = horner(2*x^4-6*x^3+9*x^2-6*x-4)
>>H2 = horner([x^2+x*y;y^3-2*y])
```

计算结果为:

H1 =

$-4+(-6+(9+(-6+2*x)*x)*x)*x$

H2 =

$[x^2+x*y]$

$[(-2+y^2)*y]$

命令 24 符号表达式求和

函数 symsum

格式 $r = \text{symsum}(s)$ %对符号表达式 s 中的符号变量 k (由命令 $\text{findsym}(s)$ 确定的) 从 0 到 $k-1$ 求和

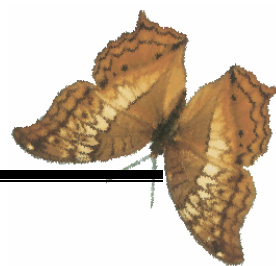
$r = \text{symsum}(s,v)$ %对符号表达式 s 中指定的符号变量 v 从 0 到 $v-1$ 求和

$r = \text{symsum}(s,a,b)$ %对符号表达式 s 中的符号变量 k (由命令 $\text{findsym}(s)$ 确定的) 从 a 到 b 求和

$r = \text{symsum}(s,v,a,b)$ %对符号表达式 s 中指定的符号变量 v 从 a 到 b 求和

例 3-23

```
>>syms k n x
>>r1 = symsum(k^3)
```



```
>>r2 = symsum(k^2-k)
>>r3 = symsum(sin(k*pi)/k,0,n)
>>r4 = symsum(k^2,0,10)
>>r5 = symsum(x^k/sym('k!'), k, 0,inf) %为使 k!通过 MATLAB 表达式的检验,必须把它作为一符号表达式。
```

计算结果为:

```
r1 =
    1/4*k^4-1/2*k^3+1/4*k^2
r2 =
    1/3*k^3-k^2+2/3*k
r3 =
    -1/2*sin(k*(n+1))/k+1/2*sin(k)/k/(cos(k)-1)*cos(k*(n+1))-1/2*sin(k)/k/(cos(k)-1)
r4 =
    385
r5 =
    exp(x)
```

命令 25 广义超几何函数

函数 **hypergeom**

格式 **hypergeom(n, d, z)** %该命令为广义超几何函数 $F(n,d,z)$, 即已知的 Barnes 扩展超几何函数, 记做 ${}_jF_k$, 其中 $j=\text{length}(n)$, $k=\text{length}(d)$ 。对于标量 a,b 与 c , **hypergeom([a,b],c, z)** 为 Gauss 超几何函数 ${}_2F_1(a,b;c,z)$ 。

说明 超几何函数的定义为: $F(n,d,z) = \sum_{k=1}^{\infty} \frac{C_{n,k}}{C_{d,k}} * \frac{z^k}{k!}$, 其中 $C_{v,k} = \prod_{j=1}^{|v|} \frac{\Gamma(v_j + k)}{\Gamma(v_j)}$

例 3-24

```
>>syms a z n
>>H1 = hypergeom([],[],z)
>>H2 = hypergeom(1,[],z)
>>H3 = hypergeom(1,2,'z')
>>H4 = hypergeom([1,2],[2,3],'z')
>>H5 = hypergeom(a,[],z)
>>H6 = hypergeom([],1,-z^2/4)
>>H7 = hypergeom([-n, n],1/2,(1-z)/2)
```

计算结果为:

```
H1 =
    exp(z)
H2 =
    -1/(-1+z)
H3 =
    (exp(z)-1)/z
H4 =
    -2*(-exp(z)+1+z)/z^2
H5 =
    (1-z)^(-a)
H6 =
    besselj(0,z)
H7 =
    hypergeom([n, -n],[1/2],1/2-1/2*z)
```



3.2.1 函数计算器

函数 funtool

格式 funtool %该命令将生成三个图形窗口，Figure No.1 用于显示函数 f 的图形，Figure No.2 用于显示函数 g 的图形，Figure No.3 为一可视化的、可操作与显示一元函数的计算器界面。在该界面上由许多按钮，可以显示两个由用户输入的函数的计算结果：加、乘、微分等。funtool 还有一函数存储器，允许用户将函数存入，以便后面调用。在开始时，funtool 显示两个函数 $f(x) = x$ 与 $g(x) = 1$ 在区间 $[-2\pi, 2\pi]$ 上的图形。Funtool 同时下面显示一控制面板，允许用户对函数 f 、 g 进行保存、更正、重新输入、联合与转换等操作。

输入命令 funtool 后，生成的界面如下：

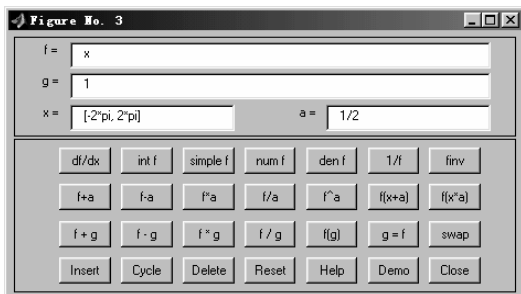


图 3-1 函数工具 funtool 界面

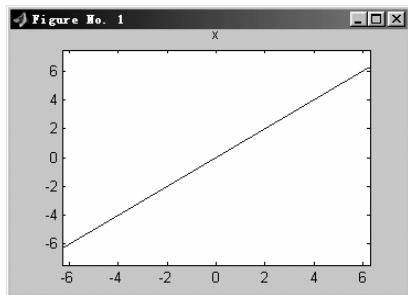


图 3-2 函数 f 的图形

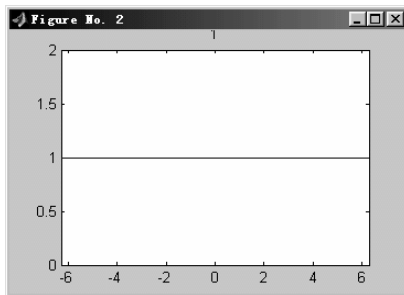


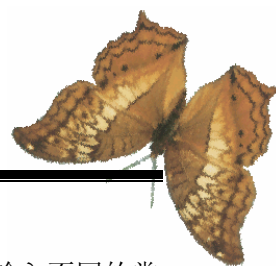
图 3-3 函数 g 的图形

说明 文本输入框区域：控制面板的上面几行，可以输入文本；

$f =$: 显示代表函数 f 的符号表达式，可在该行输入其他有效的表达式来定义 f ，再按回车键即可在 Figure No.1 中画出图形；

$g =$: 显示代表函数 g 的符号表达式，可在该行输入其他有效的表达式来定义 g ，再按回车键即可在 Figure No.2 中画出 g 图形；

$x =$: 显示用于画函数 f 与 g 的区间。可在该行输入其他的不同区间，再按回车键即可



改变 Figure No.1 与 Figure No.2 中的区间;

a = : 显示一用于改变函数 f 的常量因子(见下面的操作按钮)。可在该行输入不同的常数。

控制按钮区域: 该区域有一些按钮, 按下它们将对函数 f 转换成不同的形式与执行不同的操作。

df/dx: 函数 f 的导数;

int f: 函数 f 的积分(没有常数的一个原函数), 当函数 f 的原函数不能用初等函数表示时, 操作可能失败;

simple f: 化简函数 f (若有可能);

num f: 函数 f 的分子;

den f: 函数 f 的分母;

1/f: 函数 f 的倒数;

finv: 函数 f 的反函数, 若函数 f 的反函数不存在, 操作可能失败;

f+a: 用 $f(x)+a$ 代替函数 $f(x)$;

f-a: 用 $f(x)-a$ 代替函数 $f(x)$;

f*a: 用 $f(x)*a$ 代替函数 $f(x)$;

f/a: 用 $f(x)/a$ 代替函数 $f(x)$;

f^a: 用 $f(x)^a$ 代替函数 $f(x)$;

f(x+a): 用 $f(x+a)$ 代替函数 $f(x)$;

f(x*a): 用 $f(x-a)$ 代替函数 $f(x)$;

f+g: 用 $f(x)+g(x)$ 代替函数 $f(x)$;

f-g: 用 $f(x)-g(x)$ 代替函数 $f(x)$;

f*g: 用 $f(x)*g(x)$ 代替函数 $f(x)$;

f/g: 用 $f(x)/g(x)$ 代替函数 $f(x)$;

g=f: 用函数 $f(x)$ 代替函数 $g(x)$;

swap: 函数 $f(x)$ 与 $g(x)$ 互换;

Insert: 将函数 $f(x)$ 保存到函数内存列表中的最后;

Cycle: 用内存函数列表中的第二项代替函数 $f(x)$;

Delete: 从内存函数列表中删除函数 $f(x)$;

Reset: 重新设置计算器为初始状态;

Help: 显示在线的关于计算器的帮助;

Demo: 运行该计算器的演示程序;

Close: 关闭计算器的三个窗口。

3.2.2 微积分

命令 1 极限

函数 **limit**

格式 **limit(F,x,a)** %计算符号表达式 $F=F(x)$ 的极限值, 当 $x \rightarrow a$ 时。

limit(F,a) %用命令 **findsym(F)** 确定 F 中的自变量, 设为变量 x , 再计算 F 的



极限值, 当 $x \rightarrow a$ 时。

`limit(F)` %用命令 `findsym(F)` 确定 F 中的自变量, 设为变量 x , 再计算 F 的极限值, 当 $x \rightarrow 0$ 时。

`limit(F,x,a,'right')` 或 `limit(F,x,a,'left')` %计算符号函数 F 的单侧极限: 左极限 $x \rightarrow a^-$ 或右极限 $x \rightarrow a^+$ 。

例 3-25

```
>>syms x a t h n;
>>L1 = limit((cos(x)-1)/x)
>>L2 = limit(1/x^2,x,0,'right')
>>L3 = limit(1/x,x,0,'left')
>>L4 = limit((log(x+h)-log(x))/h,h,0)
>>v = [(1+a/x)^x, exp(-x)];
>>L5 = limit(v,x,inf,'left')
>>L6 = limit((1+2/n)^(3*n),n,inf)
```

计算结果为:

```
L1 =
    0
L2 =
    inf
L3 =
   -inf
L4 =
    1/x
L5 =
 [ exp(a),      0]
L6 =
    exp(6)
```

命令 2 导数 (包括偏导数)

函数 `diff`

格式 `diff(S,'v')`、`diff(S,sym('v'))` %对表达式 S 中指定符号变量 v 计算 S 的 1 阶导数。

`diff(S)` %对表达式 S 中的符号变量 v 计算 S 的 1 阶导数, 其中 $v=\text{findsym}(S)$ 。

`diff(S,n)` %对表达式 S 中的符号变量 v 计算 S 的 n 阶导数, 其中 $v=\text{findsym}(S)$ 。

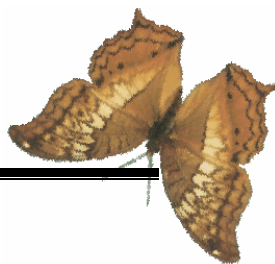
`diff(S,'v',n)` %对表达式 S 中指定的符号变量 v 计算 S 的 n 阶导数。

例 3-26

```
>>syms x y t
>>D1 = diff(sin(x^2)*y^2,2)      %计算  $\frac{\partial^2}{\partial x^2} y^2 \sin x^2$ 
>>D2 = diff(D1,y)      %计算  $\frac{\partial}{\partial y} \left( \frac{\partial^2}{\partial x^2} y^2 \sin x^2 \right)$ 
>>D3 = diff(t^6,6)
```

计算结果为:

```
D1 =
-4*sin(x^2)*x^2*y^2+2*cos(x^2)*y^2
D2 =
-8*sin(x^2)*x^2*y+4*cos(x^2)*y
D3 =
720
```


**命令 3 符号函数的积分****函数** `int`

格式 `R = int(S,v)` %对符号表达式 S 中指定的符号变量 v 计算不定积分。注意的是，表达式 R 只是函数 S 的一个原函数，后面没有带任意常数 C 。

`R = int(S)` %对符号表达式 S 中的符号变量 v 计算不定积分，其中 $v = \text{findsym}(S)$ 。

`R = int(S,v,a,b)` %对表达式 s 中指定的符号变量 v 计算从 a 到 b 的定积分

`R = int(S,a,b)` %对符号表达式 s 中的符号变量 v 计算从 a 到 b 的定积分，其中 $v = \text{findsym}(S)$ 。

例 3-27

```
>>syms x z t alpha
>>INT1 = int(-2*x/(1+x^3)^2)
>>INT2 = int(x/(1+z^2),z)
>>INT3 = int(INT2,x)
>>INT4 = int(x*log(1+x),0,1)
>>INT5 = int(2*x, sin(t), 1)
>>INT6 = int([exp(t),exp(alpha*t)])
```

计算结果为：

```
INT1 =
    -2/9/(x+1)+2/9*log(x+1)-1/9*log(x^2-x+1)-2/9*3^(1/2)*atan(1/3*(2*x-1)*...
    3^(1/2))-2/9*(2*x-1)/(x^2-x+1)
INT2 =
    x*atan(z)
INT3 =
    1/2*x^2*atan(z)
INT4 =
    1/4
INT5 =
    1-sin(t)^2
INT6 =
    [ exp(t), 1/alpha*exp(alpha*t)]
```

命令 4 常微分方程的符号解**函数** `dsolve`

格式 `r = dsolve('eq1,eq2,...','cond1,cond2,...','v')`

说明 对给定的常微分方程(组) $\text{eq1}, \text{eq2}, \dots$ 中指定的符号自变量 v ，与给定的边界条件和初始条件 $\text{cond1}, \text{cond2}, \dots$ 求符号解(即解析解) r ；若没有指定变量 v ，则缺省变量为 t ；在微分方程(组)的表达式 eq 中，大写字母 D 表示对自变量(设为 x)的微分算子： $D = d/dx$ ， $D2 = d^2/dx^2$ ，...。微分算子 D 后面的字母则表示为因变量，即待求解的未知函数。初始和边界条件由字符串表示： $y(a)=b$ ， $Dy(c)=d$ ， $D2y(e)=f$ ，等等，分别表示 $y(x)|_{x=a} = b$ ， $y'(x)|_{x=c} = d$ ， $y''(x)|_{x=e} = f$ ；若边界条件少于方程(组)的阶数，则返回的结果 r 中会出现任意常数 $C1, C2, \dots$ ；`dsolve` 命令最多可以接受 12 个输入参量(包括方程组与定解条件个数，当然我们可以做到输入的方程个数多于 12 个，只要将多个方程置于一字符串内即可)。若没有给定输出参量，则在命令窗口显示解列表。若该命令找不到解析解，则返回一警告信息，同时返回一空的 `sym` 对象。这时，用户可以用命令 `ode23` 或 `ode45` 求解方程组的数值解。

例 3-28

```
>>D1 = dsolve('D2y - Dy = exp(x)')
```



```

>>D2 = dsolve('t*D2f = Df*log((Dy)/t)')
>>D3 = dsolve('(Dy)^2 + y^2 = 1','s')
>>D4 = dsolve('Dy = a*y', 'y(0) = b') % 带一个定解条件
>>D5 = dsolve('D2y = -a^2*y', 'y(0) = 1', 'Dy(pi/a) = 0') % 带两个定解条件
>>[x,y] = dsolve('Dx = y', 'Dy = -x') % 求解线性微分方程组
>>[u,v] = dsolve('Du=u+v,Dv=u-v')

```

计算结果为:

```

D1 =
    -exp(x)*t+C1+C2*exp(t)
D2 =
    y(t)=Int(exp(t*diff(f(t),'$'(t,2))/diff(f(t),t))*t,t)+C1
D3 =
    [
        -1]
    [
        1]
    [ sin(s-C1)]
    [-sin(s-C1)]
D4 =
    b*exp(a*t)
D5 =
    cos(a*t)
x =
    cos(t)*C1+sin(t)*C2
y =
    -sin(t)*C1+cos(t)*C2
u =
    1/2*C1*exp(2^(1/2)*t) - 1/4*C1*2^(1/2)*exp(-2^(1/2)*t) + 1/4*C1*2^(1/2) *exp (2^(1/2)*t) +
    1/2*C1*exp(-2^(1/2)*t) - 1/4*C2*2^(1/2)*exp(-2^(1/2)*t) + 1/4*C2 *2^(1/2)*exp(2^(1/2)*t)
v =
    -1/4*C1*2^(1/2)*exp(-2^(1/2)*t)+1/4*C1*2^(1/2)*exp(2^(1/2)*t)+1/2*C2*exp
    (2^(1/2)*t)+1/4*C2*2^(1/2)*exp(-2^(1/2)*t)-1/4*C2*2^(1/2)*exp(2^(1/2)*t)+
    1/2*C2*exp(-2^(1/2)*t)

```

3.2.3 符号函数的作图

命令 1 画符号函数的等高线图

函数 **ezcontour**

格式 **ezcontour(f)** %画出二元符号函数 $f=f(x,y)$ 的等高线图。函数 f 将被显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义, 则这些点将不显示。

ezcontour(f, domain) %在指定的定义域 **domain** 内画出二元函数 $f(x,y)$, 参量 **domain** 可以是四维向量 $[xmin, xmax, ymin, ymax]$ 或二维向量 $[min, max]$ (其中显示区域为: $min < x < max, min < y < max$)。

ezcontour(..., n) %用指定 $n*n$ 个栅格点 (对定义域的一种划分), 在缺省 (若没有指定) 的区域内画出函数 f 的图形。 n 的缺省值为 60。

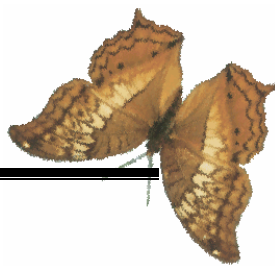
说明 该命令用函数表达式作为标题显示, 同时显示坐标轴的恰当的刻度标签。

例 3-29

```

>>syms x y
>>f = (1-x)^2*exp(-(x^2)-(y+1)^2)-5*(x/5-x^3-y^5)*sin(-x^2-y^2)-1/3*exp(-(x+1)^2-y^2);
ezcontour(f,[-3,3],49)

```



图形结果为图 3-4。

命令 2 用不同颜色填充的等高线图

函数 **ezcontourf**

格式 **ezcontourf(f)** %画出二元符号函数 $f=f(x,y)$ 的等高线图，并在不同的等高线之间自动用不同的颜色进行填充。函数 f 将被显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数变动激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。

ezcontourf(f, domain) %在指定的定义域 **domain** 内画出二元函数 $f(x,y)$ 的等高线图，并在不同的等高线之间自动用不同的颜色进行填充。定义域 **domain** 可以是四维向量 $[xmin, xmax, ymin, ymax]$ 或二维向量 $[min, max]$ （其中显示区域为： $min < x < max, min < y < max$ ）。

ezcontourf(..., n) %用指定 $n \times n$ 个栅格点（对定义域的一种划分），在缺省（若没有指定）的区域内画出函数 f 的等高线图，并在不同的等高线之间自动用不同的颜色进行填充。 n 的缺省值为 60。

例 3-30

```
>>syms x y
>>f = (1-x)^2*exp(-(x^2)-(y+1)^2)-5*(x/5-x^3-y^5)*sin(-x^2-y^2)-1/3*exp(-(x+1)^2-y^2);
ezcontourf(f,[-3,3],64)
```

图形结果为图 3-5。

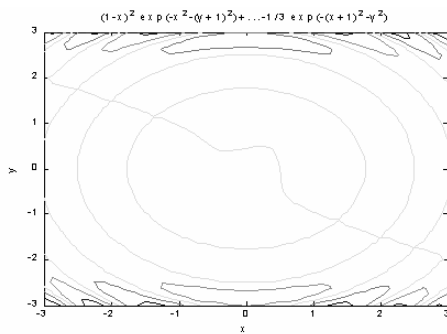


图 3-4 等高线图

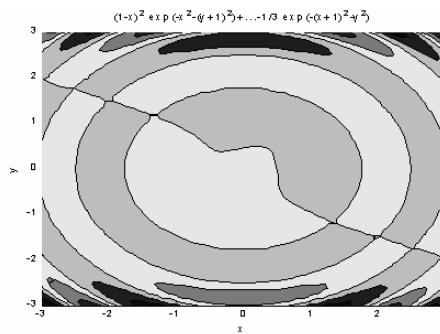


图 3-5 等高线填充图

命令 3 符号函数的三维网格图

函数 **ezmesh**

格式 **ezmesh(f)** %画出二元数学符号函数 $f=f(x,y)$ 的网格图。函数 f 将显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。

ezmesh(f, domain) %在指定的定义域 **domain** 内画出二元函数 $f(x,y)$ 的网格图，定义域 **domain** 可以是四维向量 $[xmin, xmax, ymin, ymax]$ 或二维向量 $[min, max]$ （其中显示区域为： $min < x < max, min < y < max$ ）。



`ezmesh(x,y,z)` %在缺省的矩形定义域范围 $[-2\pi < s < 2\pi, -2\pi < t < 2\pi]$ 内画参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 、 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 的网格图。

`ezmesh(x,y,z,[smin,smax,tmin,tmax])` %在指定的矩形定义域范围 $[smin < s < smax, min < t < tmax]$ 内画参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 、 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 的网格图。

`ezmesh(x,y,z,[min,max])` %用指定的矩形定义域 $[min < x < max, min < y < max]$ 画出函数 $z=f(x,y)$ 的网格图。

`ezmesh(f,...,n)` %用指定 $n*n$ 个栅格点，在缺省（若没有指定）的区域内画出函数 f 网的图形。 n 的缺省值为 60。

`ezmesh(...,'circ')` %在一圆形区域（圆心位于定义域在中心）的范围内画出函数 f 的网格图形。

例 3-31

```
>>syms x y
>>ezmesh(x*sin(-x^2-y^2),40,'circ')
>>colormap [0 0 1]
```

图形结果为：（图 3-6）

命令 4 同时画出曲面网格图与等高线图

函数 **ezmeshc**

格式 `ezmeshc(f)` %画出二元数学符号函数 $z=f(x,y)$ 的网格图形，同时在 xy 平面上显示其等高线图。函数 f 将被显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。

`ezmeshc(f,domain)` %在指定的定义域 $domain$ 内画出二元函数 $f(x,y)$ 的网格图及其等高线图， $domain$ 可以是四维向量 $[xmin,xmax,ymin,ymax]$ 或二维向量 $[min,max]$ (其中显示区域为： $min < x < max, min < y < max$)。

`ezmeshc(x,y,z)` %在缺省的矩形定义域范围 $[-2\pi < s < 2\pi, -2\pi < t < 2\pi]$ 内画出参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 、 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 的网格图形与其等高线图。

`ezmeshc(x,y,z,[smin,smax,tmin,tmax])` %在指定的矩形定义域范围 $[smin < s < smax, tmin < t < tmax]$ 内画出参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 、 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 的网格图形与其等高线图。

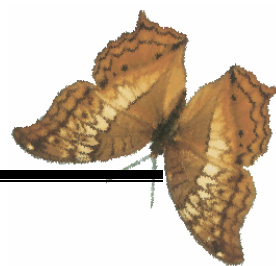
`ezsurf(x,y,z,[min,max])` %用指定的定义域 $[min < x < max, min < y < max]$ 画出函数 $z=f(x,y)$ 的网格图与等高线图。

`ezmeshc(f,...,n)` %用指定 $n*n$ 个栅格点，在缺省（若没有指定）的区域内画出函数 f 的网格图形与等高线图。 n 的缺省值为 60。

`ezmeshc(...,'circ')` %在一圆形区域（圆心位于定义域在中心）的范围内画出函数 f 的网格图形及其等高线图。

例 3-32

```
>>syms x y
>>ezmeshc(x*y/(1+x^2+y^2),[-5,5,-2*pi,2*pi],35)
```



图形结果为图 3-7。

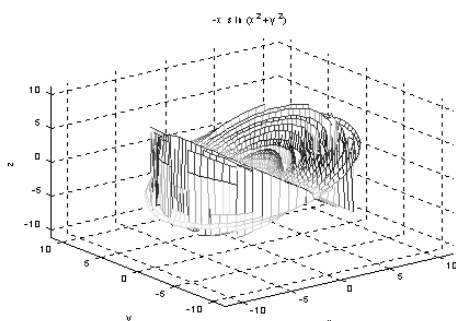


图 3-6 三维网格图

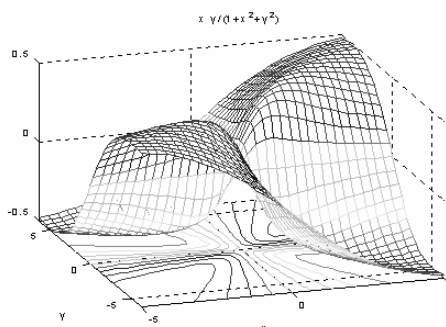


图 3-7 网格等高线图

命令 5 画符号函数的图形

函数 **ezplot**

格式 **ezplot(f)** %对于显式函数 $f=f(x)$ ，在缺省的范围 $[-\pi < x < \pi]$ 上画函数 $f(x)$ ；对于隐函数 $f=f(x,y)$ ，在缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 上画函数 $f(x,y)$ 的图形。

ezplot(f,[min,max]) %在指定的范围 $[\min < x < \max]$ 内画函数表达式 $f=f(x)$ 。若没有图形窗口存在，则该命令先生成标题为 Figure No.1 的新窗口，再在该窗口中操作；若已经有图形窗口存在，则在标号最高的图形窗口中进行操作。

ezplot(f,[xmin xmax],fign) %在指定标号 fign 的窗口中、指定的范围 $[xmin \text{ } xmax]$ 内画出函数 $f=f(x)$ 的图形。

ezplot(f,[xmin,xmax,ymin,ymax]) %在平面矩形区域 $[xmin < x < xmax, ymin < y < ymax]$ 上画出函数 $f(x,y)=0$ 的图形。

ezplot(x,y) %在缺省的范围 $0 < t < 2\pi$ 内画参数形式函数 $x=x(t)$ 与 $y=y(t)$ 的图形。

ezplot(x,y,[tmin,tmax]) %在指定的范围 $[tmin < t < tmax]$ 内画参数形式函数 $x=x(t)$ 与 $y=y(t)$ 的图形。

ezplot(...,figure) %在由参量 figure 句柄指定的图形窗口中画函数图形。

例 3-33

```
>>syms x y
>>ezplot(x^6-y^2)
```

图形结果为图 3-8。

例 3-34

```
>>syms x
>>ezplot(exp(x)*sin(x)/x)
>>grid on
```

图形结果为图 3-9。

命令 6 三维参量曲线图

函数 **ezplot3**

格式 **ezplot3(x,y,z)** %在缺省的范围 $0 < t < 2\pi$ 内画空间参数形式的曲线 $x=x(t)$ 、 $y=y(t)$



与 $z=z(t)$ 的图形。

`ezplot3(x,y,z,[tmin,tmax])` %在指定的范围 $tmin < t < tmax$ 内画空间参数形式的曲线 $x=x(t)$ 、 $y=y(t)$ 与 $z=z(t)$ 的图形。

`ezplot3(...,'animate')` %以动画形式画出空间三维曲线。

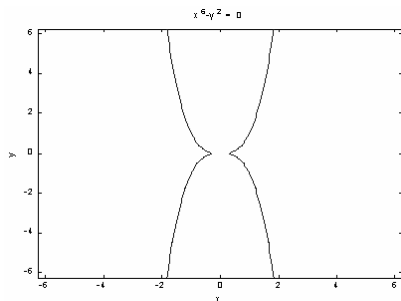


图 3-8 隐函数图

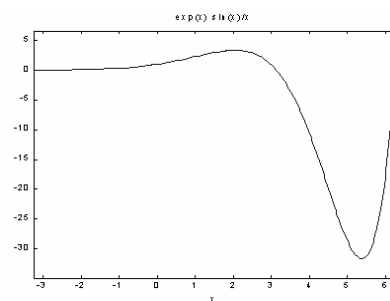


图 3-9 显函数图

例 3-35

```
>>syms t;
>>ezplot3(t*sin(t), t*cos(t), t,[0,20*pi])
```

图形结果为图 3-10。

命令 7 画极坐标图形

函数 **ezpolar**

格式 `ezpolar(f)` %在缺省的范围 $0 < \theta < 2\pi$ 内画极坐标函数 $\rho=f(\theta)$ 的图形。且将函数关系式显示于图形下方。

`ezpolar(f,[a,b])` %在指定的范围 $a < \theta < b$ 内画极坐标函数 $\rho=f(\theta)$ 的图形。且将函数关系式显示于图形下方。

例 3-36

```
>>syms t
>>ezpolar(1+cos(5*t))
```

图形结果为图 3-11。

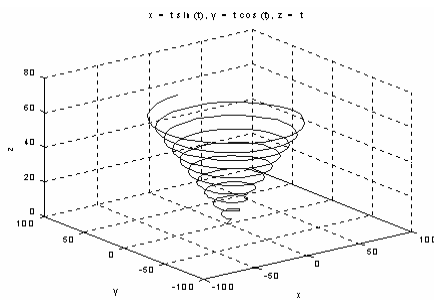


图 3-10 三维曲线图

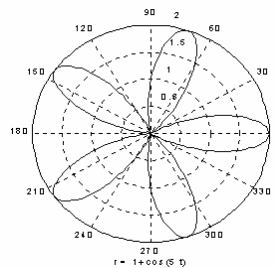
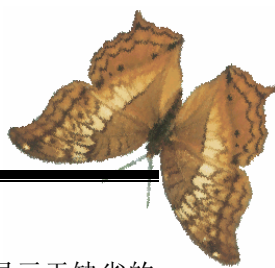


图 3-11 极坐标图

命令 8 三维带颜色的曲面图

**函数 ezsurf**

格式 `ezsurf(f)` %画出二元数学符号函数 $z=f(x,y)$ 的曲面图形。函数 f 将显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数的变动程度自动选择相应的计算栅格。若函数 f 在栅格点上没有定义,则这些点将不显示。

`ezsurf(f,domain)` %在指定的定义域 domain 内画出二元函数 $f(x,y)$ 的曲面图形, domain 可以是四维向量 $[\text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}]$, 或者是二维向量 $[\text{min}, \text{max}]$ (其中有 $\text{min} < x < \text{max}, \text{min} < y < \text{max}$)。

`ezsurf(x,y,z)` %在缺省的矩形定义域范围 $-2\pi < s < 2\pi, -2\pi < t < 2\pi$ 内画出参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 与 $z=z(s,t)$ 的曲面图形。

`ezsurf(x,y,z,[smin,smax,tmin,tmax])` 或 `ezsurf(x,y,z,[min,max])` %用指定的定义域画出参数形式的曲面图形

`ezsurf(...,n)` %用指定 $n*n$ 个栅格点,在缺省(若没有指定)的区域内画出函数 f 的图形, n 的缺省值为 60。

`ezsurf(...,'circ')` %在一圆形中心位于定义域在中心的范围内画出函数 f 的曲面图形

例 3-37

```
>>syms x y
>>ezsurf(real(atan(x+i*y)))
```

图形结果为图 3-12。

命令 9 同时画出曲面图与等高线图**函数 ezsurf**

格式 `ezsurf(f)` %画出二元数学符号函数 $z=f(x,y)$ 的曲面图形与其等高线图。函数 f 将显示于缺省的平面区域 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 内。系统将根据函数的变动程度自动选择相应的计算栅格。若函数 f 在栅格点上没有定义,则这些点将不显示。

`ezsurf(f,domain)` %在指定的定义域 domain 内画出二元函数 $f(x,y)$ 的曲面图形及其等高线图, domain 可以是四维向量 $[\text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}]$ 或二维向量 $[\text{min}, \text{max}]$ (其中有 $\text{min} < x < \text{max}, \text{min} < y < \text{max}$)。

`ezsurf(x,y,z)` %在缺省的矩形定义域范围 $-2\pi < s < 2\pi, -2\pi < t < 2\pi$ 内画出参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 与 $z=z(s,t)$ 的曲面图形与等高线图。

`ezsurf(x,y,z,[smin,smax,tmin,tmax])` 或 `ezsurf(x,y,z,[min,max])` %用指定的定义域画出参数形式的曲面图形与等高线图

`ezsurf(...,n)` %用指定 $n*n$ 个栅格点,在缺省(若没有指定)的区域内画出函数 f 的曲面图形与等高线图, n 的缺省值为 60。

`ezsurf(...,'circ')` 在一圆形中心位于定义域的中心范围内画出函数 f 的曲面图形与等高线图

例 3-38

```
>>syms x y
>>ezsurf(x*y/(1+x^2+y^2),[-5,5,-2*pi,2*pi],35,'circ')
```

图形结果为图 3-13。

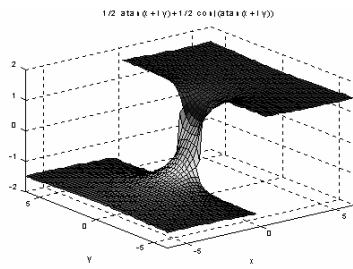


图 3-12 三维曲面图

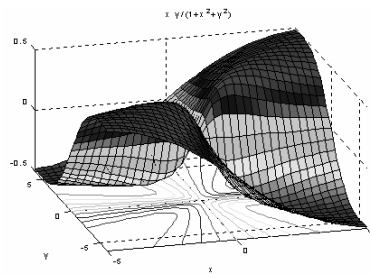


图 3-13 三维曲面等高线图

3.2.4 积分变换

命令 1 Fourier 积分变换

函数 **fourier**

格式 **F = fourier(f)**

说明 对符号单值函数 f 中的缺省变量 x (由命令 `findsym` 确定) 计算 Fourier 变换形式。

缺省的输出结果 F 是变量 w 的函数: $f = f(x) \Rightarrow F = F(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx} dx$

若 $f = f(w)$, 则 `fourier(f)` 返回变量为 t 的函数: $F = F(t)$ 。

$F = \text{fourier}(f, v)$ 对符号单值函数 f 中的指定变量 v 计算 Fourier 变换形式:

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{\infty} f(x)e^{-ivx} dx$$

$F = \text{fourier}(f, u, v)$ 令符号函数 f 为变量 u 的函数, 而 F 为变量 v 的函数:

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{\infty} f(u)e^{-ivu} dx$$

例 3-39

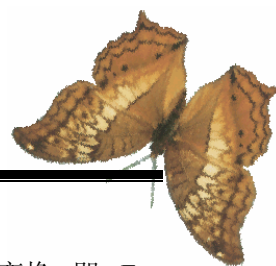
```
>>syms x w u v
>>f = sin(x)*exp(-x^2); F1 = fourier(f)
>>g = log(abs(w)); F2 = fourier(g)
>>h = x*exp(-abs(x)); F3 = fourier(h,u)
>>syms x real
>>k = cosh(-x^2*abs(v))*sinh(u)/v
>>F4 = fourier(k,v,u)
```

计算结果为:

```
F1 =
-1/2*i*pi^(1/2)*exp(-1/4*(w-1)^2)+1/2*i*pi^(1/2)*exp(-1/4*(w+1)^2)
F2 =
fourier(log(abs(w)),w,t)
F3 =
-4*i/(1+u^2)^2*u
F4 =
sinh(u)*(1/2*fourier(1/v*exp(x^2*abs(v)),v,u)-i*atan(u/x^2))
```

命令 2 逆 Fourier 积分变换

函数 **ifourier**



格式 $f = \text{ifourier}(F)$

说明 输出参量 $f = f(x)$ 为缺省变量 w 的标量符号对象 F 的逆 Fourier 积分变换。即: $F = F(w) \rightarrow f = f(x)$ 。若 $F = F(x)$, $\text{ifourier}(F)$ 返回变量 t 的函数: 即: $F = F(x) \rightarrow f = f(t)$ 。逆 Fourier

积分变换定义为: $f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)e^{iwx} dw$

$f = \text{ifourier}(F,u)$ 使函数 f 为变量 u (u 为标量符号对象) 的函数: $f(u) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)e^{iwu} dw$

$f = \text{ifourier}(F,v,u)$ 使 F 为变量 v 的函数, f 为变量 u 的函数: $f(u) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(v)e^{iuv} dv$

例 3-40

```
>>syms w v x t
>>syms a real
>>f = sqrt(exp(-w^2/(4*a^2)));
>>IF1 = ifourier(f)
>>g = exp(-abs(x));
>>IF2 = ifourier(g)
>>h = sinh(-abs(w)) - 1;
>>IF3 = simple(ifourier(h,t))
>>syms w real
>>k = exp(-w^2*abs(v))*sin(v)/v;
>>IF4 = ifourier(k,v,t)
```

计算结果为:

```
IF1 =
    ifourier(exp(-1/4*w^2/a^2)^(1/2),w,x)
IF2 =
    1/(1+t^2)/pi
IF3 =
    -1/2*(pi*ifourier(exp(abs(w)),w,t)+pi*ifourier(exp(abs(w)),w,t)*t^2-...
    1+2*pi*Dirac(t))/(1+t^2)/pi
IF4 =
    1/2*(atan((t+1)/w^2)-atan((t-1)/w^2))/pi
```

命令 3 Laplace 变换

函数 **laplace**

格式 $L = \text{laplace}(F)$

说明 输出参量 $L = L(s)$ 为有缺省符号自变量 t 的标量符号对象 F 的 Laplace 变换。即: $F = F(t) \rightarrow L = L(s)$ 。若 $F = F(s)$, 则 $\text{fourier}(F)$ 返回变量为 t 的函数 L 。

即: $F = F(s) \rightarrow L = L(t)$ 。Laplace 变换定义为: $L(s) = \int_0^{\infty} F(t)e^{-st} dt$

$\text{laplace}(F,t)$ 使函数 L 为变量 t (t 为标量符号自变量) 的函数: $L(s) = \int_0^{\infty} F(x)e^{-tx} dx$

$\text{fourier}(F,w,z)$ 使 L 为变量 z 的函数, F 为变量 w 的函数: $L(z) = \int_0^{\infty} F(w)e^{-zw} dw$

例 3-41

```
>>syms x s t v
>>f1 = sqrt(t);
```



```

>>L1 = laplace(f)
>>f2 = 1/sqrt(s);
>>L2 = laplace(f2)
>>f3 = exp(-a*t);
>>L3 = laplace(f3,x)
>>f4 = 1 - sin(t*v);
>>L4 = laplace(f4,v,x)

```

计算结果为:

```

L1 =
    1/(s-1/s^2)
L2 =
    (pi/t)^(1/2)
L3 =
    1/(x+a)
L4 =
    1/x-t/(x^2+t^2)

```

命令 4 逆 Laplace 变换

函数 ilaplace

格式 F = ilaplace(L)

说明 输出参量 F = F(t) 为缺省变量 s 的标量符号对象 L 的逆 Laplace 变换

即: $F = F(w) \rightarrow f = f(x)$ 。若 $L = L(t)$, 则 ifourier(L) 返回变量为 x 的函数 F。即: $F = F(x)$

→ $f = f(t)$ 。逆 Laplace 变换定义为: $F(t) = \int_{c-i\infty}^{c+i\infty} L(s)e^{st} dt$

其中 c 为使函数 L(s) 的所有的奇点位于直线 $s = c$ 左边的实数。

$F = \text{ilaplace}(L,y)$ 使函数 F 为变量 y (y 为标量符号对象) 的函数: $F(y) = \int_{c-i\infty}^{c+i\infty} L(y)e^{sy} ds$

$F = \text{ilaplace}(L,y,x)$ 使 F 为变量 x 的函数, L 为变量 y 的函数: $F(x) = \int_{c-i\infty}^{c+i\infty} L(y)e^{xy} dy$

例 3-42

```

>>syms a s t u v x
>>f = exp(x/s^2);
>>IL1 = ilaplace(f)
>>g = 1/(t-a)^2;
>>IL2 = ilaplace(g)
>>k = 1/(u^2-a^2);
>>IL3 = ilaplace(k,x)
>>y = s^3*v/(s^2+v^2);
>>IL4 = ilaplace(y,v,x)

```

计算结果为:

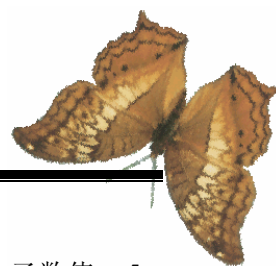
```

IL1 =
    ilaplace(exp(x/s^2),s,t)
IL2 =
    x*exp(a*x)
IL3 =
    1/(-a^2)^(1/2)*sin((-a^2)^(1/2)*x)
IL4 =
    s^3*cos((s^2)^(1/2)*x)

```

命令 5 Riemann ζ -函数



**函数 zeta**

格式 $Y = \text{zeta}(X)$ %计算数值矩阵、或符号矩阵参量 x 中每一元素的 ζ -函数值。 ζ -

$$\text{函数定义为: } \zeta(w) = \sum_{k=1}^{\infty} \frac{1}{k^w}$$

$Y = \text{zeta}(n, X)$ %返回 $\zeta(X)$ 函数的 n 阶导数

例 3-43

```
>>syms x y
>>Y1 = zeta(1.5)
>>Y2 = zeta(1.2:0.1:2.1)
>>Y3 = zeta([x 2;4 x+y])
>>DZ = diff(zeta(x),x,3)
```

计算结果为:

```
Y1 =
    2.6124
Y2 =
Columns 1 through 7
    5.5916    3.9319    3.1055    2.6124    2.2858    2.0543    1.8822
Columns 8 through 10
    1.7497    1.6449    1.5602
Y3 =
[ zeta(x,2), zeta(2,2)]
[ zeta(4,2), zeta(x+y,2)]
DZ =
    zeta(3,x)
```

命令 6 z-变换**函数 ztrans**

格式 $F = \text{ztrans}(f)$ %对缺省自变量为 n (就像由命令 `findsym` 确定的一样) 的单值函数 f 计算 z -变换。输出参量 F 为变量 z 的函数: $f = f(n) \rightarrow F = F(z)$ 。

$$\text{函数 } f \text{ 的 } z\text{-变换定义为: } F(z) = \sum_{n=0}^{\infty} \frac{f(n)}{z^n}$$

若函数 $f = f(z)$, 则 $\text{ztrans}(f)$ 返回一变量为 w 的函数: $f = f(z) \rightarrow F = F(w)$

$F = \text{ztrans}(f,w)$ %用符号变量 w 代替缺省的 z 作为函数 F 的自变量

$$F(w) = \sum_{n=0}^{\infty} \frac{f(n)}{w^n}$$

$$F = \text{ztrans}(f,k,w) \quad \% \text{对函数 } f \text{ 中指定的符号变量 } k \text{ 计算 } z\text{-变换: } F(w) = \sum_{n=0}^{\infty} \frac{f(k)}{w^n}$$

例 3-44

```
>>syms a k w x n z
>>f1 = n^4;
>>ZF1 = ztrans(f)
>>f2 = a^z;
>>ZF2 = ztrans(g)
```



```
>>f3 = sin(a*n);
>>ZF3 = ztrans(f,w)
>>f4 = exp(k*n^2)*cos(k*n);
>>ZF4 = ztrans(f,k,x)
```

计算结果为:

```
ZF1 =
      z*(z^3+11*z^2+11*z+1)/(z-1)^5
ZF2 =
      w/a/(w/a-1)
ZF3 =
      -w*sin(a)/(-w^2+2*w*cos(a)-1)
ZF5 =
      (x/exp(n^2)-cos(n))*x/exp(n^2)/(x^2/exp(n^2)^2-2*x/exp(n^2)*cos(n)+1)
```

命令 7 逆 z-变换

函数 **iztrans**

格式 **f = iztrans(F)**

说明 输出参量 $f = f(n)$ 为有缺省变量 z 的单值符号函数 F 的逆 z -变换。即: $F = F(z) \rightarrow f = f(n)$ 。若 $F = F(n)$, 则 **iztrans(F)** 返回变量为 k 的函数 $f(k)$ 。

即: $F = F(n) \rightarrow f = f(k)$ 。逆 z -变换定义为:
$$f(n) = \frac{1}{2\pi i} \oint_{|z|=R} F(z) z^{n-1} dz, \quad n=1,2,3,\dots$$

其中 R 为一正实数, 它使函数 $F(z)$ 在圆域之外 $|z| \geq R$ 是解析的。

$f = \text{iztrans}(F,k)$ 使函数 f 为变量 k (k 为标量符号对象) 的函数 $f(k)$:

$$f(k) = \frac{1}{2\pi i} \oint_{|z|=R} F(z) z^{k-1} dz, \quad k=1,2,3,\dots$$

$f = \text{iztrans}(F,w,k)$ 使函数 F 为变量 w 的函数, f 为变量 k 的函数:

$$f(k) = \frac{1}{2\pi i} \oint_{|w|=R} F(w) w^{k-1} dw, \quad k=1,2,3,\dots$$

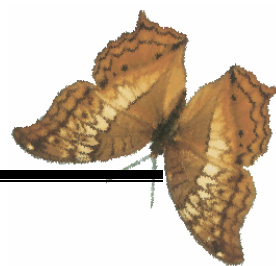
例 3-45

```
>>syms a n k x z
>>f1= 2*z/(z^2+2)^2;
>>IZ1 = iztrans(f1)
>>f2 = n/(n+1);
>>IZ2 = iztrans(f2)
>>f3 = z/sqrt(z-a);
>>IZ3 = iztrans(f3,k)
>>f4 = exp(z)/(x^2-2*x*exp(z));
>>IZ4 = iztrans(f4,x,k)
```

计算结果为:

```
IZ1 =
      -1/8*sum(1/_alpha*(1/_alpha)^n,_alpha
IZ2 =
      (-1)^k
IZ3 =
      iztrans(z/(z-a)^(1/2),z,k)
```





$$I Z 4 =$$

$$1/4*(-\operatorname{charfcn}[0](k)-2*\operatorname{charfcn}[1](k)*\exp(z)+2^k*\exp(z)^k)/\exp(z)$$

3.2.5 Taylor 级数

命令 1 符号函数的 Taylor 级数展开式

函数 **taylor**

格式 `r = taylor(f,n,v)` %返回符号表达式 `f` 中的、指定的符号自变量 `v` (若表达式 `f` 中有多个变量时) 的 `n-1` 阶的 Maclaurin 多项式 (即在零点附近 `v=0`) 近似式, 其中 `v` 可以是字符串或符号变量。

`r = taylor(f)` %返回符号表达式 `f` 中的、符号变量 `v` 的 6 阶的 Maclaurin 多项式 (即在零点附近 `v=0`) 近似式, 其中 `v=findsym(f)`。

`r = taylor(f,n,v,a)` %返回符号表达式 `f` 中的、指定的符号自变量 `v` 的 `n-1` 阶的 Taylor 级数 (在指定的 `a` 点附近 `v=a`) 的展开式。其中 `a` 可以是一数值、符号、代表一数值值的字符串或未知变量。我们指出的是, 用户可以以任意的次序输入参量 `n`、`v` 与 `a`, 命令 **taylor** 能从它们的位置与类型确定它们的目的。解析函数 `f(x)` 在点

$$x=a \text{ 的 Taylor 级数定义为: } f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

例 3-46

```
>>syms x y a pi m m1 m2
>>f = sin(x+pi/3);
>>T1 = taylor(f)
>>T2 = taylor(f,9)
>>T3 = taylor(f,a)
>>T4 = taylor(f,m1,m2)
>>T5 = taylor(f,m,a)
>>T6 = taylor(f,y)
>>T7 = taylor(f,y,m) % 或 taylor(f,m,y)
>>T8 = taylor(f,m,y,a)
>>T9 = taylor(f,y,a)
```

计算结果为:

```
T1 =
1/2*3^(1/2)+1/2*x-1/4*3^(1/2)*x^2-1/12*x^3+1/48*3^(1/2)*x^4+1/240*x^5

T2 =
1/2*3^(1/2)+1/2*x-1/4*3^(1/2)*x^2-1/12*x^3+1/48*3^(1/2)*x^4+1/240*x^5-1/1440*3^(1/2)*
x^6-1/10080*x^7+1/80640*3^(1/2)*x^8

T3 =
sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*(x-a)^2-1/6*cos(a+1/3*pi)*
(x-a)^3+1/24*sin(a+1/3*pi)*(x-a)^4+1/120*cos(a+1/3*pi)*(x-a)^5

T4 =
sin(m2+1/3*pi)+cos(m2+1/3*pi)*(x-m2)-1/2*sin(m2+1/3*pi)*(x-m2)^2-1/6*
cos(m2+1/3*pi)*(x-m2)^3+1/24*sin(m2+1/3*pi)*(x-m2)^4+1/120*
cos(m2+1/3*pi)*(x-m2)^5

T5 =
sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*(x-a)^2-1/6*cos(a+1/3*pi)*
(x-a)^3+1/24*sin(a+1/3*pi)*(x-a)^4+1/120*cos(a+1/3*pi)*(x-a)^5

T6 =
sin(y+1/3*pi)+cos(y+1/3*pi)*(x-y)-1/2*sin(y+1/3*pi)*(x-y)^2-1/6*cos(y+1/3*pi)
```



```

*(x-y)^3+1/24*sin(y+1/3*pi)*(x-y)^4+1/120*cos(y+1/3*pi)*(x-y)^5
T7 =
sin(m+1/3*pi)+cos(m+1/3*pi)*(x-m)-1/2*sin(m+1/3*pi)*(x-m)^2-1/6*cos(m+1/3*pi)
*(x-m)^3+1/24*sin(m+1/3*pi)*(x-m)^4+1/120*cos(m+1/3*pi)*(x-m)^5
T8 =
sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*(x-a)^2-1/6*cos(a+1/3*pi)*
(x-a)^3+1/24*sin(a+1/3*pi)*(x-a)^4+1/120*cos(a+1/3*pi)*(x-a)^5
T9 =
sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*(x-a)^2-1/6*cos(a+1/3*pi)*
(x-a)^3+1/24*sin(a+1/3*pi)*(x-a)^4+1/120*cos(a+1/3*pi)*(x-a)^5

```

命令 2 Taylor 级数计算器

函数 `taylorltool`

格式 `taylorltool` %该命令生成一图形用户界面，显示缺省函数 $f=x*\cos(x)$ 在区间 $[-2*\pi, 2*\pi]$ 内的图形，同时显示函数 f 的前 $N=7$ 项的 Taylor 多项式级数和(在 $a=0$ 附近的)图形，如图 1。通过更改 $f(x)$ 项可得不同的函数图形。

`taylorltool('f')` %对指定的函数 f ，用图形用户界面显示出 Taylor 展开式。(图 3-14)

例 3-47

```
>>taylorltool('sin(x*sin(x))')
```

再通过改变相关的参量，可得如图 3-15。

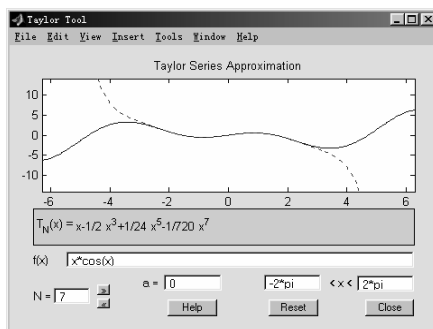


图 3-14 Taylor 级数计算器

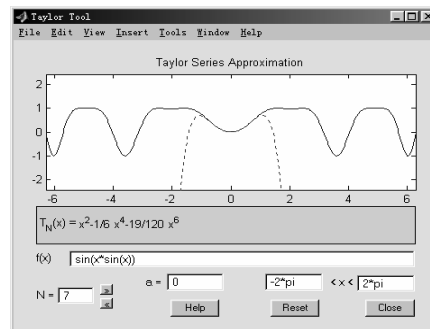


图 3-15 函数 $\sin(x*\sin(x))$ 的 `taylorltool` 界面

3.2.6 其它

命令 1 Jacobian 矩阵

函数 `jacobian`

格式 `R = jacobian(w,v)`

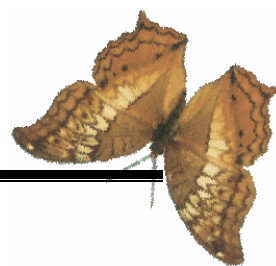
说明 计算 w 对 v 的 Jacobian 矩阵。其中 w 为符号单值函数表达式或符号列向量， v

为一符号行向量。输出参量 $R = (r_{ij})$ 的元素 r_{ij} 为： $r_{ij} = \frac{\partial w(i)}{\partial v(j)}$ ， $i=1,2,\dots,\text{size}(w)$ ，

$j=1,2,\dots,\text{length}(v)$

例 3-48

```
>>syms x y z u v w
>>w = [x*y*z; y; x+z];
```



```
>>v = [x,y,z];
>>R = jacobian(w,v)
>>b = jacobian(x+u, v)
```

计算结果为:

```
R =
[ y*z, x*z, x*y]
[ 0, 1, 0]
[ 1, 0, 1]
b =
[ 1, 0, 0]
```

命令 2 Jordan 标准形

函数 **jordan**

格式 **J = jordan(A)** %计算矩阵 A 的 Jordan 标准形。其中 A 为一确切已知的符号或数值矩阵。即它的元素必须是整数或小整数的比值。任何的矩阵输入误差将导致不同的 Jordan 标准形。即 Jordan 标准形对数据是敏感的。

[V,J] = jordan(A) %返回 Jordan 标准形矩阵 J 与相似变换矩阵 V，其中 V 的列向量为矩阵 A 的广义特征向量。它们满足： $V \backslash A * V = J$ 。

例 3-49

```
>>A = [1 -3 -2; -1 1 -1; 2 4 5]
>> [V,J] = jordan(A)
>>V = double(V);
>>Test = all(all(V \ A * V == J))
```

计算结果为:

```
V =
-1 -1 1
0 -1 0
1 2 0
J =
3 0 0
0 2 1
0 0 2
Test = 1
```

命令 3 Lamber 的 W 函数

函数 **lambertw**

格式 **Y = lambertw(X)** %计算参量 X 的每一元素 x 的 Lamber 的 W 函数值，其中 X 为一数值或符号矩阵。Lamber 的函数 $W=W(x)$ 为方程的解： $we^w = x$ 。

例 3-50

```
>>W1 = lambertw([-exp(-1); pi])
>>syms x y
>>W2 = lambertw([0 x; 1 y])
```

计算结果为:

```
W1 =
-1.0000 + 0.0000i
1.0737
W2 =
[ 0, lambertw(x)]
[ lambertw(1), lambertw(y)]
```

命令 4 符号表达式的 LaTeX 的表示式

**函数 latex**

格式 latex(S) %返回符号表达式 S 的 LaTeX 格式的表示式。该格式可以使表达式 S 在图形窗口中进行显示（如命令 title、text 等）。

例 3-51

```
>>syms x
>>f = taylor(sin(1+x));
>>Lat1 = latex(f)
>>M = sym(magic(3));
>>Lat2 = latex(M)
```

计算结果为：

```
Lat1 =
\sin(1)+\cos(1)\mbox {{\tt `x~`}}-1/2\,\sin(1)\mbox {{\tt `x~`}}^2-1/6\,\cos(1)\mbox {{\tt `x~`}}^3+1/24\,\sin(1)\mbox {{\tt `x~`}}^4+\frac {1}{120}\,\cos(1)\mbox {{\tt `x~`}}^5
Lat2 =
\left [\begin {array} {ccc} 8&1&6\\noalign{\medskip}3&5&7\\noalign{\medskip}4&9&2\end {array} \right ]
```

命令 5 调用 Maple 内核**函数 maple**

格式 r = maple('statement') %将参数命令 statement 传递给 Maple 内核，且返回计算结果。在必要时，可以在参量 statement 后面加上分号(;)。

r = maple('function',arg1,arg2,...) %该命令接受任何的带引号的函数名 'function'，与相关的输入参量 arg1,arg2,...。在必要时，要将输入参量转换成符号表达式。若输入参量为 syms，则 maple 返回一 sym，否则返回一类型为 char 的结果。

[r, status] = maple(...) %有条件地返回警告/错误信息。当语句能顺利执行，则 r 为计算结果，status 为 0；若语句不能通过执行，r 为相应的警告/错误信息，而 status 为一正整数。

maple('traceon')、maple traceon、maple trace on %将显示所有的后面的 Maple 语句与其相应的结果显示于屏幕上

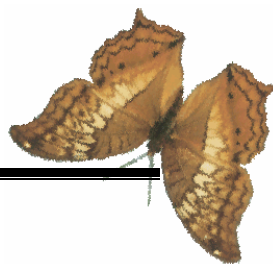
maple('traceoff')、maple traceoff、maple trace off %将关闭上面的操作特性

例 3-52

```
>>Pi = maple('evalf(Pi,100)')
>>syms x
>>v = [x^2-1;x^2-4]
>>maple traceon
>>w = factor(v)
```

计算结果为：

```
Pi =
3.1415926535897932384626433832795028841971693993751058209749445923078164...
06286208998628034825342117068
v =
[ x^2-1]
[ x^2-4]
statement:
map(ifactor,array([[x^2-1],[x^2-4]]));
```

```

result:
Error, (in ifactor) invalid arguments
statement:
map(factor,array([[x^2-1],[x^2-4]]));
result:
matrix([[ (x-1)*(x+1)], [(x-2)*(x+2)]]])
w =
[ (x-1)*(x+1)]
[ (x-2)*(x+2)]

```

命令 6 初始化 Maple 内核

函数 mapleinit

格式 mapleinit 该命令用于确定包含 Maple 库的路径，再装载 Maple 的线性代数与积分变换包、初始化命令 digits、指定几个别名。用户可以编辑 mapleinit 的 M-文件，用于改变到 Maple 包的路径，只需按如下的方法改变变量 initstring 的值：

1. 若用户已经有一 Maple V, Release 5 的库在目录 C:\Maple\Lib 上，在文件 mapleinit.m 中加入：maplelib = 'C:\MAPLE\LIB'
2. 从 MATLAB 中删除旧的 Maple 包版本。

命令 7 Maple 数学函数的数值计算

函数 mfun

格式 Y = mfun('function',par1,par2,par3,par4)

说明 计算一指定的 Maple 软件中已知的数学函数 function 的数值。每一参量 par 为该函数相应的具体数值。用户可以输入满 4 个参量。最后指定的参量可以是矩阵，通常对应于 x。其他参量的位数取决于该函数规定的范围。用户可以通过下面的命令获得相关参数的信息：help mfunlist; mhelp function; Maple 用 16 位精度计算函数 function。函数 function 中的任何奇异值将返回 NaN。

例 3-53

```

>>M1 = mfun('dilog',5)
>>M2 = mfun('Psi',[3*i 0])

```

计算结果为：

```

M1 =
-2.3699
M2 =
1.1080 + 1.7375i NaN

```

命令 8 列出命令 mfun 中特定的 Maple 函数

函数 mfunlist

格式 mfunlist

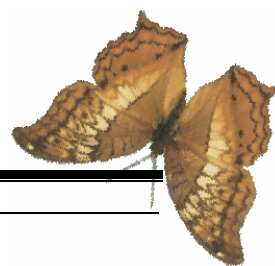
1. 列出在使用命令 mfun 中用到的特殊的数学函数。下表中参量的一些约定：x,y：实数参量；z,z1,z2：复数参量；m,n：整数参量

表 3-1 mfun 特殊函数

函数名	定 义	Mfun 名	参量说明
Bernoulli 数 与多项式	生成函数： $\frac{e^{xt}}{e^t - 1} = \sum_{n=0}^{\infty} B_n(x) * \frac{t^{n-1}}{n!}$	Bernoulli(n) Bernoulli(n,t)	$n \geq 0$ $0 < t < 2\pi$



Bessel 函数	BesselJ, BesselJ: 第一类 Bessel 函数 BesselK, BesselY: 第二类 Bessel 函数	BesselJ(v,x) BesselY(v,x) BesselI(v,x) BesselK(v,x)	v 为实数
Beta 函数	$B(x, y) = \frac{\Gamma(x) \times \Gamma(y)}{\Gamma(x + y)}$	Beta(x,y)	
二项式系数	$\binom{m}{n} = \frac{m!}{n!(m-n)!}$ $= \frac{\Gamma(m+1)}{\Gamma(n+1) \times \Gamma(m-n+1)}$	Binomial(m,n)	
完全椭圆积分	第一、二、三类 Legendre 完全椭圆积分	LegendreKc(k) LegendreEc(k) LegendrePic(a,k)	a 为任意实数 $-\infty < a < \infty$ k 为任意实数 $0 < k < 1$
带余模的完全椭圆积分	与余模相关的第一、二、三类 Legendre 完全椭圆积分	LegendreKc1(k) LegendreEc1(k) LegendrePic1(a,k)	a 为任意实数 $-\infty < a < \infty$ k 为任意实数 $0 < k < 1$
余差函数 与它的累积分	Erfc(z) = $\sqrt{2} \int_z^\infty e^{-t^2} dt = 1 - \text{erf}(z)$ erfc(n,z) = $\int_z^\infty \text{erfc}(n-1, z) dt$	erfc(z) erfc(n,z)	n>0
Dawson 积分	$F(x) = e^{-x^2} \int_0^x e^{-t^2} dt$	dawson(x)	
Ψ-函数	$\psi(x) = \frac{d}{dx} \ln \Gamma(x)$	Psi(x)	
二重对数积分	$f(x) = \int_1^x \frac{\ln t}{1-t} dt$	dilog(x)	x>1
误差函数	$\text{erf}(z) = \sqrt{2} \int_0^z e^{-t^2} dt$	erf(z)	
Euler 数与多项式	生成 Euler 数的函数: $\frac{1}{\text{cht}} = \sum_{n=0}^{\infty} E_n \times \frac{t^n}{n!}$	euler(n) euler(n,z)	n≥0 t < π/2
指数积分	$E_i(n,z) = \int_1^\infty \frac{e^{-zt}}{t^n} dt$ $E_i(x) = \text{PV} - \int_{-\infty}^x \frac{e^t}{t} dt$	Ei(n,z) Ei(x)	n≥0 real(z)>0
Fresnel 正弦 与余弦积分	$C(x) = \int_0^x \cos(\frac{\pi}{2} t^2) dt$ $S(x) = \int_0^x \sin(\frac{\pi}{2} t^2) dt$	FresnelC(x) FresnelS(x)	



Γ-函数	$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$	GAMMA(z)	
调和函数	$h(n) = \sum_{k=1}^n \frac{1}{k}$ $= \Psi(n+1) + \gamma$	harmonic(n)	n>0
双曲正弦 与余弦积分	$\text{Shi}(z) = \int_0^z \frac{\sinh t}{t} dt$ $\text{Chi}(z) = \gamma + \ln(z) + \int_0^z \frac{\cosh t - 1}{t} dt$	Shi(z) Chi(z)	
广义超几何函数	$F(n,d,z) = \sum_{k=0}^{\infty} \frac{\prod_{i=1}^j \frac{\Gamma(n_i + k)}{\Gamma(n_i)} z^k}{\prod_{i=1}^m \frac{\Gamma(d_i + k)}{\Gamma(d_i)} k!}$	hypergeom(n,d,x) 其中 n = [n1,n2,...] d = [d1,d2,...]	n1,n2,... 为实数 d1,d2,... 为非负实数
不完全椭圆积分	第一、二、三类不完全 Legendre 完全椭圆积分	LegendreF(x,k) LegendreE(x,k) LegendrePi(x,a,k)	0<x≤∞, a 为实数 -∞<a<∞, k 为实数 0<k<1
不完全 Γ-函数	$\Gamma(a,z) = \int_z^{\infty} e^{-t} t^{a-1} dt$	GAMMA(z1,z2)	
Γ-函数的对数	$\ln \Gamma(z) = \ln(\Gamma(z))$	lnGAMMA(z)	
对数积分	$\text{Li}(x) = \text{PV} \left(\int_0^x \frac{dt}{\ln t} \right)$ $= \text{Ei}(\ln(x))$ 其中 Ψ(z) 为 Γ-函数	Li(x)	x>1
Γ 多项式函数	$\psi^{(n)}(z) = \frac{d^n \psi(z)}{dz^n}$	Psi(n,z)	N≥0
移位正弦积分	$\text{Ssi}(z) = \text{Si}(z) - \pi/2$	Ssi(z)	

对于上面的特殊函数 function，用户可以通过下面的命令得到更多的帮助信息：`mhelf function`

总的来说，函数的精度跟它的根相比会较低，且当它的参数相对而言较大时，精度也较低。函数的执行时间取决于特定的函数与它的输入参量。总之，其计算将比标准的 MATLAB 计算慢一些。

2. 正交多项式函数：

下面的函数需要 Maple 正交多项式包，它们仅仅对于 MATLAB 的扩展符号数学工具箱有用。在使用这些函数之前，用户要用下面的命令初始化 Maple 正交多项式包：`maple('with','orthopoly')`

表 3-2 正交多项式函数

下表参量的约定：n：非负整数；x：任意实数

多项式	Maple 名	参量说明
Gegenbauer 多项式	G(n,a,x)	a 为非有理数代数表达式或者是大于-1/2 的有理数
Hermite 多项式	H(n,x)	
Laguerre 多项式	L(n,x)	



广义 Laguerre 多项式	$L(n,a,x)$	a 为非有理数代数表达式或者是大于-1 的有理数
Legendre	$P(n,x)$	
Jacobi	$P(n,a,b,x)$	a 与 b 为非有理数代数表达式或者是大于-1 的有理数
第一、二类 Chebyshev 多项式	$T(n,x)U(n,x)$	

命令 9 Maple 命令帮助

函数 **mhhelp**

格式 **mhhelp** topic、**mhhelp**('topic')

说明 返回 Maple 软件中指定的 Maple 标题 topic 的在线帮助文档信息。

命令 10 交互式计算 Riemann 和

函数 **rsums**

格式 **rsums**(f) %交互式地通过 Riemann 和计算函数 $f(x)$ 的积分。**rsums**(f) 显示函数 f 的图形。用户可以通过拖动图形下方的滑块来调整 Riemann 和的项数，有效的项数从 2 到 128。

例 3-54

```
>>rsums sin(-5*x^2)
```

计算图形为图 3-16。

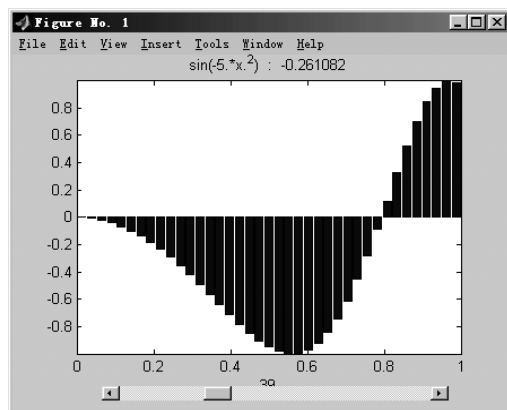


图 3-16 函数的 Riemann 和

命令 11 在一符号表达式或矩阵中进行符号替换

函数 **subs**

格式 **R = subs**(S) %用从调用的函数中获得的变量值，或 MATLAB 的工作空间中存在的变量值，替换表达式 S 中所有出现的相同的变量，同时自动进行化简计算；若是数值表达式，则计算出结果。

R = subs(S,old,new) %用新值 new 替换表达式 s 中的旧值 old ，参量 old 是一符号变量或代表一变量名的字符串， new 是一符号/数值变量或表达式。若 old 与 new 为有相同大小的阵列，则用 new 中相应的元素替换 old 中的元素；若 S 与 old 为标量，而 new 为阵列或单元阵列，则标量 S 与 old 将扩展为与 new 同型的阵列；若 new 为数值矩阵的单元阵列，则替换按元素的方向执行。若 **subs**(S,old,new) 没有改变



S, 则 `subs(S,old,new)` 被证明是可靠的。这提供了对以前版本的向后兼容性, 且不会交换参量的位置。

例 3-55

```
>>a = 980,C1=3;
>>y = dsolve('Dy = -a*y')
>>syms b
>>subs(y)
>>subs(a+b,a,4)
>>subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})
>>subs(exp(a*t),'a',-magic(2))
>>subs(x*y,{x,y},{[0 1;-1 0],[1 -1;-2 1]})
```

命令 12 创建符号数值、变量与对象

函数 `sym`

格式 `S = sym(A)` %用输入参量 A, 构造一类型为 ‘sym’ 的对象 s。若 A 为字符串, 则 S 为符号数值或变量; 若 A 为一数值标量或矩阵, 则 S 为代表所给数值的符号表达式。

`x = sym('x')` %创建一名字为 ‘x’ 的符号变量, 且将结果存于 x。

`pi = sym('pi')` %创建一符号数值, 这可避免了用浮点近似表示 π 的误差, pi 的这种创建方法将暂时地代替了有相同名字、用于生成无理数 π 的近似值的内建数值函数 `pi.m`。

`x = sym('x','real')` %创建一实符号变量。若 x 有了具体的值, 则命令 `clear x` 只能清除 x 的值, 而不能改变 x 的“属性”。

`x = sym('x','unreal')` %使 x 变成一纯粹的、没有任何附加属性的符号变量。

`S = sym(A,flag)` %将一数值标量或矩阵转换成符号形式。对浮点数值的转换方法要用第二个参量 flag 来指定。其中 flag 可以是 'r'、'd'、'e'、'f'。
 'f': 代表“浮点格式”。
 'r': 代表“有理格式”(该方式为缺省转换格式)。
 'e': 代表“估计误差”。
 'd': 代表“十进制格式”。

命令 13 创建多个符号对象的快捷命令

函数 `syms`

格式 `syms arg1 arg2 ...` %定义 arg1、arg2 为符号

`syms arg1 arg2 ... real` %该命令是下列命令的简洁形式:

`arg1 = sym('arg1','real');`

`arg2 = sym('arg2','real'); ...`

`syms arg1 arg2 ... unreal` %该命令是下列命令的简洁形式:

`arg1 = sym('arg1','unreal');`

`arg2 = sym('arg2','unreal'); ...`

注: `clear x` 不能清除符号变量 x 的属性 “real”, 只能清除变量 x。要想清除该属性, 要输入: `syms x unreal` 或 `clear mex` 或 `clear all`。执行后面的两个命令后, Maple 内核将重新装



载入 MATLAB 的工作空间（这是不可取的，因为花费时间）。

例 3-56

```
>>syms x beta real %符号对象已经生成，执行下面一些操作：
>>whos
```

将显示工作空间中存在变量的详细信息：

Name	Size	Bytes	Class
beta	1x1	132	sym object
x	1x1	126	sym object

Grand total is 7 elements using 258 bytes

```
y = x + i*beta; clear x; y
```

通过上面的操作，我们看到，当 x 被清除掉后，y 的值并没有马上改变：

```
y =
x+i*beta
```

命令 14 将符号多项式转化为数值多项式

函数 **sym2poly**

格式 **c = sym2poly(s)** %返回符号多项式 s 的数值系数行向量 c。多项式自变量次数的系数按降幂排列。即行向量 c 的第一分量 c1 为多项式 s 的最高次数项的系数，c2 为第二高次数项的系数，如此类推。

例 3-57

```
>>syms x u;
>>c1 = sym2poly(3*x^3 - 2*x^2 - sqrt(5))
>>c2 = sym2poly(u^4 - 3 + 5*u^2)
```

计算结果为：

```
c1 =
3.0000 -2.0000 0 -2.2361
c2 =
1 0 5 0 -3
```

命令 15 可变精度算法

函数 **vpa**

格式 **R = vpa(A)** %用可变精度算法来计算 A 中的每一元素，使其成为有 d 位精确度的十进制数。其中 d 为命令 digits 设置的当前位数。R 中的每一元素为一符号表达式。

R = vpa(A,d)或 **R = vpa A d** %用参量 d 指定的位数(而非命令 digits 设置的位数)来表示 A 中的每一元素。R 中的每一元素为一符号表达式。

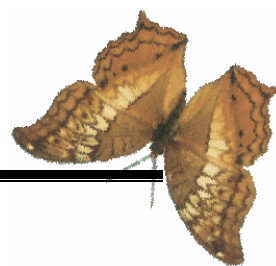
例 3-58

```
>>digits(25)
>>q = vpa(sym(sin(pi/6)))
>>p = vpa(pi)
>>gold_ratioi = vpa('(sqrt(5)-1)/2')
>>vpa pi 75
>>A = vpa(gallery(5),8)
>>B = vpa(hilb(3),5)
```

计算结果为：

```
q =
.50000000000000000000000000000000
p =
```





```

3.141592653589793238462643
gold_ratio =
    .6180339887498948482045870
ans =
3.14159265358979323846264338327950288419716939937510582097... 494459230781640629
A =
[ -9., 11., -21., 63., -252.]
[ 70., -69., 141., -421., 1684.]
[ -575., 575., -1149., 3451., -13801.]
[ 3891., -3891., 7782., -23345., 93365.]
[ 1024., -1024., 2048., -6144., 24572.]
B =
[ 1., .50000, .33333]
[ .50000, .33333, .25000]
[ .33333, .25000, .20000]

```

命令 16 符号表达式的 C 语言代码

函数 **ccode**

格式 **ccode(s)** %返回 C 语言的、用于计算符号表达式 s 的语句段落

例 3-59

```

>>syms x
>>s = taylor(exp(x));
>>ccode(s)

```

计算结果为:

```

ans =
t0 = 1.0+x+x*x/2.0+x*x*x/6.0+x*x*x*x/24.0+x*x*x*x*x/120.0;

```

注: t_0 为 e^x 在 $x=0$ 附近的计算公式 (Taylor 展式)。

命令 17 符号表达式的 Fortran 语言代码

函数 **fortran**

格式 **fortan(s)** %返回一 Fortan 语言的、用于计算符号表达式 s 的语句段落

例 3-60

```

>>syms x
>>f = taylor(sin(x));
>>F1 = fortran(f)
>>H = sym(hilb(4));
>>F2 = fortran(t*(H))

```

计算结果为:

```

F1 =
t0 = x-x**3/6+x**5/120

F2 =
T(1,1) = t      T(1,2) = t/2      T(1,3) = t/3      T(1,4) = t/4
T(2,1) = t/2    T(2,2) = t/3      T(2,3) = t/4      T(2,4) = t/5
T(3,1) = t/3    T(3,2) = t/4      T(3,3) = t/5      T(3,4) = t/6
T(4,1) = t/4    T(4,2) = t/5      T(4,3) = t/6      T(4,4) = t/7

```



第4章 概率统计

本章介绍 MATLAB 在概率统计中的若干命令和使用格式，这些命令存放于 MatlabR12\Toolbox\Stats 中。

4.1 随机数的产生

4.1.1 二项分布的随机数据的产生

命令 参数为 N 、 P 的二项随机数据

函数 **binornd**

格式 $R = \text{binornd}(N,P)$ % N 、 P 为二项分布的两个参数，返回服从参数为 N 、 P 的二项分布的随机数， N 、 P 大小相同。

$R = \text{binornd}(N,P,m)$ % m 指定随机数的个数，与 R 同维数。

$R = \text{binornd}(N,P,m,n)$ % m,n 分别表示 R 的行数和列数

例 4-1

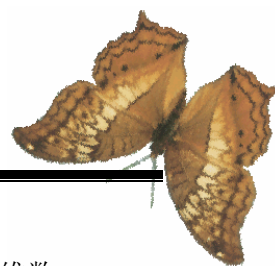
```
>> R=binornd(10,0.5)
R =
     3
>> R=binornd(10,0.5,1,6)
R =
     8     1     3     7     6     4
>> R=binornd(10,0.5,[1,10])
R =
     6     8     4     6     7     5     3     5     6     2
>> R=binornd(10,0.5,[2,3])
R =
     7     5     8
     6     5     6
>> n = 10:10:60;
>> r1 = binornd(n,1./n)
r1 =
     2     1     0     1     1     2
>> r2 = binornd(n,1./n,[1 6])
r2 =
     0     1     2     1     3     1
```

4.1.2 正态分布的随机数据的产生

命令 参数为 μ 、 σ 的正态分布的随机数据

函数 **normrnd**

格式 $R = \text{normrnd}(MU,SIGMA)$ %返回均值为 MU ，标准差为 $SIGMA$ 的正态分布的



随机数据，R 可以是向量或矩阵。

$R = \text{normrnd}(\text{MU}, \text{SIGMA}, m)$ %m 指定随机数的个数，与 R 同维数。

$R = \text{normrnd}(\text{MU}, \text{SIGMA}, m, n)$ %m,n 分别表示 R 的行数和列数

例 4-2

```
>>n1 = normrnd(1:6,1./(1:6))
n1 =
    2.1650    2.3134    3.0250    4.0879    4.8607    6.2827
>>n2 = normrnd(0,1,[1 5])
n2 =
    0.0591    1.7971    0.2641    0.8717   -1.4462
>>n3 = normrnd([1 2 3;4 5 6],0.1,2,3) %mu 为均值矩阵
n3 =
    0.9299    1.9361    2.9640
    4.1246    5.0577    5.9864
>> R=normrnd(10,0.5,[2,3]) %mu 为 10, sigma 为 0.5 的 2 行 3 列个正态随机数
R =
    9.7837    10.0627    9.4268
    9.1672    10.1438    10.5955
```

4.1.3 常见分布的随机数产生

常见分布的随机数的使用格式与上面相同

表 4-1 随机数产生函数表

函数名	调用形式	注 释
Unifrnd	unifrnd (A,B,m,n)	[A,B]上均匀分布(连续) 随机数
Unidrnd	unidrnd(N,m,n)	均匀分布(离散) 随机数
Exprnd	exprnd(Lambda,m,n)	参数为 Lambda 的指数分布随机数
Normrnd	normrnd(MU,SIGMA,m,n)	参数为 MU, SIGMA 的正态分布随机数
chi2rnd	chi2rnd(N,m,n)	自由度为 N 的卡方分布随机数
Trnd	trnd(N,m,n)	自由度为 N 的 t 分布随机数
Frnd	frnd(N ₁ , N ₂ ,m,n)	第一自由度为 N ₁ ,第二自由度为 N ₂ 的 F 分布随机数
gamrnd	gamrnd(A, B,m,n)	参数为 A, B 的 γ 分布随机数
betarnd	betarnd(A, B,m,n)	参数为 A, B 的 β 分布随机数
lognrnd	lognrnd(MU, SIGMA,m,n)	参数为 MU, SIGMA 的对数正态分布随机数
nbinrnd	nbinrnd(R, P,m,n)	参数为 R, P 的负二项式分布随机数
ncfrnd	ncfrnd(N ₁ , N ₂ , delta,m,n)	参数为 N ₁ , N ₂ , delta 的非中心 F 分布随机数
nctrnd	nctrnd(N, delta,m,n)	参数为 N, delta 的非中心 t 分布随机数
ncx2rnd	ncx2rnd(N, delta,m,n)	参数为 N, delta 的非中心卡方分布随机数
raylrnd	raylrnd(B,m,n)	参数为 B 的瑞利分布随机数
weibrnd	weibrnd(A, B,m,n)	参数为 A, B 的韦伯分布随机数
binornd	binornd(N,P,m,n)	参数为 N, p 的二项分布随机数
geornd	geornd(P,m,n)	参数为 p 的几何分布随机数
hygernd	hygernd(M,K,N,m,n)	参数为 M, K, N 的超几何分布随机数
Poissrnd	poissrnd(Lambda,m,n)	参数为 Lambda 的泊松分布随机数

4.1.4 通用函数求各分布的随机数据

命令 求指定分布的随机数

函数 random



格式 $y = \text{random}(\text{'name'}, A1, A2, A3, m, n)$ %name 的取值见表 4-2; $A1, A2, A3$ 为分布的参数; m, n 指定随机数的行和列

例 4-3 产生 12 (3 行 4 列) 个均值为 2, 标准差为 0.3 的正态分布随机数

```
>> y=random('norm',2,0.3,3,4)
y =
    2.3567    2.0524    1.8235    2.0342
    1.9887    1.9440    2.6550    2.3200
    2.0982    2.2177    1.9591    2.0178
```

4.2 随机变量的概率密度计算

4.2.1 通用函数计算概率密度函数值

命令 通用函数计算概率密度函数值

函数 pdf

格式 $Y = \text{pdf}(\text{name}, K, A)$
 $Y = \text{pdf}(\text{name}, K, A, B)$
 $Y = \text{pdf}(\text{name}, K, A, B, C)$

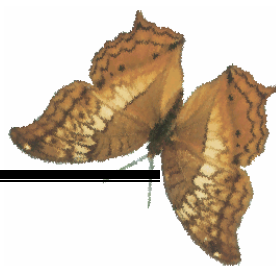
说明 返回在 $X=K$ 处、参数为 A, B, C 的概率密度值, 对于不同的分布, 参数个数是不同; name 为分布函数名, 其取值如表 4-2。

表 4-2 常见分布函数表

name 的取值		函数说明
'beta'	或 'Beta'	Beta 分布
'bino'	或 'Binomial'	二项分布
'chi2'	或 'Chisquare'	卡方分布
'exp'	或 'Exponential'	指数分布
'f'	或 'F'	F 分布
'gam'	或 'Gamma'	GAMMA 分布
'geo'	或 'Geometric'	几何分布
'hyge'	或 'Hypergeometric'	超几何分布
'logn'	或 'Lognormal'	对数正态分布
'nbin'	或 'Negative Binomial'	负二项式分布
'ncf'	或 'Noncentral F'	非中心 F 分布
'nct'	或 'Noncentral t'	非中心 t 分布
'ncx2'	或 'Noncentral Chi-square'	非中心卡方分布
'norm'	或 'Normal'	正态分布
'poiss'	或 'Poisson'	泊松分布
'rayl'	或 'Rayleigh'	瑞利分布
't'	或 'T'	T 分布
'unif'	或 'Uniform'	均匀分布
'unid'	或 'Discrete Uniform'	离散均匀分布
'weib'	或 'Weibull'	Weibull 分布

例如二项分布: 设一次试验, 事件 A 发生的概率为 p , 那么, 在 n 次独立重复试验中, 事件 A 恰好发生 K 次的概率 P_K 为: $P_K = P\{X=K\} = \text{pdf}(\text{'bino'}, K, n, p)$

例 4-4 计算正态分布 $N(0, 1)$ 的随机变量 X 在点 0.6578 的密度函数值。



解: >> pdf('norm',0.6578,0,1)
ans =
0.3213

例 4-5 自由度为 8 的卡方分布, 在点 2.18 处的密度函数值。

解: >> pdf('chi2',2.18,8)
ans =
0.0363

4.2.2 专用函数计算概率密度函数值

命令 二项分布的概率值

函数 **binopdf**

格式 binopdf(k, n, p) %等同于 pdf('bino',K,n,p), p — 每次试验事件 A 发生的概率; K—事件 A 发生 K 次; n—试验总次数

命令 泊松分布的概率值

函数 **poisspdf**

格式 poisspdf(k, Lambda) %等同于 pdf('poiss',K,Lamda)

命令 正态分布的概率值

函数 **normpdf(K,mu,sigma)** %计算参数为 $\mu = \text{mu}$, $\sigma = \text{sigma}$ 的正态分布密度函数在 K 处的值

专用函数计算概率密度函数列表如表 4-3。

表 4-3 专用函数计算概率密度函数表

函数名	调用形式	注 释
Unifpdf	unifpdf(x, a, b)	[a,b]上均匀分布(连续)概率密度在 $X=x$ 处的函数值
unidpdf	unidpdf(x,n)	均匀分布(离散)概率密度函数值
Exppdf	exppdf(x, Lambda)	参数为 Lambda 的指数分布概率密度函数值
normpdf	normpdf(x, mu, sigma)	参数为 mu, sigma 的正态分布概率密度函数值
chi2pdf	chi2pdf(x, n)	自由度为 n 的卡方分布概率密度函数值
Tpdf	tpdf(x, n)	自由度为 n 的 t 分布概率密度函数值
Fpdf	fpdf(x, n ₁ , n ₂)	第一自由度为 n ₁ , 第二自由度为 n ₂ 的 F 分布概率密度函数值
gampdf	gampdf(x, a, b)	参数为 a, b 的 γ 分布概率密度函数值
betapdf	betapdf(x, a, b)	参数为 a, b 的 β 分布概率密度函数值
lognpdf	lognpdf(x, mu, sigma)	参数为 mu, sigma 的对数正态分布概率密度函数值
nbinpdl	nbinpdl(x, R, P)	参数为 R, P 的负二项式分布概率密度函数值
Ncfpdf	ncfpdl(x, n ₁ , n ₂ , delta)	参数为 n ₁ , n ₂ , delta 的非中心 F 分布概率密度函数值
Nctpdf	nctpdf(x, n, delta)	参数为 n, delta 的非中心 t 分布概率密度函数值
ncx2pdf	ncx2pdf(x, n, delta)	参数为 n, delta 的非中心卡方分布概率密度函数值
raylpdl	raylpdl(x, b)	参数为 b 的瑞利分布概率密度函数值
weibpdf	weibpdf(x, a, b)	参数为 a, b 的韦伯分布概率密度函数值
binopdl	binopdl(x,n,p)	参数为 n, p 的二项分布的概率密度函数值
geopdl	geopdl(x,p)	参数为 p 的几何分布的概率密度函数值
hygepdf	hygepdf(x,M,K,N)	参数为 M, K, N 的超几何分布的概率密度函数值
poisspdf	poisspdf(x,Lambda)	参数为 Lambda 的泊松分布的概率密度函数值

例 4-6 绘制卡方分布密度函数在自由度分别为 1、5、15 的图形

```
>> x=0:0.1:30;  
>> y1=chi2pdf(x,1); plot(x,y1,':')  
>> hold on
```



```
>> y2=chi2pdf(x,5);plot(x,y2,'+')
>> y3=chi2pdf(x,15);plot(x,y3,'o')
>> axis([0,30,0,0.2]) %指定显示的图形区域
```

则图形为图 4-1。

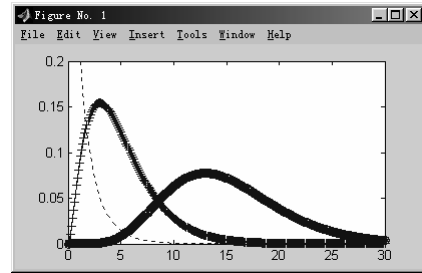


图 4-1

4.2.3 常见分布的密度函数作图

1. 二项分布

例 4-7

```
>> x = 0:10;
>> y = binopdf(x,10,0.5);
>> plot(x,y,'+')
```

2. 卡方分布

例 4-8

```
>> x = 0:0.2:15;
>> y = chi2pdf(x,4);
>> plot(x,y)
```

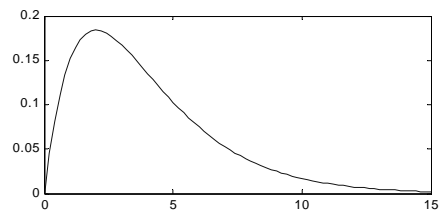
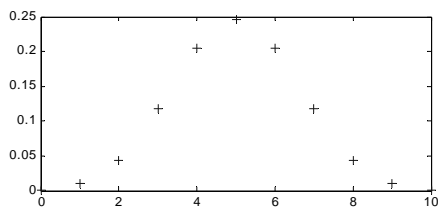


图 4-2

3. 非中心卡方分布

例 4-9

```
>> x = (0:0.1:10)';
>> p1 = ncx2pdf(x,4,2);
>> p = chi2pdf(x,4);
>> plot(x,p,'--',x,p1,'-')
```

4. 指数分布

例 4-10

```
>> x = 0:0.1:10;
>> y = exppdf(x,2);
>> plot(x,y)
```

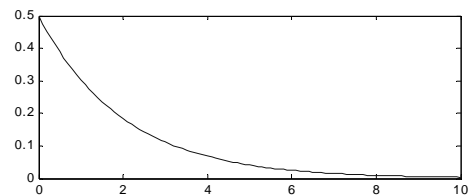
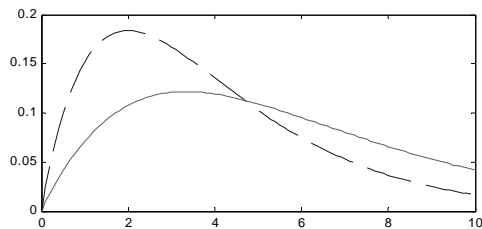
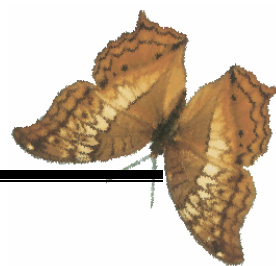


图 4-3



5. F 分布

例 4-11

```
>>x = 0:0.01:10;
>>y = fpdf(x,5,3);
>>plot(x,y)
```

6. 非中心 F 分布

例 4-12

```
>>x = (0.01:0.1:10.01)';
>>p1 = ncfpdf(x,5,20,10);
>>p = fpdf(x,5,20);
>>plot(x,p,'--',x,p1,'-')
```

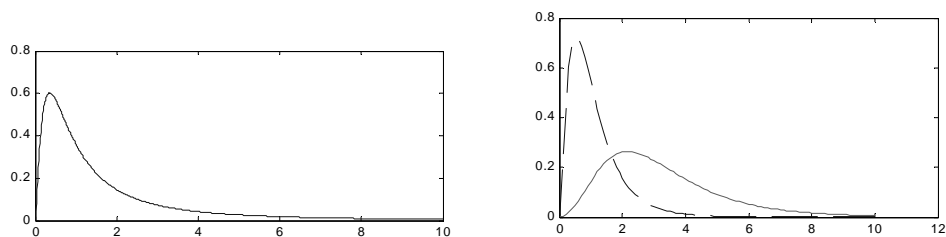


图 4-4

7. Γ 分布

例 4-13

```
>>x = gaminv((0.005:0.01:0.995),100,10);
>>y = gampdf(x,100,10);
>>y1 = normpdf(x,1000,100);
>>plot(x,y,'-',x,y1,'-')
```

8. 对数正态分布

例 4-14

```
>>x = (10:1000:125010)';
>>y = lognpdf(x,log(20000),1.0);
>>plot(x,y)
>>set(gca,'xtick',[0 30000 60000 90000 120000])
>>set(gca,'xticklabel',str2mat('0','$30,000','$60,000',...
'$90,000','$120,000'))
```

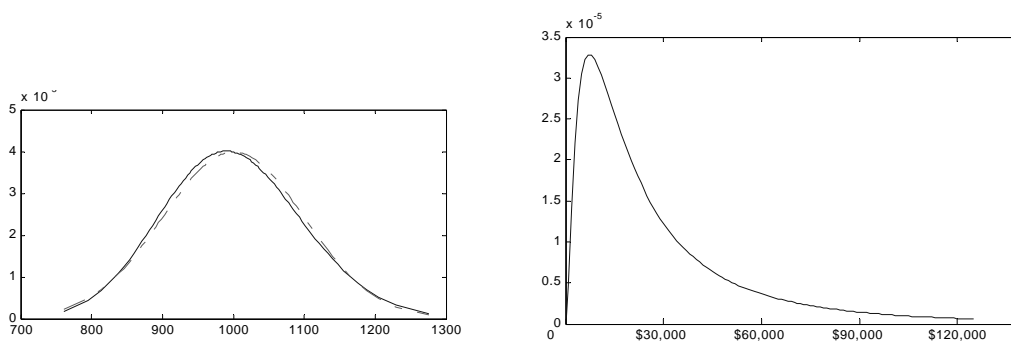


图 4-5



9. 负二项分布

例 4-15

```
>>x = (0:10);  
>>y = nbinpdf(x,3,0.5);  
>>plot(x,y,'+')
```

10. 正态分布

例 4-16

```
>>x=-3:0.2:3;  
>>y=normpdf(x,0,1);  
>>plot(x,y)
```

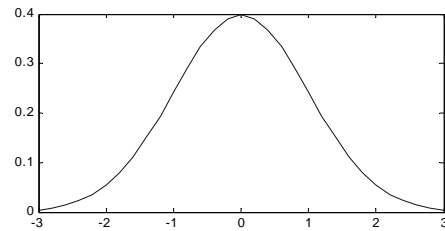
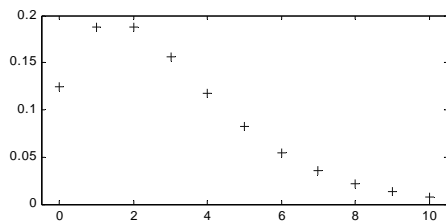


图 4-6

11. 泊松分布

例 4-17

```
>>x = 0:15;  
>>y = poisspdf(x,5);  
>>plot(x,y,'+')
```

12. 瑞利分布

例 4-18

```
>>x = [0:0.01:2];  
>>p = raylpdf(x,0.5);  
>>plot(x,p)
```

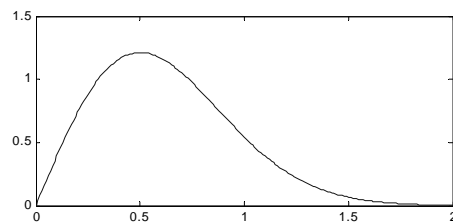
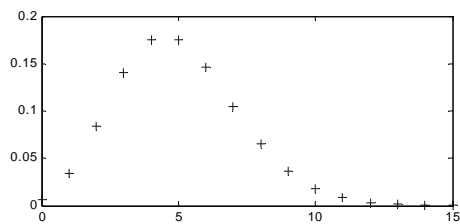


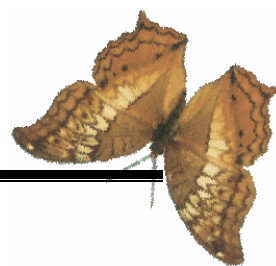
图 4-7

13. T 分布

例 4-19

```
>>x = -5:0.1:5;  
>>y = tpdf(x,5);  
>>z = normpdf(x,0,1);  
>>plot(x,y,'-',x,z,'-')
```

14. 威布尔分布



例 4-20

```
>> t=0:0.1:3;
>> y=weibpdf(t,2,2);
>> plot(y)
```

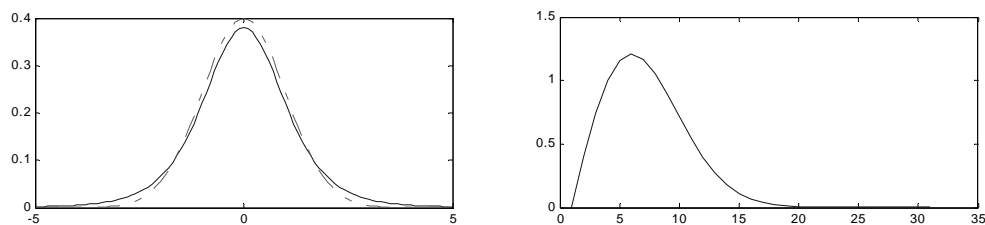


图 4-8

4.3 随机变量的累积概率值(分布函数值)

4.3.1 通用函数计算累积概率值

命令 通用函数 `cdf` 用来计算随机变量 $X \leq K$ 的概率之和 (累积概率值)

函数 `cdf`

格式 `cdf('name', K, A)`
`cdf('name', K, A, B)`
`cdf('name', K, A, B, C)`

说明 返回以 `name` 为分布、随机变量 $X \leq K$ 的概率之和的累积概率值, `name` 的取值见表 4-1 常见分布函数表

例 4-21 求标准正态分布随机变量 X 落在区间 $(-\infty, 0.4)$ 内的概率 (该值就是概率统计教材中的附表: 标准正态数值表)。

解:

```
>> cdf('norm',0.4,0,1)
ans =
    0.6554
```

例 4-22 求自由度为 16 的卡方分布随机变量落在 $[0, 6.91]$ 内的概率

```
>> cdf('chi2',6.91,16)
ans =
    0.0250
```

4.3.2 专用函数计算累积概率值 (随机变量 $X \leq K$ 的概率之和)

命令 二项分布的累积概率值

函数 `binocdf`

格式 `binocdf(k, n, p)` %n 为试验总次数, p 为每次试验事件 A 发生的概率, k 为 n 次试验中事件 A 发生的次数, 该命令返回 n 次试验中事件 A 恰好发生 k 次的概率。



命令 正态分布的累积概率值

函数 **normcdf**

格式 **normcdf(x, mu, sigma)** %返回 $F(x) = \int_{-\infty}^x p(t)dt$ 的值, μ 、 σ 为正态分布的两个参数

例 4-23 设 $X \sim N(3, 2^2)$

(1) 求 $P\{2 < X < 5\}$, $P\{-4 < X < 10\}$, $P\{|X| > 2\}$, $P\{X > 3\}$

(2) 确定 c , 使得 $P\{X > c\} = P\{X < c\}$

解 (1) $p1 = P\{2 < X < 5\}$

$p2 = P\{-4 < X < 10\}$

$p3 = P\{|X| > 2\} = 1 - P\{|X| \leq 2\}$

$p4 = P\{X > 3\} = 1 - P\{X \leq 3\}$

则有:

```
>>p1=normcdf(5,3,2)-normcdf(2,3,2)
```

```
p1 =
```

```
0.5328
```

```
>>p2=normcdf(10,3,2)-normcdf(-4,3,2)
```

```
p2 =
```

```
0.9995
```

```
>>p3=1-normcdf(2,3,2)-normcdf(-2,3,2)
```

```
p3 =
```

```
0.6853
```

```
>>p4=1-normcdf(3,3,2)
```

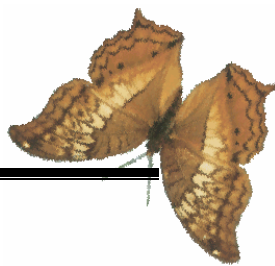
```
p4 =
```

```
0.5000
```

专用函数计算累积概率值函数列表如表 4-4。

表 4-4 专用函数的累积概率值函数表

函数名	调用形式	注 释
unifcdf	unifcdf(x, a, b)	[a,b]上均匀分布(连续)累积分布函数值 $F(x)=P\{X \leq x\}$
unidcdf	unidcdf(x,n)	均匀分布(离散)累积分布函数值 $F(x)=P\{X \leq x\}$
expcdf	expcdf(x, Lambda)	参数为 Lambda 的指数分布累积分布函数值 $F(x)=P\{X \leq x\}$
normcdf	normcdf(x, mu, sigma)	参数为 mu, sigma 的正态分布累积分布函数值 $F(x)=P\{X \leq x\}$
chi2cdf	chi2cdf(x, n)	自由度为 n 的卡方分布累积分布函数值 $F(x)=P\{X \leq x\}$
tcdf	tcdf(x, n)	自由度为 n 的 t 分布累积分布函数值 $F(x)=P\{X \leq x\}$
fcdf	fcdf(x, n1, n2)	第一自由度为 n1, 第二自由度为 n2 的 F 分布累积分布函数值
gamcdf	gamcdf(x, a, b)	参数为 a, b 的 γ 分布累积分布函数值 $F(x)=P\{X \leq x\}$
betacdf	betacdf(x, a, b)	参数为 a, b 的 β 分布累积分布函数值 $F(x)=P\{X \leq x\}$
logncdf	logncdf(x, mu, sigma)	参数为 mu, sigma 的对数正态分布累积分布函数值
nbincdf	nbincdf(x, R, P)	参数为 R, P 的负二项式分布累积分布函数值 $F(x)=P\{X \leq x\}$
ncfcdf	ncfcdf(x, n1, n2, delta)	参数为 n1, n2, delta 的非中心 F 分布累积分布函数值
nctcdf	nctcdf(x, n, delta)	参数为 n, delta 的非中心 t 分布累积分布函数值 $F(x)=P\{X \leq x\}$
ncx2cdf	ncx2cdf(x, n, delta)	参数为 n, delta 的非中心卡方分布累积分布函数值
raylcdf	raylcdf(x, b)	参数为 b 的瑞利分布累积分布函数值 $F(x)=P\{X \leq x\}$
weibcdf	weibcdf(x, a, b)	参数为 a, b 的韦伯分布累积分布函数值 $F(x)=P\{X \leq x\}$
binocdf	binocdf(x,n,p)	参数为 n, p 的二项分布的累积分布函数值 $F(x)=P\{X \leq x\}$
geocdf	geocdf(x,p)	参数为 p 的几何分布的累积分布函数值 $F(x)=P\{X \leq x\}$
hygecdf	hygecdf(x,M,K,N)	参数为 M, K, N 的超几何分布的累积分布函数值
poisscdf	poisscdf(x,Lambda)	参数为 Lambda 的泊松分布的累积分布函数值 $F(x)=P\{X \leq x\}$



说明 累积概率函数就是分布函数 $F(x)=P\{X \leq x\}$ 在 x 处的值。

4.4 随机变量的逆累积分布函数

MATLAB 中的逆累积分布函数是已知 $F(x) = P\{X \leq x\}$ ，求 x 。

逆累积分布函数值的计算有两种方法

4.4.1 通用函数计算逆累积分布函数值

命令 **icdf** 计算逆累积分布函数

格式 **icdf** ('name', P, a₁, a₂, a₃)

说明 返回分布为 name，参数为 a₁, a₂, a₃，累积概率值为 P 的临界值，这里 name 与前面表 4.1 相同。

如果 $P = \text{cdf}(\text{'name'}, x, a_1, a_2, a_3)$ ，则 $x = \text{icdf}(\text{'name'}, P, a_1, a_2, a_3)$

例 4-24 在标准正态分布表中，若已知 $\Phi(x)=0.975$ ，求 x

解：>> x=icdf('norm',0.975,0,1)

x =
1.9600

例 4-25 在 χ^2 分布表中，若自由度为 10， $\alpha=0.975$ ，求临界值 Lambda。

解：因为表中给出的值满足 $P\{\chi^2 > \lambda\} = \alpha$ ，而逆累积分布函数 icdf 求满足 $P\{\chi^2 < \lambda\} = \alpha$ 的临界值 λ 。所以，这里的 α 取为 0.025，即

>> Lambda=icdf('chi2',0.025,10)
Lambda =
3.2470

例 4-26 在假设检验中，求临界值问题：

已知： $\alpha = 0.05$ ，查自由度为 10 的双边界检验 t 分布临界值

>> lambda=icdf('t',0.025,10)
lambda =
-2.2281

4.4.2 专用函数-inv 计算逆累积分布函数

命令 正态分布逆累积分布函数

函数 **norminv**

格式 **X=norminv(p,mu,sigma)** %p 为累积概率值，mu 为均值，sigma 为标准差，X 为临界值，满足： $p=P\{X \leq x\}$ 。

例 4-27 设 $X \sim N(3, 2^2)$ ，确定 c 使得 $P\{X > c\} = P\{X < c\}$ 。

解：由 $P\{X > c\} = P\{X < c\}$ 得， $P\{X > c\} = P\{X < c\} = 0.5$ ，所以

>> X=norminv(0.5, 3, 2)
X =
3

关于常用临界值函数可查下表 4-5。



表 4-5 常用临界值函数表

函数名	调用形式	注 释
unifinv	$x = \text{unifinv}(p, a, b)$	均匀分布(连续)逆累积分布函数 ($P=P\{X \leq x\}$, 求 x)
unidinv	$x = \text{unidinv}(p, n)$	均匀分布(离散)逆累积分布函数, x 为临界值
expinv	$x = \text{expinv}(p, \text{Lambda})$	指数分布逆累积分布函数
norminv	$x = \text{Norminv}(x, \mu, \sigma)$	正态分布逆累积分布函数
chi2inv	$x = \text{chi2inv}(x, n)$	卡方分布逆累积分布函数
tinv	$x = \text{tinv}(x, n)$	t 分布累积分布函数
finv	$x = \text{finv}(x, n_1, n_2)$	F 分布逆累积分布函数
gaminv	$x = \text{gaminv}(x, a, b)$	γ 分布逆累积分布函数
betainv	$x = \text{betainv}(x, a, b)$	β 分布逆累积分布函数
logninv	$x = \text{logninv}(x, \mu, \sigma)$	对数正态分布逆累积分布函数
nbinv	$x = \text{nbinv}(x, R, P)$	负二项式分布逆累积分布函数
ncfinv	$x = \text{ncfinv}(x, n_1, n_2, \text{delta})$	非中心 F 分布逆累积分布函数
nctinv	$x = \text{nctinv}(x, n, \text{delta})$	非中心 t 分布逆累积分布函数
ncx2inv	$x = \text{ncx2inv}(x, n, \text{delta})$	非中心卡方分布逆累积分布函数
raylinv	$x = \text{raylinv}(x, b)$	瑞利分布逆累积分布函数
weibinv	$x = \text{weibinv}(x, a, b)$	韦伯分布逆累积分布函数
binoinv	$x = \text{binoinv}(x, n, p)$	二项分布的逆累积分布函数
geoinv	$x = \text{geoinv}(x, p)$	几何分布的逆累积分布函数
hygeinv	$x = \text{hygeinv}(x, M, K, N)$	超几何分布的逆累积分布函数
poissinv	$x = \text{poissinv}(x, \text{Lambda})$	泊松分布的逆累积分布函数

例 4-28 公共汽车门的高度是按成年男子与车门顶碰头的机会不超过 1%设计的。设男子身高 X (单位: cm) 服从正态分布 $N(175, 36)$, 求车门的最低高度。

解: 设 h 为车门高度, X 为身高

求满足条件 $P\{X > h\} \leq 0.01$ 的 h , 即 $P\{X < h\} \geq 0.99$, 所以

```
>>h=norminv(0.99, 175, 6)
```

```
h =
```

```
188.9581
```

例 4-29 卡方分布的逆累积分布函数的应用

在 MATLAB 的编辑器下建立 M 文件如下:

```
n=5; a=0.9; %n 为自由度, a 为置信水平或累积概率
```

```
x_a=chi2inv(a,n); %x_a 为临界值
```

```
x=0:0.1:15; yd_c=chi2pdf(x,n); %计算  $\chi^2(5)$  的概率密度函数值, 供绘图用
```

```
plot(x,yd_c,'b'), hold on %绘密度函数图形
```

```
xxf=0:0.1:x_a; yyf=chi2pdf(xxf,n); %计算  $[0, x_a]$  上的密度函数值, 供填色用
```

```
fill([xxf,x_a], [yyf,0], 'g') %填色, 其中: 点  $(x_a, 0)$  使得填色区域封闭
```

```
text(x_a*1.01,0.01,num2str(x_a)) %标注临界值点
```

```
text(10,0.10,['fontsize{16}X~{\chi}^2(4)'])
```

```
%图中标注
```

```
text(1.5,0.05,['fontsize{22}alpha=0.9']) %图中标注
```

结果显示如图 4-9。

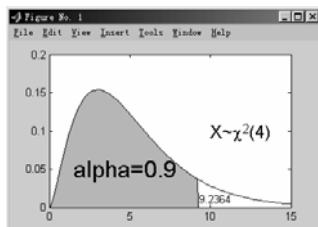
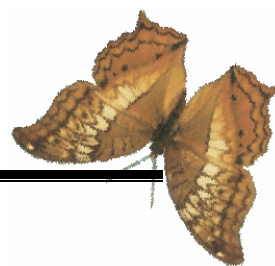


图 4-9



4.5 随机变量的数字特征

4.5.1 平均值、中值

命令 利用 mean 求算术平均值

格式 mean(X) %X 为向量，返回 X 中各元素的平均值
 mean(A) %A 为矩阵，返回 A 中各列元素的平均值构成的向量
 mean(A,dim) %在给出的维数内的平均值

说明 X 为向量时，算术平均值的数学含义是 $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ ，即样本均值。

例 4-30

```
>> A=[1 3 4 5;2 3 4 6;1 3 1 5]
A =
     1     3     4     5
     2     3     4     6
     1     3     1     5
>> mean(A)
ans =
    1.3333    3.0000    3.0000    5.3333
>> mean(A,1)
ans =
    1.3333    3.0000    3.0000    5.3333
```

命令 忽略 NaN 计算算术平均值

格式 nanmean(X) %X 为向量，返回 X 中除 NaN 外元素的算术平均值。
 nanmean(A) %A 为矩阵，返回 A 中各列除 NaN 外元素的算术平均值向量。

例 4-31

```
>> A=[1 2 3;nan 5 2;3 7 nan]
A =
     1     2     3
    NaN     5     2
     3     7    NaN
>> nanmean(A)
ans =
    2.0000    4.6667    2.5000
```

命令 利用 median 计算中值（中位数）

格式 median(X) %X 为向量，返回 X 中各元素的中位数。
 median(A) %A 为矩阵，返回 A 中各列元素的中位数构成的向量。
 median(A,dim) %求给出的维数内的中位数

例 4-32

```
>> A=[1 3 4 5;2 3 4 6;1 3 1 5]
A =
     1     3     4     5
     2     3     4     6
     1     3     1     5
>> median(A)
ans =
```



命令 忽略 NaN 计算中位数

格式 `nanmedian(X)` %X 为向量, 返回 X 中除 NaN 外元素的中位数。
`nanmedian(A)` %A 为矩阵, 返回 A 中各列除 NaN 外元素的中位数向量。

例 4-33

```
>> A=[1 2 3;nan 5 2;3 7 nan]
A =
     1     2     3
    NaN     5     2
     3     7    NaN
>> nanmedian(A)
ans =
    2.0000    5.0000    2.5000
```

命令 利用 `geomean` 计算几何平均数

格式 `M=geomean(X)` %X 为向量, 返回 X 中各元素的几何平均数。
`M=geomean(A)` %A 为矩阵, 返回 A 中各列元素的几何平均数构成的向量。

说明 几何平均数的数学含义是 $M = \left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}$, 其中: 样本数据非负, 主要用于对数正

态分布。

例 4-34

```
>> B=[1 3 4 5]
B =
     1     3     4     5
>> M=geomean(B)
M =
    2.7832
>> A=[1 3 4 5;2 3 4 6;1 3 1 5]
A =
     1     3     4     5
     2     3     4     6
     1     3     1     5
>> M=geomean(A)
M =
    1.2599    3.0000    2.5198    5.3133
```

命令 利用 `harmmean` 求调和平均值

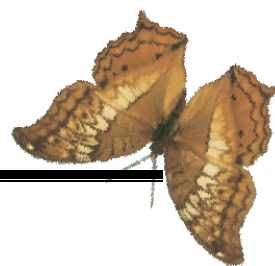
格式 `M=harmmean(X)` %X 为向量, 返回 X 中各元素的调和平均值。
`M=harmmean(A)` %A 为矩阵, 返回 A 中各列元素的调和平均值构成的向量。

说明 调和平均值的数学含义是 $M = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}}$, 其中: 样本数据非 0, 主要用于严重偏斜

分布。

例 4-35

```
>> B=[1 3 4 5]
B =
     1     3     4     5
>> M=harmmean(B)
M =
```



```

2.2430
>> A=[1 3 4 5;2 3 4 6;1 3 1 5]
A =
     1     3     4     5
     2     3     4     6
     1     3     1     5
>> M=harmmean(A)
M =
    1.2000    3.0000    2.0000    5.2941

```

4.5.2 数据比较

命令 排序

格式 `Y=sort(X)` %X 为向量，返回 X 按由小到大排序后的向量。
`Y=sort(A)` %A 为矩阵，返回 A 的各列按由小到大排序后的矩阵。
`[Y,I]=sort(A)` %Y 为排序的结果，I 中元素表示 Y 中对应元素在 A 中位置。
`sort(A,dim)` %在给定的维数 dim 内排序

说明 若 X 为复数，则通过|X|排序。

例 4-36

```

>> A=[1 2 3;4 5 2;3 7 0]
A =
     1     2     3
     4     5     2
     3     7     0
>> sort(A)
ans =
     1     2     0
     3     5     2
     4     7     3
>> [Y,I]=sort(A)
Y =
     1     2     0
     3     5     2
     4     7     3
I =
     1     1     3
     3     2     2
     2     3     1

```

命令 按行方式排序

函数 `sortrows`

格式 `Y=sortrows(A)` %A 为矩阵，返回矩阵 Y，Y 按 A 的第 1 列由小到大，以行方式排序后生成的矩阵。
`Y=sortrows(A, col)` %按指定列 col 由小到大进行排序
`[Y,I]=sortrows(A, col)` %Y 为排序的结果，I 表示 Y 中第 col 列元素在 A 中位置。

说明 若 X 为复数，则通过|X|的大小排序。

例 4-37

```

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

```



```
3    7    0
>> sortrows(A)
ans =
1    2    3
3    7    0
4    5    2
>> sortrows(A,1)
ans =
1    2    3
3    7    0
4    5    2
>> sortrows(A,3)
ans =
3    7    0
4    5    2
1    2    3
>> sortrows(A,[3 2])
ans =
3    7    0
4    5    2
1    2    3
>> [Y,I]=sortrows(A,3)
Y =
3    7    0
4    5    2
1    2    3
I =
3
2
1
```

命令 求最大值与最小值之差

函数 **range**

格式 **Y=range(X)** %X 为向量，返回 X 中的最大值与最小值之差。

Y=range(A) %A 为矩阵，返回 A 中各列元素的最大值与最小值之差。

例 4-38

```
>> A=[1 2 3;4 5 2;3 7 0]
A =
1    2    3
4    5    2
3    7    0
>> Y=range(A)
Y =
3    5    3
```

4.5.3 期望

命令 计算样本均值

函数 **mean**

格式 用法与前面一样

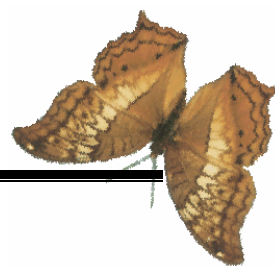
例 4-39 随机抽取 6 个滚珠测得直径如下：（直径：mm）

14.70 15.21 14.90 14.91 15.32 15.32

试求样本平均值

解：**>>X=[14.70 15.21 14.90 14.91 15.32 15.32];**

>>mean(X) %计算样本均值



则结果如下:

```
ans =
    15.0600
```

命令 由分布律计算均值

利用 sum 函数计算

例 4-40 设随机变量 X 的分布律为:

X	-2	-1	0	1	2
P	0.3	0.1	0.2	0.1	0.3

求 $E(X)$ $E(X^2-1)$

解: 在 Matlab 编辑器中建立 M 文件如下:

```
X=[-2 -1 0 1 2];
p=[0.3 0.1 0.2 0.1 0.3];
EX=sum(X.*p)
Y=X.^2-1
EY=sum(Y.*p)
```

运行后结果如下:

```
EX =
    0
Y =
    3     0    -1     0     3
EY =
    1.6000
```

4.5.4 方差

命令 求样本方差

函数 var

格式 $D=\text{var}(X)$ % $\text{var}(X)=s^2=\frac{1}{n-1}\sum_{i=1}^n(x_i-\bar{X})^2$,若 X 为向量,则返回向量的样本方差。

$D=\text{var}(A)$ %A 为矩阵,则 D 为 A 的列向量的样本方差构成的行向量。

$D=\text{var}(X, 1)$ %返回向量(矩阵)X 的简单方差(即置前因子为 $\frac{1}{n}$ 的方差)

$D=\text{var}(X, w)$ %返回向量(矩阵)X 的以 w 为权重的方差

命令 求标准差

函数 std

格式 $\text{std}(X)$ %返回向量(矩阵)X 的样本标准差(置前因子为 $\frac{1}{n-1}$)即:

$$\text{std} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n x_i - \bar{X}}$$

$\text{std}(X, 1)$ %返回向量(矩阵)X 的标准差(置前因子为 $\frac{1}{n}$)

$\text{std}(X, 0)$ %与 $\text{std}(X)$ 相同

$\text{std}(X, \text{flag}, \text{dim})$ %返回向量(矩阵)中维数为 dim 的标准差值,其中 flag=0

时,置前因子为 $\frac{1}{n-1}$;否则置前因子为 $\frac{1}{n}$ 。



例 4-41 求下列样本的样本方差和样本标准差，方差和标准差

14.70 15.21 14.90 15.32 15.32

解：

```
>>X=[14.7 15.21 14.9 14.91 15.32 15.32];
>>DX=var(X,1)      %方差
DX =
    0.0559
>>sigma=std(X,1)    %标准差
sigma =
    0.2364
>>DX1=var(X)        %样本方差
DX1 =
    0.0671
>>sigma1=std(X)     %样本标准差
sigma1 =
    0.2590
```

命令 忽略 NaN 的标准差

函数 **nanstd**

格式 **y = nanstd(X)** %若 X 为含有元素 NaN 的向量，则返回除 NaN 外的元素的标准差，若 X 为含元素 NaN 的矩阵，则返回各列除 NaN 外的标准差构成的向量。

例 4-42

```
>> M=magic(3)      %产生 3 阶魔方阵
M =
     8     1     6
     3     5     7
     4     9     2
>> M([1 6 8])=[NaN NaN NaN]    %替换 3 阶魔方阵中第 1、6、8 个元素为 NaN
M =
    NaN     1     6
     3     5    NaN
     4    NaN     2
>> y=nanstd(M)      %求忽略 NaN 的各列向量的标准差
y =
    0.7071    2.8284    2.8284
>> X=[1 5];        %忽略 NaN 的第 2 列元素
>> y2=std(X)        %验证第 2 列忽略 NaN 元素的标准差
y2 =
    2.8284
```

命令 样本的偏斜度

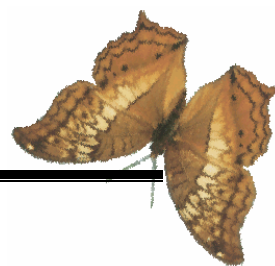
函数 **skewness**

格式 **y = skewness(X)** %X 为向量，返回 X 的元素的偏斜度；X 为矩阵，返回 X 各列元素的偏斜度构成的行向量。

y = skewness(X,flag) %flag=0 表示偏斜纠正，flag=1（默认）表示偏斜不纠正。

说明 偏斜度样本数据关于均值不对称的一个测度，如果偏斜度为负，说明均值左边的数据比均值右边的数据更散；如果偏斜度为正，说明均值右边的数据比均值左边的数据更散，

因而正态分布的偏斜度为 0；偏斜度是这样定义的： $y = \frac{E(x - \mu)^3}{\sigma^3}$



其中： μ 为 x 的均值， σ 为 x 的标准差， $E(\cdot)$ 为期望值算子

例 4-43

```
>> X=randn([5,4])
X =
    0.2944    0.8580   -0.3999    0.6686
   -1.3362    1.2540    0.6900    1.1908
    0.7143   -1.5937    0.8156   -1.2025
    1.6236   -1.4410    0.7119   -0.0198
   -0.6918    0.5711    1.2902   -0.1567
>> y=skewness(X)
y =
   -0.0040   -0.3136   -0.8865   -0.2652
>> y=skewness(X,0)
y =
   -0.0059   -0.4674   -1.3216   -0.3954
```

4.5.5 常见分布的期望和方差

命令 均匀分布（连续）的期望和方差

函数 **unifstat**

格式 $[M,V] = \text{unifstat}(A,B)$ %A、B 为标量时，就是区间上均匀分布的期望和方差，
A、B 也可为向量或矩阵，则 M、V 也是向量或矩阵。

例 4-44

```
>> a = 1:6; b = 2.*a;
>> [M,V] = unifstat(a,b)
M =
    1.5000    3.0000    4.5000    6.0000    7.5000    9.0000
V =
    0.0833    0.3333    0.7500    1.3333    2.0833    3.0000
```

命令 正态分布的期望和方差

函数 **normstat**

格式 $[M,V] = \text{normstat}(MU,SIGMA)$ %MU、SIGMA 可为标量也可为向量或矩阵，
则 $M=MU$ ， $V=SIGMA^2$ 。

例 4-45

```
>> n=1:4;
>> [M,V]=normstat(n'*n,n'*n)
M =
     1     2     3     4
     2     4     6     8
     3     6     9    12
     4     8    12    16
V =
     1     4     9    16
     4    16    36    64
     9    36    81   144
    16    64   144   256
```

命令 二项分布的均值和方差

函数 **binostat**

格式 $[M,V] = \text{binostat}(N,P)$ %N、P 为二项分布的两个参数，可为标量也可为向量
或矩阵。



例 4-46

```

>>n = logspace(1,5,5)
n =
    10    100   1000  10000 100000
>>[M,V] = binostat(n,1./n)
M =
    1     1     1     1     1
V =
    0.9000    0.9900    0.9990    0.9999    1.0000
>>[m,v] = binostat(n,1/2)
m =
     5    50    500   5000  50000
v =
1.0e+04 *
    0.0003    0.0025    0.0250    0.2500    2.5000

```

常见分布的期望和方差见下表 4-6。

表 4-6 常见分布的均值和方差

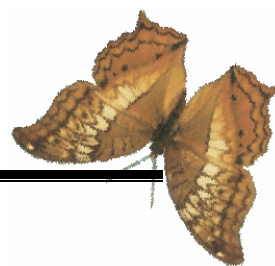
函数名	调用形式	注 释
unifstat	[M,V]=unifstat (a, b)	均匀分布(连续)的期望和方差, M 为期望, V 为方差
unidstat	[M,V]=unidstat (n)	均匀分布(离散)的期望和方差
expstat	[M,V]=expstat (p, Lambda)	指数分布的期望和方差
normstat	[M,V]=normstat(mu,sigma)	正态分布的期望和方差
chi2stat	[M,V]=chi2stat (x, n)	卡方分布的期望和方差
tstat	[M,V]=tstat (n)	t 分布的期望和方差
fstat	[M,V]=fstat (n ₁ , n ₂)	F 分布的期望和方差
gamstat	[M,V]=gamstat (a, b)	γ 分布的期望和方差
betastat	[M,V]=betastat (a, b)	β 分布的期望和方差
lognstat	[M,V]=lognstat (mu, sigma)	对数正态分布的期望和方差
nbinstat	[M,V]=nbinstat (R, P)	负二项式分布的期望和方差
ncfstat	[M,V]=ncfstat (n ₁ , n ₂ , delta)	非中心 F 分布的期望和方差
nctstat	[M,V]=nctstat (n, delta)	非中心 t 分布的期望和方差
ncx2stat	[M,V]=ncx2stat (n, delta)	非中心卡方分布的期望和方差
raylstat	[M,V]=raylstat (b)	瑞利分布的期望和方差
Weibstat	[M,V]=weibstat (a, b)	韦伯分布的期望和方差
Binostat	[M,V]=binostat (n,p)	二项分布的期望和方差
Geostat	[M,V]=geostat (p)	几何分布的期望和方差
hygestat	[M,V]=hygestat (M,K,N)	超几何分布的期望和方差
Poisstat	[M,V]=poisstat (Lambda)	泊松分布的期望和方差

4.5.6 协方差与相关系数

命令 协方差

函数 cov

格式 cov(X) %求向量 X 的协方差
 cov(A) %求矩阵 A 的协方差矩阵, 该协方差矩阵的对角线元素是 A 的各列的方差, 即: $\text{var}(A)=\text{diag}(\text{cov}(A))$ 。
 cov(X,Y) %X,Y 为等长列向量, 等同于 $\text{cov}([X \ Y])$ 。



例 4-47

```
>> X=[0 -1 1]';Y=[1 2 2]';
>> C1=cov(X)      %X 的协方差
C1 =
    1
>> C2=cov(X,Y)    %列向量 X、Y 的协方差矩阵，对角线元素为各列向量的方差
C2 =
    1.0000    0
         0    0.3333
>> A=[1 2 3;4 0 -1;1 7 3]
A =
     1     2     3
     4     0    -1
     1     7     3
>> C1=cov(A)      %求矩阵 A 的协方差矩阵
C1 =
    3.0000   -4.5000   -4.0000
   -4.5000   13.0000    6.0000
   -4.0000    6.0000    5.3333
>> C2=var(A(:,1)) %求 A 的第 1 列向量的方差
C2 =
     3
>> C3=var(A(:,2)) %求 A 的第 2 列向量的方差
C3 =
    13
>> C4=var(A(:,3))
C4 =
    5.3333
```

命令 相关系数

函数 **corrcoef**

格式 **corrcoef(X,Y)** %返回列向量 X,Y 的相关系数，等同于 **corrcoef([X Y])**。
corrcoef(A) %返回矩阵 A 的列向量的相关系数矩阵

例 4-48

```
>> A=[1 2 3;4 0 -1;1 3 9]
A =
     1     2     3
     4     0    -1
     1     3     9
>> C1=corrcoef(A) %求矩阵 A 的相关系数矩阵
C1 =
    1.0000   -0.9449   -0.8030
   -0.9449    1.0000    0.9538
   -0.8030    0.9538    1.0000
>> C1=corrcoef(A(:,2),A(:,3)) %求 A 的第 2 列与第 3 列列向量的相关系数矩阵
C1 =
    1.0000    0.9538
    0.9538    1.0000
```

4.6 统计作图

4.6.1 正整数的频率表

命令 正整数的频率表

**函数 tabulate**

格式 table = tabulate(X) %X 为正整数构成的向量, 返回 3 列: 第 1 列中包含 X 的值
第 2 列为这些值的个数, 第 3 列为这些值的频率。

例 4-49

```
>> A=[1 2 2 5 6 3 8]
A =
     1     2     2     5     6     3     8
>> tabulate(A)
Value    Count    Percent
     1         1    14.29%
     2         2    28.57%
     3         1    14.29%
     4         0     0.00%
     5         1    14.29%
     6         1    14.29%
     7         0     0.00%
     8         1    14.29%
```

4.6.2 经验累积分布函数图形**函数 cdfplot**

格式 cdfplot(X) %作样本 X (向量) 的累积分布函数图形
h = cdfplot(X) %h 表示曲线的句柄
[h,stats] = cdfplot(X) %stats 表示样本的一些特征

例 4-50

```
>> X=normrnd(0,1,50,1);
>> [h,stats]=cdfplot(X)
h =
    3.0013
stats =
    min: -1.8740    %样本最小值
    max:  1.6924    %最大值
    mean:  0.0565    %平均值
    median: 0.1032    %中间值
    std:  0.7559    %样本标准差
```

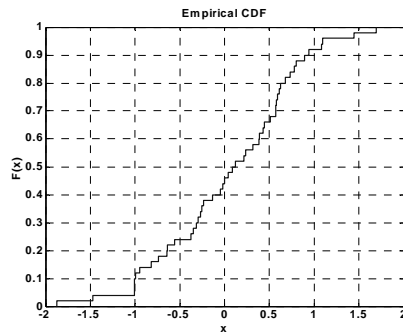


图 4-10

4.6.3 最小二乘拟合直线**函数 lsline**

格式 lsline %最小二乘拟合直线
h = lsline %h 为直线的句柄

例 4-51

```
>> X=[2 3.4 5.6 8 11 12.3 13.8 16 18.8 19.9];
>> plot(X,'+')
>> lsline
```

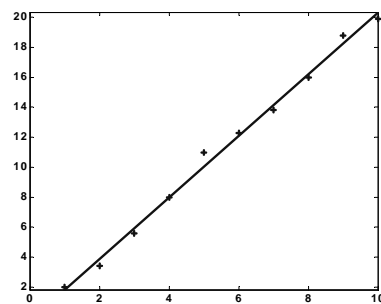
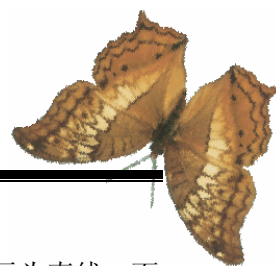


图 4-11

4.6.4 绘制正态分布概率图形**函数 normplot**

格式 normplot(X) %若 X 为向量, 则显示正态分布概率图形, 若 X 为矩阵, 则显示每一列的正态分布概率图形。



`h = normplot(X)` %返回绘图直线的句柄

说明 样本数据在图中用“+”显示；如果数据来自正态分布，则图形显示为直线，而其它分布可能在图中产生弯曲。

例 4-53

```
>> X=normrnd(0,1,50,1);
>> normplot(X)
```

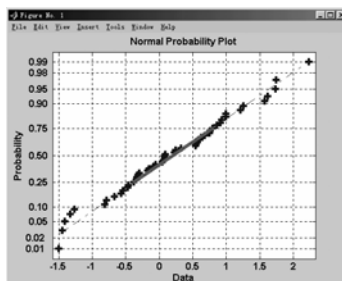


图 4-12

4.6.5 绘制威布尔(Weibull)概率图形

函数 `weibplot`

格式 `weibplot(X)` %若 X 为向量，则显示威布尔(Weibull)概率图形，若 X 为矩阵，则显示每一列的威布尔概率图形。

`h = weibplot(X)` %返回绘图直线的柄

说明 绘制威布尔(Weibull)概率图形的目的是用图解法估计来自威布尔分布的数据 X ，如果 X 是威布尔分布数据，其图形是直线的，否则图形中可能产生弯曲。

例 4-54

```
>> r = weibrnd(1.2,1.5,50,1);
>> weibplot(r)
```

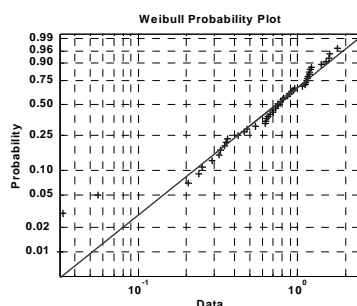


图 4-13

4.6.6 样本数据的盒图

函数 `boxplot`

格式 `boxplot(X)` %产生矩阵 X 的每一列的盒图和“须”图，“须”是从盒的尾部延



伸出来，并表示盒外数据长度的线，如果“须”的外面没有数据，则在“须”的底部有一个点。

`boxplot(X,notch)` %当 `notch=1` 时，产生一凹盒图，`notch=0` 时产生一矩箱图。

`boxplot(X,notch,'sym')` %`sym` 表示图形符号，默认值为“+”。

`boxplot(X,notch,'sym',vert)` %当 `vert=0` 时，生成水平盒图，`vert=1` 时，生成竖直盒图（默认值 `vert=1`）。

`boxplot(X,notch,'sym',vert,whis)` %`whis` 定义“须”图的长度，默认值为 1.5，若 `whis=0` 则 `boxplot` 函数通过绘制 `sym` 符号图来显示盒外的所有数据值。

例 4-55

```
>>x1 = normrnd(5,1,100,1);
>>x2 = normrnd(6,1,100,1);
>>x = [x1 x2];
>> boxplot(x,1,'g+',1,0)
```

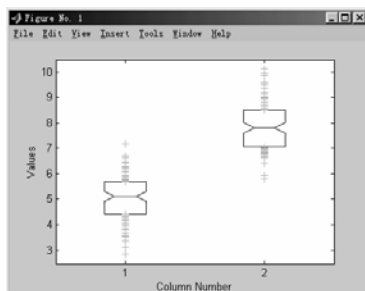


图 4-14

4.6.7 给当前图形加一条参考线

函数 **refline**

格式 `refline(slope,intercept)` % `slope` 表示直线斜率，`intercept` 表示截距
`refline(slope)` `slope=[a b]`，图中加一条直线： $y=b+ax$ 。

例 4-56

```
>>y = [3.2 2.6 3.1 3.4 2.4 2.9 3.0 3.3 3.2 2.1 2.6];
>>plot(y,'+')
>>refline(0,3)
```

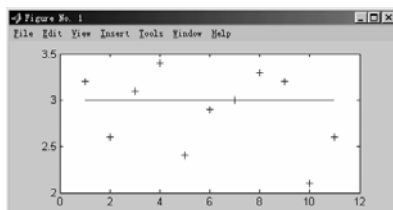


图 4-15

4.6.8 在当前图形中加入一条多项式曲线

函数 **refcurve**



格式 `h = refcurve(p)` %在图中加入一条多项式曲线, `h` 为曲线的环柄, `p` 为多项式系数向量, `p=[p1,p2,p3,...,pn]`, 其中 `p1` 为最高幂项系数。

例 4-57 火箭的高度与时间图形, 加入一条理论高度曲线, 火箭初速为 100m/秒。

```
>>h = [85 162 230 289 339 381 413 437 452 458 456 440 400 356];
>>plot(h,'+')
>>refcurve([-4.9 100 0])
```

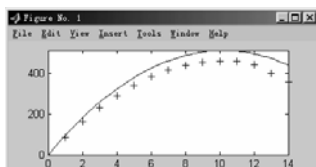


图 4-16

4.6.9 样本的概率图形

函数 `capaplot`

格式 `p = capaplot(data,specs)` %`data` 为所给样本数据, `specs` 指定范围, `p` 表示在指定范围内的概率。

说明 该函数返回来自于估计分布的随机变量落在指定范围内的概率

例 4-58

```
>> data=normrnd(0,1,30,1);
>> p=capaplot(data,[-2,2])
p =
    0.9199
```

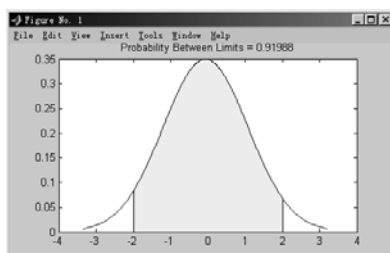


图 4-17

4.6.10 附加有正态密度曲线的直方图

函数 `histfit`

格式 `histfit(data)` %`data` 为向量, 返回直方图和正态曲线。

`histfit(data,nbins)` % `nbins` 指定 bar 的个数, 缺省时为 `data` 中数据个数的平方根。

例 4-59

```
>>r = normrnd(10,1,100,1);
>>histfit(r)
```

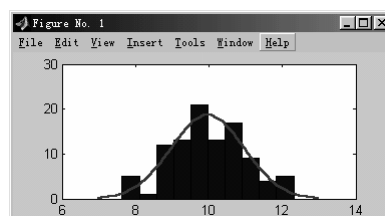


图 4-18



4.6.11 在指定的界线之间画正态密度曲线

函数 **normspec**

格式 `p = normspec(specs,mu,sigma)` %specs 指定界线, mu,sigma 为正态分布的参数 p 为样本落在上、下界之间的概率。

例 4-60

```
>>normspec([10 Inf],11.5,1.25)
```

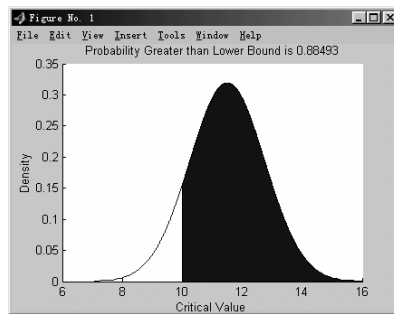


图 4-19

4.7 参数估计

4.7.1 常见分布的参数估计

命令 β 分布的参数 a 和 b 的最大似然估计值和置信区间

函数 **betafit**

格式 `PHAT=betafit(X)`

`[PHAT,PCI]=betafit(X,ALPHA)`

说明 PHAT 为样本 X 的 β 分布的参数 a 和 b 的估计量

PCI 为样本 X 的 β 分布参数 a 和 b 的置信区间, 是一个 2×2 矩阵, 其第 1 例为参数 a 的置信下界和上界, 第 2 例为 b 的置信下界和上界, ALPHA 为显著水平, $(1-\alpha) \times 100\%$ 为置信度。

例 4-61 随机产生 100 个 β 分布数据, 相应的分布参数真值为 4 和 3。则 4 和 3 的最大似然估计值和置信度为 99%的置信区间为:

解:

```
>>X = betarnd(4,3,100,1); %产生 100 个  $\beta$  分布的随机数
```

```
>>[PHAT,PCI] = betafit(X,0.01) %求置信度为 99%的置信区间和参数 a、b 的估计值
```

结果显示

```
PHAT =
    3.9010    2.6193
PCI =
    2.5244    1.7488
    5.2776    3.4898
```




说明 估计值 3.9010 的置信区间是[2.5244 5.2776], 估计值 2.6193 的置信区间是[1.7488 3.4898]。

命令 正态分布的参数估计

函数 **normfit**

格式 [muhat,sigmahat,muci,sigmaci] = normfit(X)

[muhat,sigmahat,muci,sigmaci] = normfit(X,alpha)

说明 muhat,sigmahat 分别为正态分布的参数 μ 和 σ 的估计值, muci,sigmaci 分别为置信区间, 其置信度为 $(1-\alpha)\times 100\%$; alpha 给出显著水平 α , 缺省时默认为 0.05, 即置信度为 95%。

例 4-62 有两组(每组 100 个元素)正态随机数据, 其均值为 10, 均方差为 2, 求 95%的置信区间和参数估计值。

解: >>r = normrnd(10,2,100,2); %产生两列正态随机数据

>>[mu,sigma,muci,sigmaci] = normfit(r)

则结果为

```
mu =
    10.1455    10.0527    %各列的均值的估计值
sigma =
     1.9072     2.1256    %各列的均方差的估计值
muci =
     9.7652     9.6288
    10.5258    10.4766
sigmaci =
     1.6745     1.8663
     2.2155     2.4693
```

说明 muci, sigmaci 中各列分别为原随机数据各列估计值的置信区间, 置信度为 95%。

例 4-63 分别使用金球和铂球测定引力常数

(1) 用金球测定观察值为: 6.683 6.681 6.676 6.678 6.679 6.672

(2) 用铂球测定观察值为: 6.661 6.661 6.667 6.667 6.664

设测定值总体为 $N(\mu, \sigma^2)$, μ 和 σ 为未知。对(1)、(2)两种情况分别求 μ 和 σ 的置信度为

0.9 的置信区间。

解: 建立 M 文件: LX0833.m

X=[6.683 6.681 6.676 6.678 6.679 6.672];

Y=[6.661 6.661 6.667 6.667 6.664];

[mu,sigma,muci,sigmaci]=normfit(X,0.1) %金球测定的估计

[MU,SIGMA,MUCI,SIGMACI]=normfit(Y,0.1) %铂球测定的估计

运行后结果显示如下:

```
mu =
    6.6782
sigma =
    0.0039
muci =
    6.6750
    6.6813
sigmaci =
    0.0026
    0.0081
MU =
```



```

        6.6640
SIGMA =
        0.0030
MUCI =
        6.6611
        6.6669
SIGMACI =
        0.0019
        0.0071

```

由上可知，金球测定的 μ 估计值为 6.6782，置信区间为[6.6750, 6.6813]；

σ 的估计值为 0.0039，置信区间为[0.0026, 0.0081]。

泊球测定的 μ 估计值为 6.6640，置信区间为[6.6611, 6.6669]；

σ 的估计值为 0.0030，置信区间为[0.0019, 0.0071]。

命令 利用 mle 函数进行参数估计

函数 mle

格式 phat=mle('dist', X) %返回用 dist 指定分布的最大似然估计值

[phat, pci]=mle('dist', X) %置信度为 95%

[phat, pci]=mle('dist', X, alpha) %置信度由 alpha 确定

[phat, pci]=mle('dist', X, alpha, pl) %仅用于二项分布，pl 为试验次数。

说明 dist 为分布函数名，如：beta(β 分布)、bino (二项分布) 等，X 为数据样本，alpha 为显著水平 α ， $(1-\alpha) \times 100\%$ 为置信度。

例 4-64

```

>> X=binornd(20,0.75) %产生二项分布的随机数
X =
    16
>> [p,pci]=mle('bino',X,0.05,20) %求概率的估计值和置信区间，置信度为 95%
p =
    0.8000
pci =
    0.5634
    0.9427

```

常用分布的参数估计函数

表 4-7 参数估计函数表

函数名	调用形式	函数说明
binofit	PHAT=binofit(X, N) [PHAT, PCI]=binofit(X,N) [PHAT, PCI]=binofit(X, N, ALPHA)	二项分布的概率的最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的参数估计和置信区间
poissfit	Lambdahat=poissfit(X) [Lambdahat, Lambdaci]=poissfit(X) [Lambdahat, Lambdaci]=poissfit(X, ALPHA)	泊松分布的参数最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的 λ 参数和置信区间
normfit	[muhat,sigmahat,muci,sigmaci]=normfit(X) [muhat,sigmahat,muci,sigmaci]=normfit(X, ALPHA)	正态分布的最大似然估计，置信度为 95% 返回水平 α 的期望、方差值和置信区间
betafit	PHAT=betafit(X) [PHAT, PCI]=betafit(X, ALPHA)	返回 β 分布参数 a 和 b 的最大似然估计 返回最大似然估计值和水平 α 的置信区间
unifit	[ahat,bhat]=unifit(X) [ahat,bhat,ACI,BCI]=unifit(X) [ahat,bhat,ACI,BCI]=unifit(X, ALPHA)	均匀分布参数的最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的参数估计和置信区间
expfit	muhat=expfit(X) [muhat,muci]=expfit(X) [muhat,muci]=expfit(X,alpha)	指数分布参数的最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的参数估计和置信区间
gamfit	phat=gamfit(X)	γ 分布参数的最大似然估计



	[phat,pci] = gamfit(X) [phat,pci] = gamfit(X,alpha)	置信度为 95%的参数估计和置信区间 返回最大似然估计值和水平 α 的置信区间
weibfit	phat = weibfit(X) [phat,pci] = weibfit(X) [phat,pci] = weibfit(X,alpha)	韦伯分布参数的最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的参数估计及其区间估计
Mle	phat = mle('dist',data) [phat,pci] = mle('dist',data) [phat,pci] = mle('dist',data,alpha) [phat,pci] = mle('dist',data,alpha,pl)	分布函数名为 dist 的最大似然估计 置信度为 95%的参数估计和置信区间 返回水平 α 的最大似然估计值和置信区间 仅用于二项分布, pl 为试验总次数

说明 各函数返回已给数据向量 X 的参数最大似然估计值和置信度为 $(1-\alpha) \times 100\%$ 的置信区间。 α 的默认值为 0.05, 即置信度为 95%。

4.7.2 非线性模型置信区间预测

命令 高斯—牛顿法的非线性最小二乘法数据拟合

函数 **nlinfit**

格式 $\text{beta} = \text{nlinfit}(X, y, \text{FUN}, \text{beta0})$ %返回在 FUN 中描述的非线性函数的系数。FUN 为用户提供形如 $\hat{y} = f(\beta, X)$ 的函数, 该函数返回已给初始参数估计值 β 和自变量 X 的 y 的预测值 \hat{y} 。

$[\text{beta}, r, J] = \text{nlinfit}(X, y, \text{FUN}, \text{beta0})$ %beta 为拟合系数, r 为残差, J 为 Jacobi 矩阵, beta0 为初始预测值。

说明 若 X 为矩阵, 则 X 的每一列为自变量的取值, y 是一个相应的列向量。如果 FUN 中使用了 @, 则表示函数的柄。

例 4-65 调用 MATLAB 提供的数据文件 reaction.mat

```
>>load reaction
>>betafit = nlinfit(reactants,rate,@hougen,beta)
betafit =
    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
```

命令 非线性模型的参数估计的置信区间

函数 **nlparci**

格式 $\text{ci} = \text{nlparci}(\text{beta}, r, J)$ %返回置信度为 95%的置信区间, beta 为非线性最小二乘法估计的参数值, r 为残差, J 为 Jacobian 矩阵。nlparci 可以用 nlinfit 函数的输出作为其输入。

例 4-66 调用 MATLAB 中的数据 reaction。

```
>>load reaction
>>[beta,resids,J] = nlinfit(reactants,rate,'hougen',beta)
beta =
    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
resids =
    0.1321
   -0.1642
   -0.0909
```



```

0.0310
0.1142
0.0498
-0.0262
0.3115
-0.0292
0.1096
0.0716
-0.1501
-0.3026
J =
6.8739 -90.6536 -57.8640 -1.9288 0.1614
3.4454 -48.5357 -13.6240 -1.7030 0.3034
5.3563 -41.2099 -26.3042 -10.5217 1.5095
1.6950 0.1091 0.0186 0.0279 1.7913
2.2967 -35.5658 -6.0537 -0.7567 0.2023
11.8670 -89.5655 -170.1745 -8.9566 0.4400
4.4973 -14.4262 -11.5409 -9.3770 2.5744
4.1831 -41.7896 -16.8937 -5.7794 1.0082
11.8286 -51.3721 -154.1164 -27.7410 1.5001
9.1514 -25.5948 -76.7844 -30.7138 2.5790
3.3373 0.0900 0.0720 0.1080 3.5269
9.3663 -102.0611 -107.4327 -3.5811 0.2200
4.7512 -24.4631 -16.3087 -10.3002 2.1141
>>ci = nlparci(beta,resids,J)
ci =
-0.7467 3.2519
-0.0377 0.1632
-0.0312 0.1113
-0.0609 0.2857
-0.7381 3.1208

```

命令 非线性拟合和显示交互图形

函数 **nlintool**

格式 **nlintool(x,y,FUN,beta0)** %返回数据(x,y)的非线性曲线的预测图形, 它用 2 条红色曲线预测全局置信区间。**beta0** 为参数的初始预测值, 置信度为 95%。

nlintool(x,y,FUN,beta0,alpha) %置信度为 $(1-\alpha) \times 100\%$

例 4-67 调用 MATLAB 数据

```

>> load reaction
>> nlintool(reactants,rate,'hougen',beta)

```

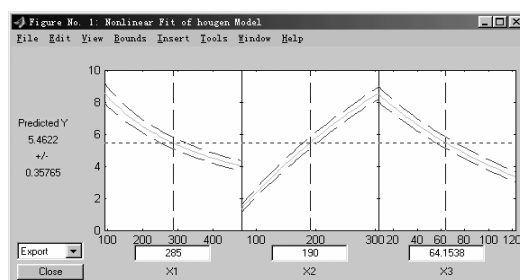
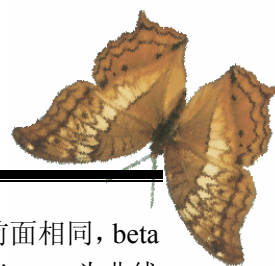


图 4-20

命令 非线性模型置信区间预测

函数 **nlpredci**



格式 `ypred = nlpredci(FUN,inputs,beta,r,J)` % `ypred` 为预测值, `FUN` 与前面相同, `beta` 为给出的适当参数, `r` 为残差, `J` 为 Jacobian 矩阵, `inputs` 为非线性函数中的独立变量的矩阵值。

`[ypred,delta] = nlpredci(FUN,inputs,beta,r,J)` % `delta` 为非线性最小二乘法估计的置信区间长度的一半, 当 `r` 长度超过 `beta` 的长度并且 `J` 的列满秩时, 置信区间的计算是有效的。 `[ypred-delta,ypred+delta]` 为置信度为 95% 的不同步置信区间。

`ypred = nlpredci(FUN,inputs,beta,r,J,alpha,'simopt','predopt')` % 控制置信区间的类型, 置信度为 $100(1-\alpha)\%$ 。 '`simopt`' = 'on' 或 'off' (默认值) 分别表示同步或不同步置信区间。 '`predopt`' = 'curve' (默认值) 表示输入函数值的置信区间, '`predopt`' = 'observation' 表示新响应值的置信区间。 `nlpredci` 可以用 `nlinfit` 函数的输出作为其输入。

例 4-68 续前例, 在 [100 300 80] 处的预测函数值 `ypred` 和置信区间一半宽度 `delta`

```
>> load reaction
>> [beta,resids,J] = nlinfit(reactants,rate,@hougen,beta);
>> [ypred,delta] = nlpredci(@hougen,[100 300 80],beta,resids,J)
```

结果为:

```
ypred =
    10.9113
delta =
    0.3195
```

命令 非负最小二乘

函数 `nnls` (该函数已被函数 `lsqnonneg` 代替, 在 6.0 版中使用 `nnls` 将产生警告信息)

格式 `x = nnls(A,b)` % 最小二乘法判断方程 $A \times x = b$ 的解, 返回在 $x \geq 0$ 的条件下使得 $\|A \times x - b\|$ 最小的向量 `x`, 其中 `A` 和 `b` 必须为实矩阵或向量。

`x = nnls(A,b,tol)` % `tol` 为指定的误差

`[x,w] = nnls(A,b)` % 当 `x` 中元素 $x_i = 0$ 时, $w_i \leq 0$, 当 $x_i > 0$ 时 $w_i \geq 0$ 。

`[x,w] = nnls(A,b,tol)`

例 4-69

```
>> A=[0.0372 0.2869;0.6861 0.7071;0.6233 0.6245;0.6344 0.6170];
>> b=[0.8587 0.1781 0.0747 0.8405]';
>> x=nnls(A,b)
```

Warning: NNLS is obsolete and has been replaced by LSQNONNEG.

NNLS now calls LSQNONNEG which uses the following syntax:

`[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT,LAMBDA]`

`=lsqnonneg(A,b,X0, Options) ;`

Use OPTIMSET to define optimization options, or type

'edit nnls' to view the code used here. NNLS will be

removed in the future; please use NNLS with the new syntax.

```
x =
```

```
0
0.6929
```

命令 有非负限制的最小二乘

函数 `lsqnonneg`



格式 `x = lsqnonneg(C,d)` %返回在 $x \geq 0$ 的条件下使得 $\|C \times x - d\|$ 最小的向量 x , 其中 C 和 d 必须为实矩阵或向量。

`x = lsqnonneg(C,d,x0)` % x_0 为初始点, $x_0 \geq 0$

`x = lsqnonneg(C,d,x0,options)` %options 为指定的优化参数, 参见 options 函数。

`[x,resnorm] = lsqnonneg(...)` %resnorm 表示 $\text{norm}(C \times x - d)^2$ 的残差

`[x,resnorm,residual] = lsqnonneg(...)` %residual 表示 $C \times x - d$ 的残差

例 4-70

```
>> A=[0.0372 0.2869;0.6861 0.7071;0.6233 0.6245;0.6344 0.6170];
```

```
>> b=[0.8587 0.1781 0.0747 0.8405]';
```

```
>> [x,resnorm,residual] = lsqnonneg(A,b)
```

```
x =  
      0  
    0.6929  
resnorm =  
    0.8315  
residual =  
    0.6599  
   -0.3119  
   -0.3580  
    0.4130
```

4.7.3 对数似然函数

命令 负 β 分布的对数似然函数

函数 **Betalike**

格式 `logL=betalike(params,data)` %返回负 β 分布的对数似然函数, params 为向量[a, b], 是 β 分布的参数, data 为样本数据。

`[logL,info]=betalike(params,data)` %返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值, 那么 info 的对角元素为相应参数的渐近方差。

说明 betalike 是 β 分布最大似然估计的实用函数。似然函数假设数据样本中, 所有的元素相互独立。因为 betalike 返回负 β 对数似然函数, 用 fmins 函数最小化 betalike 与最大似然估计的功能是相同的。

例 4-71 本例所取的数据是随机产生的 β 分布数据。

```
>>r = betarnd(3,3,100,1);
```

```
>>[logL,info] = betalike([2.1234,3.4567],r)
```

```
logL =  
   -12.4340
```

```
info =  
    0.1185    0.1364  
    0.1364    0.2061
```

命令 负 γ 分布的对数似然估计

函数 **Gamlike**

格式 `logL=gamlike(params,data)` %返回由给定样本数据 data 确定的 γ 分布的参数为 params (即[a, b]) 的负对数似然函数值

`[logL,info]=gamlike(params,data)` %返回 Fisher 逆信息矩阵 info。如果 params



中输入的参数是极大似然估计值,那么 info 的对角元素为相应参数的渐近方差。

说明 gamlike 是 γ 分布的最大似然估计函数。因为 gamlike 返回 γ 对数似然函数值,故用 fmins 函数将 gamlike 最小化后,其结果与最大似然估计是相同的。

例 4-72

```
>>r=gamrnd(2,3,100,1);
>>[logL,info]=gamlike([2.4212, 2.5320],r)
logL =
    275.4602
info =
    0.0453    -0.0538
   -0.0538     0.0867
```

命令 负正态分布的对数似然函数

函数 normlike

格式 logL=normlike(params,data) %返回由给定样本数据data确定的、负正态分布的、参数为params(即[mu, sigma])的对数似然函数值。

[logL,info]=normlike(params,data) %返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值,那么 info 的对角元素为相应参数的渐近方差。

命令 威布尔分布的对数似然函数

函数 Weiblike

格式 logL = weiblike(params,data) %返回由给定样本数据 data 确定的、威布尔分布的、参数为 params(即[a, b])的对数似然函数值。

[logL,info]=weiblike(params,data) %返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值,那么 info 的对角元素为相应参数的渐近方差。

说明 威布尔分布的负对数似然函数定义为

$$-\log L = -\log \prod_{i=1}^n f(a,b|x_i) = -\sum_{i=1}^n \log f(a,b|x_i)$$

例 4-73

```
>>r=weibrnd(0.4,0.98,100,1);
>>[logL,info]=weiblike([0.1342,0.9876],r)
logL =
    237.6682
info =
    0.0004    -0.0002
   -0.0002     0.0078
```

4.8 假设检验

4.8.1 σ^2 已知, 单个正态总体的均值 μ 的假设检验 (U 检验法)

函数 ztest

格式 h = ztest(x,m,sigma) % x 为正态总体的样本, m 为均值 μ_0 , sigma 为标准差,



显著性水平为 0.05(默认值)

$h = ztest(x, m, sigma, alpha)$ %显著性水平为 α

$[h, sig, ci, zval] = ztest(x, m, sigma, alpha, tail)$ %sig 为观察值的概率, 当 sig 为小概率时则对原假设提出质疑, ci 为真正均值 μ 的 $1-\alpha$ 置信区间, zval 为统计量的值。

说明 若 $h=0$, 表示在显著性水平 α 下, 不能拒绝原假设;

若 $h=1$, 表示在显著性水平 α 下, 可以拒绝原假设。

原假设: $H_0: \mu = \mu_0 = m$,

若 $tail=0$, 表示备择假设: $H_1: \mu \neq \mu_0 = m$ (默认, 双边检验);

$tail=1$, 表示备择假设: $H_1: \mu > \mu_0 = m$ (单边检验);

$tail=-1$, 表示备择假设: $H_1: \mu < \mu_0 = m$ (单边检验)。

例 4-74 某车间用一台包装机包装葡萄糖, 包得的袋装糖重是一个随机变量, 它服从正态分布。当机器正常时, 其均值为 0.5 公斤, 标准差为 0.015。某日开工后检验包装机是否正常, 随机地抽取所包装的糖 9 袋, 称得净重为 (公斤)

0.497, 0.506, 0.518, 0.524, 0.498, 0.511, 0.52, 0.515, 0.512

问机器是否正常?

解: 总体 μ 和 σ 已知, 该问题是当 σ^2 为已知时, 在水平 $\alpha = 0.05$ 下, 根据样本值判断 $\mu = 0.5$ 还是 $\mu \neq 0.5$ 。为此提出假设:

原假设: $H_0: \mu = \mu_0 = 0.5$

备择假设: $H_1: \mu \neq 0.5$

$\gg X = [0.497, 0.506, 0.518, 0.524, 0.498, 0.511, 0.52, 0.515, 0.512];$

$\gg [h, sig, ci, zval] = ztest(X, 0.5, 0.015, 0.05, 0)$

结果显示为

$h =$

1

$sig =$

0.0248

%样本观察值的概率

$ci =$

0.5014 0.5210

%置信区间, 均值 0.5 在此区间之外

$zval =$

2.2444

%统计量的值

结果表明: $h=1$, 说明在水平 $\alpha = 0.05$ 下, 可拒绝原假设, 即认为包装机工作不正常。

4.8.2 σ^2 未知, 单个正态总体的均值 μ 的假设检验(t 检验法)

函数 **ttest**

格式 $h = ttest(x, m)$ % x 为正态总体的样本, m 为均值 μ_0 , 显著性水平为 0.05

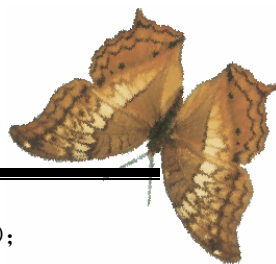
$h = ttest(x, m, alpha)$ % α 为给定显著性水平

$[h, sig, ci] = ttest(x, m, alpha, tail)$ %sig 为观察值的概率, 当 sig 为小概率时则对原假设提出质疑, ci 为真正均值 μ 的 $1-\alpha$ 置信区间。

说明 若 $h=0$, 表示在显著性水平 α 下, 不能拒绝原假设;

若 $h=1$, 表示在显著性水平 α 下, 可以拒绝原假设。

原假设: $H_0: \mu = \mu_0 = m$,



若 $\text{tail}=0$, 表示备择假设: $H_1: \mu \neq \mu_0 = m$ (默认, 双边检验);

$\text{tail}=1$, 表示备择假设: $H_1: \mu > \mu_0 = m$ (单边检验);

$\text{tail}=-1$, 表示备择假设: $H_1: \mu < \mu_0 = m$ (单边检验)。

例 4-75 某种电子元件的寿命 X (以小时计) 服从正态分布, μ 、 σ^2 均未知。现测得 16 只元件的寿命如下

159 280 101 212 224 379 179 264 222 362 168 250
149 260 485 170

问是否有理由认为元件的平均寿命大于 225 (小时)?

解: 未知 σ^2 , 在水平 $\alpha = 0.05$ 下检验假设: $H_0: \mu < \mu_0 = 225$, $H_1: \mu > 225$

```
>> X=[159 280 101 212 224 379 179 264 222 362 168 250 149 260 485 170];
```

```
>> [h,sig,ci]=ttest(X,225,0.05,1)
```

结果显示为:

```
h =
```

```
0
```

```
sig =
```

```
0.2570
```

```
ci =
```

```
198.2321
```

```
Inf
```

```
%均值 225 在该置信区间内
```

结果表明: $H=0$ 表示在水平 $\alpha = 0.05$ 下应该接受原假设 H_0 , 即认为元件的平均寿命不大于 225 小时。

4.8.3 两个正态总体均值差的检验 (t 检验)

两个正态总体方差未知但等方差时, 比较两正态总体样本均值的假设检验

函数 `ttest2`

格式 `[h,sig,ci]=ttest2(X,Y)` %X, Y 为两个正态总体的样本, 显著性水平为 0.05

`[h,sig,ci]=ttest2(X,Y,alpha)` %alpha 为显著性水平

`[h,sig,ci]=ttest2(X,Y,alpha,tail)` %sig 为当原假设为真时得到观察值的概率, 当 sig 为小概率时则对原假设提出质疑, ci 为真正均值 μ 的 $1-\alpha$ 置信区间。

说明 若 $h=0$, 表示在显著性水平 α 下, 不能拒绝原假设;

若 $h=1$, 表示在显著性水平 α 下, 可以拒绝原假设。

原假设: $H_0: \mu_1 = \mu_2$, (μ_1 为 X 为期望值, μ_2 为 Y 的期望值)

若 $\text{tail}=0$, 表示备择假设: $H_1: \mu_1 \neq \mu_2$ (默认, 双边检验);

$\text{tail}=1$, 表示备择假设: $H_1: \mu_1 > \mu_2$ (单边检验);

$\text{tail}=-1$, 表示备择假设: $H_1: \mu_1 < \mu_2$ (单边检验)。

例 4-76 在平炉上进行一项试验以确定改变操作方法的建议是否会增加钢的产率, 试验是在同一只平炉上进行的。每炼一炉钢时除操作方法外, 其他条件都尽可能做到相同。先用标准方法炼一炉, 然后用建议的新方法炼一炉, 以后交替进行, 各炼 10 炉, 其产率分别为

(1) 标准方法: 78.1 72.4 76.2 74.3 77.4 78.4 76.0 75.5 76.7 77.3

(2) 新方法: 79.1 81.0 77.3 79.1 80.0 79.1 79.1 77.3 80.2 82.1

设这两个样本相互独立, 且分别来自正态总体 $N(\mu_1, \sigma^2)$ 和 $N(\mu_2, \sigma^2)$, μ_1 、 μ_2 、 σ^2 均未知。问建议的新操作方法能否提高产率? (取 $\alpha = 0.05$)



解：两个总体方差不变时，在水平 $\alpha = 0.05$ 下检验假设： $H_0: \mu_1 = \mu_2$, $H_1: \mu_1 < \mu_2$

```
>> X=[78.1 72.4 76.2 74.3 77.4 78.4 76.0 75.5 76.7 77.3];
>> Y=[79.1 81.0 77.3 79.1 80.0 79.1 79.1 77.3 80.2 82.1];
>> [h,sig,ci]=ttest2(X,Y,0.05,-1)
```

结果显示为：

```
h =
    1
sig =
    2.1759e-004    %说明两个总体均值相等的概率很小
ci =
    -Inf    -1.9083
```

结果表明： $H=1$ 表示在水平 $\alpha = 0.05$ 下，应该拒绝原假设，即认为建议的新操作方法提高了产率，因此，比原方法好。

4.8.4 两个总体一致性的检验——秩和检验

函数 **ranksum**

格式 $p = \text{ranksum}(x,y,\alpha)$ % x 、 y 为两个总体的样本，可以不等长， α 为显著性水平

$[p,h] = \text{ranksum}(x,y,\alpha)$ % h 为检验结果， $h=0$ 表示 X 与 Y 的总体差别不显著 $h=1$ 表示 X 与 Y 的总体差别显著

$[p,h,stats] = \text{ranksum}(x,y,\alpha)$ % $stats$ 中包括：**ranksum** 为秩和统计量的值以及 **zval** 为过去计算 p 的正态统计量的值

说明 P 为两个总体样本 X 和 Y 为一致的显著性概率，若 P 接近于 0，则不一致较明显。

例 4-77 某商店为了确定向公司 A 或公司 B 购买某种商品，将 A 和 B 公司以往的各次进货的次品率进行比较，数据如下所示，设两样本独立。问两公司的商品的质量有无显著差异。设两公司的商品的次品的密度最多只差一个平移，取 $\alpha = 0.05$ 。

A: 7.0 3.5 9.6 8.1 6.2 5.1 10.4 4.0 2.0 10.5

B: 5.7 3.2 4.1 11.0 9.7 6.9 3.6 4.8 5.6 8.4 10.1 5.5 12.3

解：设 μ_A 、 μ_B 分别为 A、B 两个公司的商品次品率总体的均值。则该问题为在水平 $\alpha = 0.05$ 下检验假设： $H_0: \mu_A = \mu_B$, $H_1: \mu_A \neq \mu_B$

```
>> A=[7.0 3.5 9.6 8.1 6.2 5.1 10.4 4.0 2.0 10.5];
>> B=[5.7 3.2 4.1 11.0 9.7 6.9 3.6 4.8 5.6 8.4 10.1 5.5 12.3];
>> [p,h,stats]=ranksum(A,B,0.05)
```

结果为：

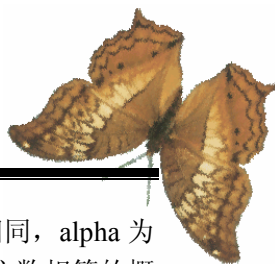
```
p =
    0.8041
h =
    0
stats =
    zval: -0.2481
    ranksum: 116
```

结果表明：一方面，两样本总体均值相等的概率为 0.8041，不接近于 0；另一方面， $H=0$ 也说明可以接受原假设 H_0 ，即认为两个公司的商品的质量无明显差异。

4.8.5 两个总体中位数相等的假设检验——符号秩检验

函数 **signrank**





格式 `p = signrank(X,Y,alpha)` % X、Y 为两个总体的样本，长度必须相同，alpha 为显著性水平，P 两个样本 X 和 Y 的中位数相等的概率，p 接近于 0 则可对原假设质疑。

`[p,h] = signrank(X,Y,alpha)` % h 为检验结果：h=0 表示 X 与 Y 的中位数之差不显著，h=1 表示 X 与 Y 的中位数之差显著。

`[p,h,stats] = signrank(x,y,alpha)` % stats 中包括：signrank 为符号秩统计量的值以及 zval 为过去计算 p 的正态统计量的值。

例 4-78 两个正态随机样本的中位数相等的假设检验

```
>> x=normrnd(0,1,20,1);
>> y=normrnd(0,2,20,1);
>> [p,h,stats]=signrank(x,y,0.05)
p =
    0.3703
h =
     0
stats =
    zval: -0.8960
    signedrank: 81
```

结果表明：h=0 表示 X 与 Y 的中位数之差不显著

4.8.6 两个总体中位数相等的假设检验——符号检验

函数 `signtest`

格式 `p=signtest(X, Y, alpha)` % X、Y 为两个总体的样本，长度必须相同，alpha 为显著性水平，P 两个样本 X 和 Y 的中位数相等的概率，p 接近于 0 则可对原假设质疑。

`[p, h]=signtest(X, Y, alpha)` % h 为检验结果：h=0 表示 X 与 Y 的中位数之差不显著，h=1 表示 X 与 Y 的中位数之差显著。

`[p,h,stats] = signtest(X,Y,alpha)` % stats 中 sign 为符号统计量的值

例 4-79 两个正态随机样本的中位数相等的假设检验

```
>> X=normrnd(0,1,20,1);
>> Y=normrnd(0,2,20,1);
>> [p,h,stats]=signtest(X,Y,0.05)
p =
    0.2632
h =
     0
stats =
    sign: 7
```

结果表明：h=0 表示 X 与 Y 的中位数之差不显著

4.8.7 正态分布的拟合优度测试

函数 `jbtest`

格式 `H = jbtest(X)` %对输入向量 X 进行 Jarque-Bera 测试，显著性水平为 0.05。

`H = jbtest(X,alpha)` %在水平 alpha 而非 5%下施行 Jarque-Bera 测试，alpha 在 0 和 1 之间。

`[H,P,JBSTAT,CV] = jbtest(X,alpha)` %P 为接受假设的概率值，P 越接近于 0，则



可以拒绝是正态分布的原假设; JBSTAT 为测试统计量的值, CV 为是否拒绝原假设的临界值。

说明 H 为测试结果, 若 $H=0$, 则可以认为 X 是服从正态分布的; 若 $H=1$, 则可以否定 X 服从正态分布。X 为大样本, 对于小样本用 lillietest 函数。

例 4-80 调用 MATLAB 中关于汽车重量的数据, 测试该数据是否服从正态分布

```
>> load carsmall
>> [h,p,j,cv]=jbtest(Weight)
h =
    1
p =
    0.0267
j =
    7.2448
cv =
    5.9915
```

说明 $p=2.67\%$ 表示应该拒绝服从正态分布的假设; $h=1$ 也可否定服从正态分布; 统计量的值 $j=7.2448$ 大于接受假设的临界值 $cv=5.9915$, 因而拒绝假设(测试水平为 5%)。

4.8.8 正态分布的拟合优度测试

函数 **lillietest**

格式 $H = \text{lillietest}(X)$ %对输入向量 X 进行 Lilliefors 测试, 显著性水平为 0.05。

$H = \text{lillietest}(X, \alpha)$ %在水平 α 而非 5%下施行 Lilliefors 测试, α 在 0.01 和 0.2 之间。

$[H,P,LSTAT,CV] = \text{lillietest}(X, \alpha)$ %P 为接受假设的概率值, P 越接近于 0, 则可以拒绝是正态分布的原假设; LSTAT 为测试统计量的值, CV 为是否拒绝原假设的临界值。

说明 H 为测试结果, 若 $H=0$, 则可以认为 X 是服从正态分布的; 若 $H=1$, 则可以否定 X 服从正态分布。

例 4-81

```
>> Y=chi2rnd(10,100,1);
>> [h,p,l,cv]=lillietest(Y)
h =
    1
p =
    0.0175
l =
    0.1062
cv =
    0.0886
```

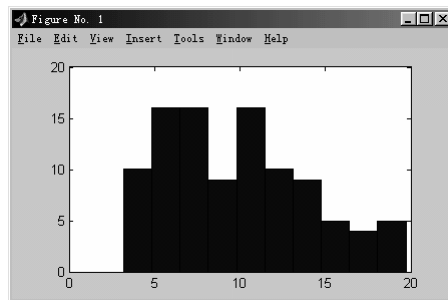


图 4-21

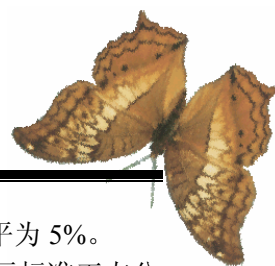
说明 $h=1$ 表示拒绝正态分布的假设; $p = 0.0175$ 表示服从正态分布的概率很小; 统计量的值 $l = 0.1062$ 大于接受假设的临界值 $cv = 0.0886$, 因而拒绝假设(测试水平为 5%)。

```
>> hist(Y)
```

从图中看出, 数据 Y 不服从正态分布。

4.8.9 单个样本分布的 Kolmogorov-Smirnov 测试

函数 **kstest**



格式 `H = kstest(X)` %测试向量 `X` 是否服从标准正态分布，测试水平为 5%。
`H = kstest(X,cdf)` %指定累积分布函数为 `cdf` 的测试(`cdf=[]` 时表示标准正态分布)，测试水平为 5%
`H = kstest(X,cdf,alpha)` % `alpha` 为指定测试水平
`[H,P,KSSTAT,CV] = kstest(X,cdf,alpha)` %`P` 为原假设成立的概率，`KSSTAT` 为测试统计量的值，`CV` 为是否接受假设的临界值。

说明 原假设为 `X` 服从标准正态分布。若 `H=0` 则不能拒绝原假设，`H=1` 则可以拒绝原假设。

例 4-82 产生 100 个威布尔随机数，测试该随机数服从的分布

```
>> x=weibrnd(1,2,100,1);
>> [H,p,ksstat,cv]=kstest(x,[x weibcdf(x,1,2)],0.05)      %测试是否服从威布尔分布
H =
    0
p =
    0.3022
ksstat =
    0.0959
cv =
    0.1340
```

说明 `H=0` 表示接受原假设，统计量 `ksstat` 小于临界值表示接受原假设。

```
>> [H,p,ksstat,cv]=kstest(x,[x expcdf(x,1)],0.05)      %测试是否服从指数分布
H =
    1
p =
    0.0073
ksstat =
    0.1653
cv =
    0.1340
```

说明 `H=1` 表明拒绝服从指数分布的假设。

```
>> [H,p,ksstat,cv]=kstest(x,[],0.05)      %测试是否服从标准正态分布
H =
    1
p =
    3.1285e-026
ksstat =
    0.5380
cv =
    0.1340
```

说明 `H=1` 表明不服从标准正态分布。

4.8.10 两个样本具有相同的连续分布的假设检验

函数 `kstest2`

格式 `H = kstest2(X1,X2)` %测试向量 `X1` 与 `X2` 是具有相同的连续分布，测试水平为 5%。

`H = kstest2(X1,X2,alpha)` % `alpha` 为测试水平

`[H,P,KSSTAT] = kstest(X,cdf,alpha)` %与指定累积分布 `cdf` 相同的连续分布，`P` 为假设成立的概率，`KSSTAT` 为测试统计量的值。

说明 原假设为具有相同连续分布。测试结果为 `H`，若 `H=0`，表示应接受原假设；若



H=1, 表示可以拒绝原假设。这是 Kolmogorov-Smirnov 测试方法。

例 4-83

```
>> x=-1:1:5;
>> y=randn(20,1);
>> [h,p,k]=kstest2(x,y)
h =
    1
p =
    0.0444
k =
    0.5643
```

说明 h=1 表示可以认为向量 x 与 y 的分布不相同, 相同的概率只有 4.4%。

4.9 方差分析

4.9.1 单因素方差分析

单因素方差分析是比较两组或多组数据的均值, 它返回原假设——均值相等的概率

函数 **anova1**

格式 **p = anova1(X)** %X 的各列为彼此独立的样本观察值, 其元素个数相同, p 为各列均值相等的概率值, 若 p 值接近于 0, 则原假设受到怀疑, 说明至少有一列均值与其余列均值有明显不同。

p = anova1(X,group) %X 和 group 为向量且 group 要与 X 对应

p = anova1(X,group,'displayopt') % displayopt=on/off 表示显示与隐藏方差分析表图和盒图

[p,table] = anova1(...) % table 为方差分析表

[p,table,stats] = anova1(...) % stats 为分析结果的构造

说明 anova1 函数产生两个图: 标准的方差分析表图和盒图。

方差分析表中有 6 列: 第 1 列(source)显示: X 中数据可变性的来源; 第 2 列(SS)显示: 用于每一列的平方和; 第 3 列(df)显示: 与每一种可变性来源有关的自由度; 第 4 列(MS)显示: 是 SS/df 的比值; 第 5 列(F)显示: F 统计量数值, 它是 MS 的比率; 第 6 列显示: 从 F 累积分布中得到的概率, 当 F 增加时, p 值减少。

例 4-84 设有 3 台机器, 用来生产规格相同的铝合金薄板。取样测量薄板的厚度, 精确至 % 厘米。得结果如下:

机器 1: 0.236 0.238 0.248 0.245 0.243

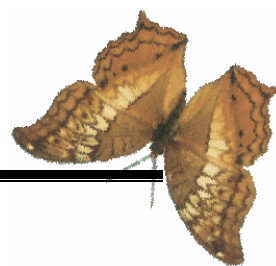
机器 2: 0.257 0.253 0.255 0.254 0.261

机器 3: 0.258 0.264 0.259 0.267 0.262

检验各台机器所生产的薄板的厚度有无显著的差异?

解:

```
>> X=[0.236 0.238 0.248 0.245 0.243; 0.257 0.253 0.255 0.254 0.261;...
      0.258 0.264 0.259 0.267 0.262];
>> P=anova1(X)
```



结果为:

P =
1.3431e-005

还有两个图, 即图 4-22 和图 4-23。

ANOVA Table					
Source	SS	df	MS	F	Prob>F
Columns	0.00105	2	0.00053	32.92	1.34305e-005
Error	0.00019	12	0.00002		
Total	0.00125	14			

图 4-22

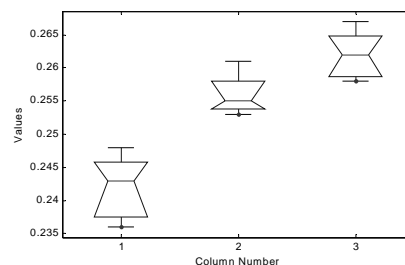


图 4-23

例 4-85 建筑横梁强度的研究: 3000 磅力量作用在一英寸的横梁上来测量横梁的挠度, 钢筋横梁的测试强度是: 82 86 79 83 84 85 86 87; 其余两种更贵的合金横梁强度测试为合金 1: 74 82 78 75 76 77; 合金 2: 79 79 77 78 82 79]。

检验这些合金强度有无明显差异?

解:

```
>> strength = [82 86 79 83 84 85 86 87 74 82 78 75 76 77 79 79 77 78 82 79];
>> alloy = {'st','st','st','st','st','st','st','st', 'al1','al1','al1','al1','al1','al1',...
            'al2','al2','al2','al2','al2','al2'};
>> [p,table,stats] = anova1(strength,alloy,'on')
```

结果为

```
p =
1.5264e-004
table =
    'Source'    'SS'          'df'    'MS'          'F'          'Prob>F'
    'Groups'    [184.8000]    [ 2]    [92.4000]    [15.4000]    [1.5264e-004]
    'Error'     [102.0000]    [17]    [ 6.0000]    []           []
    'Total'     [286.8000]    [19]    []           []           []
stats =
gnames: {3x1 cell}
n: [8 6 6]
source: 'anova1'
means: [84 77 79]
df: 17
s: 2.4495
```

ANOVA Table					
Source	SS	df	MS	F	Prob>F
Groups	184.8	2	92.4	15.4	0.0002
Error	102	17	6		
Total	286.8	19			

图 4-24

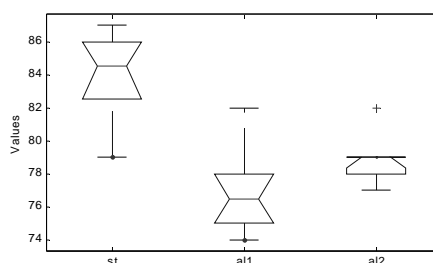


图 4-25



说明 p 值显示, 3 种合金是明显不同的, 盒图显示钢横梁的挠度大于另两种合金横梁的挠度。

4.9.2 双因素方差分析

函数 **anova2**

格式 $p = \text{anova2}(X, \text{reps})$

$p = \text{anova2}(X, \text{reps}, 'displayopt')$

$[p, \text{table}] = \text{anova2}(\dots)$

$[p, \text{table}, \text{stats}] = \text{anova2}(\dots)$

说明 执行平衡的双因素试验的方差分析来比较 X 中两个或多个列(行)的均值, 不同列的数据表示因素 A 的差异, 不同行的数据表示另一因素 B 的差异。如果行列对有多于一个的观察点, 则变量 reps 指出每一单元观察点的数目, 每一单元包含 reps 行, 如:

$$\begin{array}{cc} & \begin{array}{cc} A=1 & A=2 \end{array} \\ \begin{array}{c} X_{111} \quad X_{112} \\ X_{121} \quad X_{122} \\ X_{211} \quad X_{212} \\ X_{221} \quad X_{222} \\ X_{311} \quad X_{312} \\ X_{321} \quad X_{322} \end{array} & \begin{array}{l} \} B=1 \\ \\ \} B=2 \\ \\ \} B=3 \end{array} \end{array}$$

reps=2

其余参数与单因素方差分析参数相似。

例 4-86 一火箭使用了 4 种燃料, 3 种推进器作射程试验, 每种燃料与每种推进器的组合各发射火箭 2 次, 得到结果如下:

推进器 (B)		B1	B2	B3
燃料 A	A1	58.2000	56.2000	65.3000
		52.6000	41.2000	60.8000
	A2	49.1000	54.1000	51.6000
		42.8000	50.5000	48.4000
	A3	60.1000	70.9000	39.2000
		58.3000	73.2000	40.7000
	A4	75.8000	58.2000	48.7000
		71.5000	51.0000	41.4000

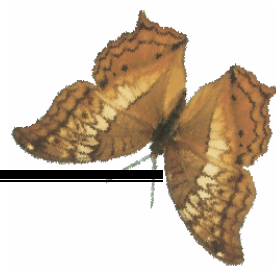
考察推进器和燃料这两个因素对射程是否有显著的影响?

解: 建立 M 文件

```
X=[58.2000    56.2000    65.3000
    52.6000    41.2000    60.8000
    49.1000    54.1000    51.6000
    42.8000    50.5000    48.4000
    60.1000    70.9000    39.2000
    58.3000    73.2000    40.7000
    75.8000    58.2000    48.7000
    71.5000    51.0000    41.4000];
```

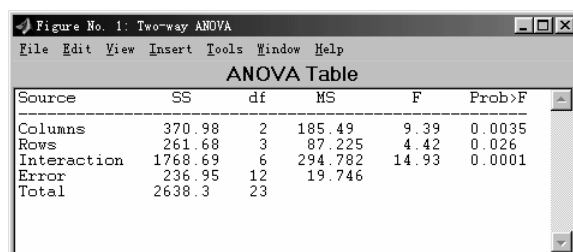
```
P=anova2(X,2)
```

结果为:



P =
0.0035 0.0260 0.0001

显示方差分析图为图 4-26。



Source	SS	df	MS	F	Prob>F
Columns	370.98	2	185.49	9.39	0.0035
Rows	261.68	3	87.225	4.42	0.026
Interaction	1768.69	6	294.782	14.93	0.0001
Error	236.95	12	19.746		
Total	2638.3	23			

图 4-26

第5章 优化问题

5.1 线性规划问题

线性规划问题是目标函数和约束条件均为线性函数的问题，MATLAB6.0 解决的线性规划问题的标准形式为：

$$\begin{aligned} \min \quad & f'x \quad x \in R^n \\ \text{sub.to:} \quad & A \cdot x \leq b \\ & A_{eq} \cdot x = b_{eq} \\ & lb \leq x \leq ub \end{aligned}$$

其中 f 、 x 、 b 、 b_{eq} 、 lb 、 ub 为向量， A 、 A_{eq} 为矩阵。

其它形式的线性规划问题都可经过适当变换化为此标准形式。

在 MATLAB6.0 版中，线性规划问题（Linear Programming）已用函数 `linprog` 取代了 MATLAB5.x 版中的 `lp` 函数。当然，由于版本的向下兼容性，一般说来，低版本中的函数在 6.0 版中仍可使用。

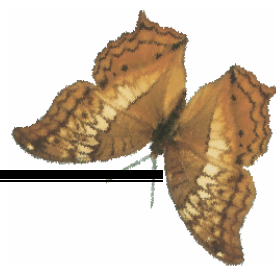
函数 `linprog`

格式 `x = linprog(f,A,b)` %求 $\min f'x$ sub.to $A \cdot x \leq b$ 线性规划的最优解。
`x = linprog(f,A,b,Aeq,beq)` %等式约束 $A_{eq} \cdot x = b_{eq}$ ，若没有不等式约束 $A \cdot x \leq b$ ，则 $A=[]$ ， $b=[]$ 。
`x = linprog(f,A,b,Aeq,beq,lb,ub)` %指定 x 的范围 $lb \leq x \leq ub$ ，若没有等式约束 $A_{eq} \cdot x = b_{eq}$ ，则 $A_{eq}=[]$ ， $b_{eq}=[]$ 。
`x = linprog(f,A,b,Aeq,beq,lb,ub,x0)` %设置初值 x_0
`x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)` % `options` 为指定的优化参数
`[x,fval] = linprog(...)` % 返回目标函数最优值，即 $fval = f'x$ 。
`[x,lambda,exitflag] = linprog(...)` % `lambda` 为解 x 的 Lagrange 乘子。
`[x,lambda,fval,exitflag] = linprog(...)` % `exitflag` 为终止迭代的错误条件。
`[x,fval,lambda,exitflag,output] = linprog(...)` % `output` 为关于优化的一些信息

说明 若 `exitflag>0` 表示函数收敛于解 x ，`exitflag=0` 表示超过函数估值或迭代的最大数字，`exitflag<0` 表示函数不收敛于解 x ；若 `lambda=lower` 表示下界 lb ，`lambda=upper` 表示上界 ub ，`lambda=ineqlin` 表示不等式约束，`lambda=eqlin` 表示等式约束，`lambda` 中的非 0 元素表示对应的约束是有效约束；`output=iterations` 表示迭代次数，`output=algorithm` 表示使用的运算规则，`output=cgiterations` 表示 PCG 迭代次数。

例 5-1 求下面的优化问题

$$\begin{aligned} \min \quad & -5x_1 - 4x_2 - 6x_3 \\ \text{sub.to} \quad & x_1 - x_2 + x_3 \leq 20 \\ & 3x_1 + 2x_2 + 4x_3 \leq 42 \end{aligned}$$



$$3x_1 + 2x_2 \leq 30$$

$$0 \leq x_1, 0 \leq x_2, 0 \leq x_3$$

解:

```
>>f = [-5; -4; -6];
>>A = [1 -1 1;3 2 4;3 2 0];
>>b = [20; 42; 30];
>>lb = zeros(3,1);
>>[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
```

结果为:

```
x =          %最优解
    0.0000
   15.0000
    3.0000
fval =        %最优值
   -78.0000
exitflag =     %收敛
         1
output =
    iterations: 6    %迭代次数
   cgiterations: 0
    algorithm: 'lipsol'    %所使用规则
lambda =
    ineqlin: [3x1 double]
     eqlin: [0x1 double]
     upper: [3x1 double]
     lower: [3x1 double]
>> lambda.ineqlin
ans =
    0.0000
    1.5000
    0.5000
>> lambda.lower
ans =
    1.0000
    0.0000
    0.0000
```

表明: 不等约束条件 2 和 3 以及第 1 个下界是有效的

5.2 foptions 函数

对于优化控制, MATLAB 提供了 18 个参数, 这些参数的具体意义为:

- options(1)-参数显示控制(默认值为 0)。等于 1 时显示一些结果。
- options(2)-优化点 x 的精度控制(默认值为 $1e-4$)。
- options(3)-优化函数 F 的精度控制(默认值为 $1e-4$)。
- options(4)-违反约束的结束标准(默认值为 $1e-6$)。
- options(5)-算法选择, 不常用。

options(6)-优化程序方法选择, 为 0 则为 BFGS 算法, 为 1 则采用 DFP 算法。
 options(7)-线性插值算法选择, 为 0 则为混合插值算法, 为 1 则采用立方插算法。
 options(8)-函数值显示 (目标—达到问题中的 Lambda)
 options(9)-若需要检测用户提供的梯度, 则设为 1。
 options(10)-函数和约束估值的数目。
 options(11)-函数梯度估值的个数。
 options(12)-约束估值的数目。
 options(13)-等约束条件的个数。
 options(14)-函数估值的最大次数 (默认值是 $100 \times$ 变量个数)
 options(15)-用于目标 — 达到问题中的特殊目标。
 options(16)-优化过程中变量的最小有限差分梯度值。
 options(17)- 优化过程中变量的最大有限差分梯度值。
 options(18)-步长设置 (默认为 1 或更小)。

Foptions 已经被 optimset 和 optimget 代替, 详情请查函数 optimset 和 optimget。

5.3 非线性规划问题

5.3.1 有约束的一元函数的最小值

单变量函数求最小值的标准形式为 $\min_x f(x) \quad \text{sub.to} \quad x_1 < x < x_2$

在 MATLAB5.x 中使用 fmin 函数求其最小值。

函数 fminbnd

格式 $x = \text{fminbnd}(\text{fun}, x_1, x_2)$ % 返回自变量 x 在区间 $x_1 < x < x_2$ 上函数 fun 取最小值时 x 值, fun 为目标函数的表达式字符串或 MATLAB 自定义函数的函数柄。

$x = \text{fminbnd}(\text{fun}, x_1, x_2, \text{options})$ % options 为指定优化参数选项

$[x, \text{fval}] = \text{fminbnd}(\cdots)$ % fval 为目标函数的最小值

$[x, \text{fval}, \text{exitflag}] = \text{fminbnd}(\cdots)$ % exitflag 为终止迭代的条件

$[x, \text{fval}, \text{exitflag}, \text{output}] = \text{fminbnd}(\cdots)$ % output 为优化信息

说明 若参数 $\text{exitflag} > 0$, 表示函数收敛于 x , 若 $\text{exitflag} = 0$, 表示超过函数估计值或迭代的最大数字, $\text{exitflag} < 0$ 表示函数不收敛于 x ; 若参数 $\text{output} = \text{iterations}$ 表示迭代次数, $\text{output} = \text{funccount}$ 表示函数赋值次数, $\text{output} = \text{algorithm}$ 表示所使用的算法。

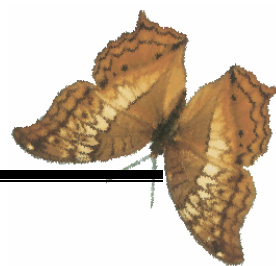
例 5-2 计算下面函数在区间(0, 1)内的最小值。

$$f(x) = \frac{x^3 + \cos x + x \log x}{e^x}$$

解: `>> [x,fval,exitflag,output]=fminbnd('(x^3+cos(x)+x*log(x))/exp(x)',0,1)`

`x =`
0.5223





```
fval =
    0.3974
exitflag =
    1
output =
    iterations: 9
    funcCount: 9
    algorithm: 'golden section search, parabolic interpolation'
```

例 5-3 在 $[0, 5]$ 上求下面函数的最小值 $f(x) = (x - 3)^3 - 1$

解：先自定义函数：在 MATLAB 编辑器中建立 M 文件为：

```
function f = myfun(x)
f = (x-3).^2 - 1;
```

保存为 myfun.m，然后在命令窗口键入命令：

```
>> x=fminbnd(@myfun,0,5)
```

则结果显示为：

```
x =
    3
```

5.3.2 无约束多元函数最小值

多元函数最小值的标准形式为 $\min_x f(x)$

其中：x 为向量，如 $x = [x_1, x_2, \dots, x_n]$

在 MATLAB5.x 中使用 fmins 求其最小值。

命令 利用函数 fminsearch 求无约束多元函数最小值

函数 **fminsearch**

格式 $x = \text{fminsearch}(\text{fun}, x_0)$ % x_0 为初始点，fun 为目标函数的表达式字符串或 MATLAB 自定义函数的函数柄。

$x = \text{fminsearch}(\text{fun}, x_0, \text{options})$ % options 查 optimset

$[x, \text{fval}] = \text{fminsearch}(\dots)$ % 最优点的函数值

$[x, \text{fval}, \text{exitflag}] = \text{fminsearch}(\dots)$ % exitflag 与单变量情形一致

$[x, \text{fval}, \text{exitflag}, \text{output}] = \text{fminsearch}(\dots)$ % output 与单变量情形一致

注意：fminsearch 采用了 Nelder-Mead 型简单搜寻法。

例 5-4 求 $y = 2x_1^3 + 4x_1x_2^3 - 10x_1x_2 + x_2^2$ 的最小值点

解：>> X=fminsearch('2*x(1)^3+4*x(1)*x(2)^3-10*x(1)*x(2)+x(2)^2', [0,0])

结果为

```
X =
    1.0016    0.8335
```

或在 MATLAB 编辑器中建立函数文件

```
function f=myfun(x)
f=2*x(1)^3+4*x(1)*x(2)^3-10*x(1)*x(2)+x(2)^2;
```

保存为 myfun.m，在命令窗口键入

```
>> X=fminsearch('myfun', [0,0]) 或 >> X=fminsearch(@myfun, [0,0])
```

结果为:

```
X =
    1.0016    0.8335
```

命令 利用函数 `fminunc` 求多变量无约束函数最小值

函数 `fminunc`

格式 `x = fminunc(fun,x0)` % 返回给定初始点 `x0` 的最小函数值点
`x = fminunc(fun,x0,options)` % `options` 为指定优化参数
`[x,fval] = fminunc(...)` % `fval` 最优点 `x` 处的函数值
`[x,fval,exitflag] = fminunc(...)` % `exitflag` 为终止迭代的条件, 与上同。
`[x,fval,exitflag,output] = fminunc(...)` % `output` 为输出优化信息
`[x,fval,exitflag,output,grad] = fminunc(...)` % `grad` 为函数在解 `x` 处的梯度值
`[x,fval,exitflag,output,grad,hessian] = fminunc(...)` % 目标函数在解 `x` 处的海赛 (Hessian) 值

注意: 当函数的阶数大于 2 时, 使用 `fminunc` 比 `fminsearch` 更有效, 但当所选函数高度不连续时, 使用 `fminsearch` 效果较好。

例 5-5 求 $f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$ 的最小值。

```
>> fun='3*x(1)^2+2*x(1)*x(2)+x(2)^2';
>> x0=[1 1];
>> [x,fval,exitflag,output,grad,hessian]=fminunc(fun,x0)
```

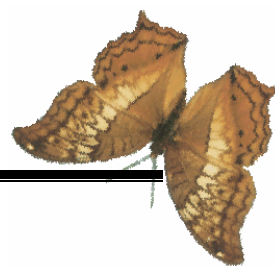
结果为:

```
x =
    1.0e-008 *
   -0.7591    0.2665
fval =
    1.3953e-016
exitflag =
     1
output =
    iterations: 3
    funcCount: 16
    stepsize: 1.2353
    firstorderopt: 1.6772e-007
    algorithm: 'medium-scale: Quasi-Newton line search'
grad =
    1.0e-006 *
   -0.1677
    0.0114
hessian =
    6.0000    2.0000
    2.0000    2.0000
```

或用下面方法:

```
>> fun=inline('3*x(1)^2+2*x(1)*x(2)+x(2)^2')
fun =
    Inline function:
    fun(x) = 3*x(1)^2+2*x(1)*x(2)+x(2)^2
>> x0=[1 1];
>> x=fminunc(fun,x0)
```





```
x =
    1.0e-008 *
   -0.7591    0.2665
```

5.3.3 有约束的多元函数最小值

非线性有约束的多元函数的标准形式为：

$$\begin{aligned} \min_x \quad & f(x) \\ \text{sub.to} \quad & C(x) \leq 0 \\ & C_{eq}(x) = 0 \\ & A \cdot x \leq b \\ & A_{eq} \cdot x = b_{eq} \\ & lb \leq x \leq ub \end{aligned}$$

其中： x 、 b 、 b_{eq} 、 lb 、 ub 是向量， A 、 A_{eq} 为矩阵， $C(x)$ 、 $C_{eq}(x)$ 是返回向量的函数， $f(x)$ 为目标函数， $f(x)$ 、 $C(x)$ 、 $C_{eq}(x)$ 可以是非线性函数。

在 MATLAB5.x 中，它的求解由函数 `constr` 实现。

函数 `fmincon`

格式

```
x = fmincon(fun,x0,A,b)
x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval] = fmincon(...)
[x,fval,exitflag] = fmincon(...)
[x,fval,exitflag,output] = fmincon(...)
[x,fval,exitflag,output,lambda] = fmincon(...)
[x,fval,exitflag,output,lambda,grad] = fmincon(...)
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)
```

参数说明：`fun` 为目标函数，它可用前面的方法定义；

`x0` 为初始值；

`A`、`b` 满足线性不等式约束 $A \cdot x \leq b$ ，若没有不等式约束，则取 `A=[]`，`b=[]`；

`Aeq`、`beq` 满足等式约束 $A_{eq} \cdot x = b_{eq}$ ，若没有，则取 `Aeq=[]`，`beq=[]`；

`lb`、`ub` 满足 $lb \leq x \leq ub$ ，若没有界，可设 `lb=[]`，`ub=[]`；

`nonlcon` 的作用是通过接受的向量 `x` 来计算非线性不等约束 $C(x) \leq 0$ 和等式约束 $C_{eq}(x) = 0$ 分别在 `x` 处的估计 `C` 和 `Ceq`，通过指定函数柄来使用，

如：`>>x = fmincon(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)`，先建立非线性约束函数，并保存为 `mycon.m`：`function [C,Ceq] = mycon(x)`

`C = ...` % 计算 `x` 处的非线性不等约束 $C(x) \leq 0$ 的函数值。

`Ceq = ...` % 计算 `x` 处的非线性等式约束 $C_{eq}(x) = 0$ 的函数值。

`lambda` 是 Lagrange 乘子，它体现哪一个约束有效。

output 输出优化信息;

grad 表示目标函数在 x 处的梯度;

hessian 表示目标函数在 x 处的 Hessian 值。

例 5-6 求下面问题在初始点 $(0, 1)$ 处的最优解

$$\min \quad x_1^2 + x_2^2 - x_1 x_2 - 2x_1 - 5x_2$$

$$\text{sub.to} \quad -(x_1 - 1)^2 + x_2 \geq 0$$

$$2x_1 - 3x_2 + 6 \geq 0$$

解: 约束条件的标准形式为

$$\text{sub.to} \quad (x_1 - 1)^2 - x_2 \leq 0$$

$$-2x_1 + 3x_2 \leq 6$$

先在 MATLAB 编辑器中建立非线性约束函数文件:

```
function [c, ceq]=mycon(x)
```

```
c=(x(1)-1)^2-x(2);
```

```
ceq=[]; %无等式约束
```

然后, 在命令窗口键入如下命令或建立 M 文件:

```
>>fun='x(1)^2+x(2)^2-x(1)*x(2)-2*x(1)-5*x(2)'; %目标函数
```

```
>>x0=[0 1];
```

```
>>A=[-2 3]; %线性不等式约束
```

```
>>b=6;
```

```
>>Aeq=[]; %无线性等式约束
```

```
>>beq=[];
```

```
>>lb=[]; %x 没有下、上界
```

```
>>ub=[];
```

```
>>[x,fval,exitflag,output,lambda,grad,hessian]
```

```
=fmincon(fun,x0,A,b,Aeq,beq,lb,ub,@mycon)
```

则结果为

```
x =
```

```
3 4
```

```
fval =
```

```
-13
```

```
exitflag = %解收敛
```

```
1
```

```
output =
```

```
iterations: 2
```

```
funcCount: 9
```

```
stepsize: 1
```

```
algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
```

```
firstorderopt: []
```

```
cgiterations: []
```

```
lambda =
```

```
lower: [2x1 double] %x 下界有效情况, 通过 lambda.lower 可查看。
```

```
upper: [2x1 double] %x 上界有效情况, 为 0 表示约束无效。
```

```
eqlin: [0x1 double] %线性等式约束有效情况, 不为 0 表示约束有效。
```

```
eqnonlin: [0x1 double] %非线性等式约束有效情况。
```

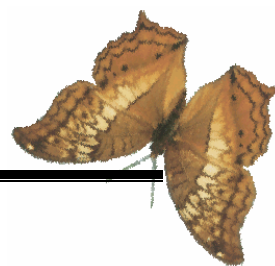
```
ineqlin: 2.5081e-008 %线性不等式约束有效情况。
```

```
ineqnonlin: 6.1938e-008 %非线性不等式约束有效情况。
```

```
grad = %目标函数在最小值点的梯度
```

```
1.0e-006 *
```

```
-0.1776
```

```

0
hessian = %目标函数在最小值点的 Hessian 值
1.0000 -0.0000
-0.0000 1.0000

```

例 5-7 求下面问题在初始点 $x=(10, 10, 10)$ 处的最优解。

```

Min    f(x) = -x1x2x3
Sub.to  0 ≤ x1 + 2x2 + 2x3 ≤ 72

```

解：约束条件的标准形式为

```

sub.to    -x1 - 2x2 - 2x3 ≤ 0
           x1 + 2x2 + 2x3 ≤ 72

```

```

>> fun = '-x(1)*x(2)*x(3)';
>> x0 = [10,10,10];
>> A = [-1 -2 -2; 1 2 2];
>> b = [0;72];
>> [x,fval] = fmincon(fun,x0,A,b)

```

结果为：

```

x =
24.0000 12.0000 12.0000
fval =
-3456

```

5.3.4 二次规划问题

二次规划问题（quadratic programming）的标准形式为：

$$\min \quad \frac{1}{2} x' H x + f' x$$

```

sub.to  A · x ≤ b
        Aeq · x = beq
        lb ≤ x ≤ ub

```

其中， H 、 A 、 Aeq 为矩阵， f 、 b 、 beq 、 lb 、 ub 、 x 为向量

其它形式的二次规划问题都可转化为标准形式。

MATLAB5.x 版中的 `qp` 函数已被 6.0 版中的函数 `quadprog` 取代。

函数 quadprog

格式 `x = quadprog(H,f,A,b)` % 其中 H, f, A, b 为标准形中的参数， x 为目标函数的最小值。

`x = quadprog(H,f,A,b,Aeq,beq)` % Aeq, beq 满足等约束条件 $Aeq \cdot x = beq$ 。

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub)` % lb, ub 分别为解 x 的下界与上界。

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)` % $x0$ 为设置的初值

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)` % $options$ 为指定的优化参数

`[x,fval] = quadprog(...)` % $fval$ 为目标函数最优值

`[x,fval,exitflag] = quadprog(...)` % $exitflag$ 与线性规划中参数意义相同

`[x,fval,exitflag,output] = quadprog(...)` % $output$ 与线性规划中参数意义相同

`[x,fval,exitflag,output,lambda] = quadprog(...)` % $lambda$ 与线性规划中参数意义相同

例 5-8 求解下面二次规划问题

$$\min \quad f(x) = \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 - 2x_1 - 6x_2$$

sub.to

$$x_1 + x_2 \leq 2$$

$$-x_1 + 2x_2 \leq 2$$

$$2x_1 + x_2 \leq 3$$

$$0 \leq x_1, \quad 0 \leq x_2$$

解: $f(x) = \frac{1}{2}x'Hx + f'x$

则 $H = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$, $f = \begin{bmatrix} -2 \\ -6 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

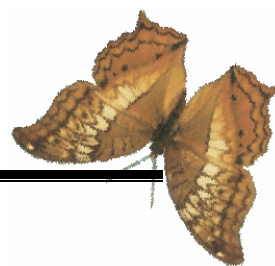
在 MATLAB 中实现如下:

```
>>H = [1 -1; -1 2];
>>f = [-2; -6];
>>A = [1 1; -1 2; 2 1];
>>b = [2; 2; 3];
>>lb = zeros(2,1);
>>[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[],[],lb)
```

结果为:

```
x =          %最优解
    0.6667
    1.3333
fval =          %最优值
   -8.2222
exitflag =          %收敛
         1
output =
    iterations: 3
    algorithm: 'medium-scale: active-set'
    firstorderopt: []
    cgiterations: []
lambda =
    lower: [2x1 double]
    upper: [2x1 double]
    eqlin: [0x1 double]
    ineqlin: [3x1 double]
>> lambda.ineqlin
ans =
    3.1111
    0.4444
         0
>> lambda.lower
ans =
         0
         0
```

说明 第 1、2 个约束条件有效, 其余无效。



例 5-9 求二次规划的最优解

$$\begin{aligned} \max \quad & f(x_1, x_2) = x_1 x_2 + 3 \\ \text{sub.to} \quad & x_1 + x_2 - 2 = 0 \end{aligned}$$

解：化成标准形式：

$$\min \quad f(x_1, x_2) = -x_1 x_2 - 3 = \frac{1}{2} (x_1 \ x_2) \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (0, 0) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 3$$

$$\text{sub.to} \quad x_1 + x_2 = 2$$

在 Matlab 中实现如下：

```
>>H=[0,-1;-1,0];
>>f=[0;0];
>>Aeq=[1 1];
>>b=2;
>>[x,fval,exitflag,output,lambda] = quadprog(H,f,[],[],Aeq,b)
```

结果为：

```
x =
    1.0000
    1.0000
fval =
   -1.0000
exitflag =
     1
output =
    firstorderopt: 0
      iterations: 1
    cgiterations: 1
      algorithm: [1x58 char]
lambda =
      eqlin: 1.0000
    ineqlin: []
      lower: []
      upper: []
```

5.4 “半无限”有约束的多元函数最优解

“半无限”有约束多元函数最优解问题的标准形式为

$$\begin{aligned} \min_x \quad & f(x) \\ \text{sub.to} \quad & C(x) \leq 0 \\ & C_{\text{eq}}(x) = 0 \\ & A \cdot x \leq b \\ & A_{\text{eq}} \cdot x = b_{\text{eq}} \\ & K_1(x, w_1) \leq 0 \\ & K_2(x, w_2) \leq 0 \\ & \dots \\ & K_n(x, w_n) \leq 0 \end{aligned}$$

其中： x 、 b 、 beq 、 lb 、 ub 都是向量； A 、 Aeq 是矩阵； $C(x)$ 、 $Ceq(x)$ 、 $K_i(x, w_i)$ 是返回向量的函数， $f(x)$ 为目标函数； $f(x)$ 、 $C(x)$ 、 $Ceq(x)$ 是非线性函数； $K_i(x, w_i)$ 为半无限约束， w_1, w_2, \dots, w_n 通常是长度为 2 的向量。

在 MTALAB5.x 中，使用函数 `seminf` 解决这类问题。

函数 `fseminf`

格式 `x = fseminf(fun,x0,ntheta,seminfcon)`
`x = fseminf(fun,x0,ntheta,seminfcon,A,b)`
`x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq)`
`x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub)`
`x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options)`
`[x,fval] = fseminf(...)`
`[x,fval,exitflag] = fseminf(...)`
`[x,fval,exitflag,output] = fseminf(...)`
`[x,fval,exitflag,output,lambda] = fseminf(...)`

参数说明： x_0 为初始估计值；

fun 为目标函数，其定义方式与前面相同；

A 、 b 由线性不等式约束 $A \cdot x \leq b$ 确定，没有，则 $A=[]$ ， $b=[]$ ；

Aeq 、 beq 由线性等式约束 $Aeq \cdot x = beq$ 确定，没有，则 $Aeq=[]$ ， $beq=[]$ ；

Lb 、 ub 由变量 x 的范围 $lb \leq x \leq ub$ 确定；

$options$ 为优化参数；

$ntheta$ 为半无限约束的个数；

`seminfcon` 用来确定非线性约束向量 C 和 Ceq 以及半无限约束的向量 K_1 ， K_2 ， \dots ， K_n ，通过指定函数柄来使用，如：

```
x = fseminf(@myfun,x0,ntheta,@myinfcon)
```

先建立非线性约束和半无限约束函数文件，并保存为 `myinfcon.m`：

```
function [C,Ceq,K1,K2,...,Ktheta,S] = myinfcon(x,S)
```

```
%S 为向量 w 的采样值
```

```
% 初始化样本间距
```

```
if isnan(S(1,1)),
```

```
    S = ... % S 有 ntheta 行 2 列
```

```
end
```

```
w1 = ... % 计算样本集
```

```
w2 = ... % 计算样本集
```

```
...
```

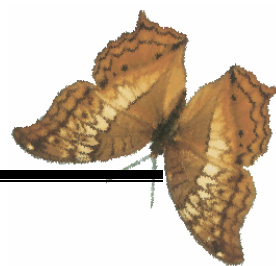
```
wtheta = ... % 计算样本集
```

```
K1 = ... % 在 x 和 w 处的第 1 个半无限约束值
```

```
K2 = ... % 在 x 和 w 处的第 2 个半无限约束值
```

```
...
```

```
Ktheta = ... % 在 x 和 w 处的第 ntheta 个半无限约束值
```



$C = \dots$ % 在 x 处计算非线性不等式约束值
 $Ceq = \dots$ % 在 x 处计算非线性等式约束值
 如果没有约束, 则相应的值取为 “[]”, 如 $Ceq=[]$
 $fval$ 为在 x 处的目标函数最小值;
 $exitflag$ 为终止迭代的条件;
 $output$ 为输出的优化信息;
 $lambda$ 为解 x 的 Lagrange 乘子。

例 5-10 求下面一维情形的最优化问题

$$\min_x f(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2$$

sub.to

$$K_1(x, w_1) = \sin(w_1 x_1) \cos(w_1 x_2) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 \leq 1$$

$$K_2(x, w_2) = \sin(w_2 x_2) \cos(w_2 x_1) - \frac{1}{1000}(w_2 - 50)^2 - \sin(w_2 x_3) - x_3 \leq 1$$

$$1 \leq w_1 \leq 100$$

$$1 \leq w_2 \leq 100$$

解: 将约束方程化为标准形式:

$$K_1(x, w_1) = \sin(w_1 x_1) \cos(w_1 x_2) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 - 1 \leq 0$$

$$K_2(x, w_2) = \sin(w_2 x_2) \cos(w_2 x_1) - \frac{1}{1000}(w_2 - 50)^2 - \sin(w_2 x_3) - x_3 - 1 \leq 0$$

先建立非线性约束和半无限约束函数文件, 并保存为 mycon.m:

```

function [C,Ceq,K1,K2,S] = mycon(X,S)
% 初始化样本间距:
if isnan(S(1,1)),
    S = [0.2 0; 0.2 0];
end
% 产生样本集:
w1 = 1:S(1,1):100;
w2 = 1:S(2,1):100;
% 计算半无限约束:
K1 = sin(w1*X(1)).*cos(w1*X(2)) - 1/1000*(w1-50).^2 - sin(w1*X(3))-X(3)-1;
K2 = sin(w2*X(2)).*cos(w2*X(1)) - 1/1000*(w2-50).^2 - sin(w2*X(3))-X(3)-1;
% 无非线性约束:
C = []; Ceq=[];
% 绘制半无限约束图形
plot(w1,K1,'-',w2,K2,'-'),title('Semi-infinite constraints')
  
```

然后在 MATLAB 命令窗口或编辑器中建立 M 文件:

```

fun = 'sum((x-0.5).^2)';
x0 = [0.5; 0.2; 0.3];      % Starting guess
[x,fval] = fseminf(fun,x0,2,@mycon)
  
```

结果为:

```

x =
    0.6673
  
```

```

0.3013
0.4023
fval =
0.0770
>>[C,Ceq,K1,K2] = mycon (x,NaN);    % 利用初始样本间距
>>max(K1)
ans =
-0.0017
>>max(K2)
ans =
-0.0845

```

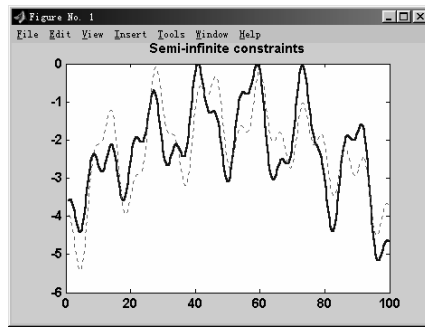


图 5-1

例 5-11 求下面二维情形的最优化问题

$$\min_x f(x) = (x_1 - 0.2)^2 + (x_2 - 0.2)^2 + (x_3 - 0.2)^2$$

sub.to

$$K_1(x, w) = \sin(w_1 x_1) \cos(w_2 x_2) - \frac{1}{1000} (w_1 - 50)^2 - \sin(w_1 x_3) - x_3 + \dots$$

$$\sin(w_2 x_2) \cos(w_1 x_1) - \frac{1}{1000} (w_2 - 50)^2 - \sin(w_2 x_3) - x_3 \leq 1.5$$

$$1 \leq w_1 \leq 100$$

$$1 \leq w_2 \leq 100$$

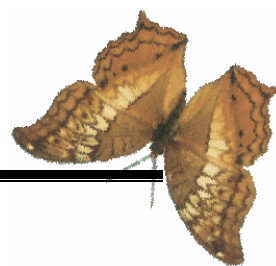
初始点为 $x_0 = [0.25, 0.25, 0.25]$ 。

解：先建立非线性和半无限约束函数文件，并保存为 mycon.m:

```

function [C,Ceq,K1,S] = mycon(X,S)
% 初始化样本间距:
if isnan(s(1,1)),
    s = [2 2];
end
% 设置样本集
w1x = 1:s(1,1):100;
w1y = 1:s(1,2):100;
[wx, wy] = meshgrid(w1x,w1y);
% 计算半无限约束函数值
K1 = sin(wx*X(1)).*cos(wx*X(2))-1/1000*(wx-50).^2 -sin(wx*X(3))-X(3)+...
sin(wy*X(2)).*cos(wx*X(1))-1/1000*(wy-50).^2-sin(wy*X(3))-X(3)-1.5;
% 无非线性约束
C = []; Ceq=[];

```



```
%作约束曲面图形
m = surf(wx,wy,K1,'edgecolor','none','facecolor','interp');
camlight headlight
title('Semi-infinite constraint')
drawnow
```

然后在 MATLAB 命令窗口下键入命令：

```
>>fun = 'sum((x-0.2).^2)';
>>x0 = [0.25, 0.25, 0.25];
>>[x,fval] = fseminf(fun,x0,1,@mycon)
```

结果为（如图）

```
x =
    0.2926    0.1874    0.2202
fval =
    0.0091
>>[c,ceq,K1] = mycon(x,[0.5,0.5]); % 样本间距为 0.5
>>max(max(K1))
ans =
   -0.0027
```

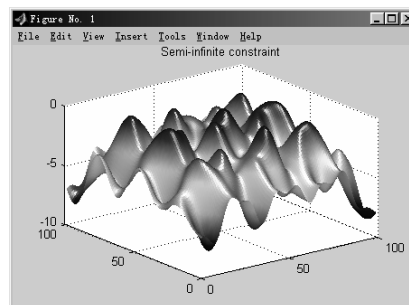


图 5-2

5.5 极小化极大（Minmax）问题

极小化极大问题的标准形式为

$$\begin{aligned} & \min_x \max_{\{F_i(x)\}} \{F_i(x)\} \\ \text{sub.to} \quad & C(x) \leq 0 \\ & C_{eq}(x) = 0 \\ & A \cdot x \leq b \\ & A_{eq} \cdot x = b_{eq} \\ & lb \leq x \leq ub \end{aligned}$$

其中：x、b、beq、lb、ub 是向量，A、Aeq 为矩阵，C(x)、Ceq(x)和 F(x)是返回向量的函数，F(x)、C(x)、Ceq(x)可以是非线性函数。

在 MATLAB5.x 中，它的求解由函数 minmax 实现。

函数 **fminimax**

格式 **x = fminimax(fun,x0)**

x = fminimax(fun,x0,A,b)

x = fminimax(fun,x0,A,b,Aeq,beq)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)

[x,fval,maxfval] = fminimax(...)

[x,fval,maxfval,exitflag] = fminimax(...)

[x,fval,maxfval,exitflag,output] = fminimax(...)

[x,fval,maxfval,exitflag,output,lambdas] = fminimax(...)

参数说明: fun 为目标函数;

x0 为初始值;

A、b 满足线性不等约束 $A \cdot x \leq b$, 若没有不等约束, 则取 $A=[]$, $b=[]$;

Aeq、beq 满足等式约束 $Aeq \cdot x = beq$, 若没有, 则取 $Aeq=[]$, $beq=[]$;

lb、ub 满足 $lb \leq x \leq ub$, 若没有界, 可设 $lb=[]$, $ub=[]$;

nonlcon 的作用是通过接受的向量 x 来计算非线性不等约束 $C(x) \leq 0$ 和等式约束

$Ceq(x) = 0$ 分别在 x 处的值 C 和 Ceq, 通过指定函数柄来使用, 如: `>>x =`

`fminimax(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)`, 先建立非线性约束函数,

并保存为 mycon.m: `function [C,Ceq] = mycon(x)`

`C = ...` % 计算 x 处的非线性不等约束 $C(x) \leq 0$ 的函数值。

`Ceq = ...` % 计算 x 处的非线性等式约束 $Ceq(x) = 0$ 的函数值。

options 为指定的优化参数;

fval 为最优点处的目标函数值;

maxfval 为目标函数在 x 处的最大值;

exitflag 为终止迭代的条件;

lambda 是 Lagrange 乘子, 它体现哪一个约束有效。

output 输出优化信息。

例 5-12 求下列函数最大值的最小化问题

$$[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)]$$

其中: $f_1(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2 + 304$

$$f_2(x) = -x_2^2 - 3x_2^2$$

$$f_3(x) = x_1 + 3x_2 - 18$$

$$f_4(x) = -x_1 - x_2$$

$$f_5(x) = x_1 + x_2 - 8$$

解: 先建立目标函数文件, 并保存为 myfun.m: `function f = myfun(x)`

$$f(1) = 2*x(1)^2 + x(2)^2 - 48*x(1) - 40*x(2) + 304;$$

$$f(2) = -x(1)^2 - 3*x(2)^2;$$

$$f(3) = x(1) + 3*x(2) - 18;$$

$$f(4) = -x(1) - x(2);$$

$$f(5) = x(1) + x(2) - 8;$$

然后, 在命令窗口键入命令:

$$x0 = [0.1; 0.1]; \quad \% \text{ 初始值}$$

$$[x, fval] = \text{fminimax}(@\text{myfun}, x0)$$

结果为:

$$x =$$

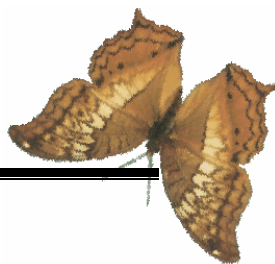
$$4.0000$$

$$4.0000$$

$$fval =$$

$$0.0000 \quad -64.0000 \quad -2.0000 \quad -8.0000 \quad -0.0000$$

例 5-13 求上述问题的绝对值的最大值最小化问题。



目标函数为: $[|f_1(x)|, |f_2(x)|, |f_3(x)|, |f_4(x)|, |f_5(x)|]$

解: 先建立目标函数文件 (与上例相同)

然后, 在命令窗口或编辑器中建立 M 文件:

```
>>x0 = [0.1; 0.1];           % 初始点
>>options = optimset('MinAbsMax',5); % 指定绝对值的最小化
>>[x,fval] = fminimax(@myfun,x0,[],[],[],[],[],[],options)
```

则结果为

```
x =
    4.9256
    2.0796
fval =
    37.2356   -37.2356   -6.8357   -7.0052   -0.9948
```

5.6 多目标规划问题

多目标规划是指在一组约束下, 对多个不同目标函数进行优化。它的一般形式为

$$\begin{aligned} \min \quad & [f_1(x), f_2(x), \dots, f_m(x)] \\ \text{sub.to} \quad & g_j(x) \leq 0 \quad j=1, 2, \dots, p \end{aligned}$$

其中: $x = (x_1, x_2, \dots, x_n)$ 。

在同一约束下, 当目标函数处于冲突状态时, 不存在最优解 x 使所有目标函数同时达到最优。此时, 我们使用有效解, 即如果不存在 $x \in S$, 使得 $f_i(x) \geq f_i(x^*)$, $i=1, 2, \dots, m$, 则称 x^* 为有效解。

在 MATLAB 中, 多目标问题的标准形式为

$$\begin{aligned} \underset{x, \gamma}{\text{minimize}} \quad & \gamma \\ \text{sub.to} \quad & F(x) - \text{weight} \cdot \gamma \leq \text{goal} \\ & C(x) \leq 0 \\ & \text{Ceq}(x) = 0 \\ & A \cdot x \leq b \\ & \text{Aeq} \cdot x = \text{beq} \\ & \text{lb} \leq x \leq \text{ub} \end{aligned}$$

其中: x 、 b 、 beq 、 lb 、 ub 是向量; A 、 Aeq 为矩阵; $C(x)$ 、 $\text{Ceq}(x)$ 和 $F(x)$ 是返回向量的函数; $F(x)$ 、 $C(x)$ 、 $\text{Ceq}(x)$ 可以是非线性函数; weight 为权值系数向量, 用于控制对应的目标函数与用户定义的目标函数值的接近程度; goal 为用户设计的与目标函数相应的目标函数值向量; γ 为一个松弛因子标量; $F(x)$ 为多目标规划中的目标函数向量。

在 MATLAB5.x 中, 它的最优解由 `attgoal` 函数实现。

函数 fgoalattain

格式 `x = fgoalattain(fun,x0,goal,weight)`

`x = fgoalattain(fun,x0,goal,weight,A,b)`

`x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)`

```

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval] = fgoalattain(...)
[x,fval,attainfactor] = fgoalattain(...)
[x,fval,attainfactor,exitflag] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)

```

参数说明:

x_0 为初始解向量;

fun 为多目标函数的文件名字符串, 其定义方式与前面 fun 的定义方式相同;

$goal$ 为用户设计的目标函数值向量;

$weight$ 为权值系数向量, 用于控制目标函数与用户自定义目标值的接近程度;

A 、 b 满足线性不等式约束 $A \cdot x \leq b$, 没有时取 $A=[]$, $b=[]$;

Aeq 、 beq 满足线性等式约束 $Aeq \cdot x = beq$, 没有时取 $Aeq=[]$, $beq=[]$;

lb 、 ub 为变量的下界和上界: $lb \leq x \leq ub$;

$nonlcon$ 的作用是通过接受的向量 x 来计算非线性不等约束 $C(x) \leq 0$ 和等式约束 $Ceq(x) = 0$ 分别在 x 处的值 C 和 Ceq , 通过指定函数柄来使用。

如: `>>x = fgoalattain(@myfun,x0,goal,weight,A,b,Aeq,beq,lb,ub,@mycon)`,

先建立非线性约束函数, 并保存为 `mycon.m`: `function [C,Ceq] = mycon(x)`

`C = ...` % 计算 x 处的非线性不等式约束 $C(x) \leq 0$ 的函数值。

`Ceq = ...` % 计算 x 处的非线性等式约束 $Ceq(x) = 0$ 的函数值。

$options$ 为指定的优化参数;

$fval$ 为多目标函数在 x 处的值;

$attainfactor$ 为解 x 处的目标规划因子;

$exitflag$ 为终止迭代的条件;

$output$ 为输出的优化信息;

$lambda$ 为解 x 处的 Lagrange 乘子

例 5-14 控制系统输出反馈器设计。

设如下线性系统

$$\dot{x} = Ax + Bu$$

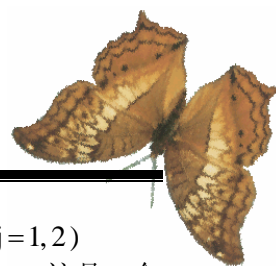
$$y = Cx$$

$$\text{其中: } A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

要求设计输出反馈控制器 K , 使闭环系统

$$\dot{x} = (A + BKC)x + Bu$$

$$y = Cx$$



在复平面实轴上点 $[-5, -3, -1]$ 的左侧有极点，并要求 $-4 \leq K_{ij} \leq 4$ ($i, j=1, 2$)

解：上述问题就是要求解矩阵 K ，使矩阵 $(A+BKC)$ 的极点为 $[-5, -3, -1]$ ，这是一个多目标规划问题。

先建立目标函数文件，保存为 eigfun.m:

```
function F = eigfun(K,A,B,C)
```

```
F = sort(eig(A+B*K*C)); % 估计目标函数值
```

然后，输入参数并调用优化程序：

```
A = [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0; -2 2; 0 1];
C = [1 0 0; 0 0 1];
K0 = [-1 -1; -1 -1]; % 初始化控制器矩阵
goal = [-5 -3 -1]; % 为闭环路的特征值（极点）设置目标值向量
weight = abs(goal) % 设置权值向量
lb = -4*ones(size(K0)); % 设置控制器的下界
ub = 4*ones(size(K0)); % 设置控制器的上界
options = optimset('Display','iter'); % 设置显示参数：显示每次迭代的输出
[K,fval,attainfactor] = fgoalattain(@eigfun,K0,goal,weight,[],[],lb,ub,[],options,A,B,C)
```

结果为：

```
weight =
```

```
5 3 1
```

Iter	F-count	Attainment factor	Step-size	Directional derivative	Procedure
1	6	1.885	1	1.03	
2	13	1.061	1	-0.679	
3	20	0.4211	1	-0.523	Hessian modified
4	27	-0.06352	1	-0.053	Hessian modified twice
5	34	-0.1571	1	-0.133	
6	41	-0.3489	1	-0.00768	Hessian modified
7	48	-0.3643	1	-4.25e-005	Hessian modified
8	55	-0.3645	1	-0.00303	Hessian modified twice
9	62	-0.3674	1	-0.0213	Hessian modified
10	69	-0.3806	1	0.00266	
11	76	-0.3862	1	-2.73e-005	Hessian modified twice
12	83	-0.3863	1	-1.22e-013	Hessian modified twice

Optimization terminated successfully:

Search direction less than 2*options.TolX and maximum constraint violation is less than options.TolCon

Active Constraints:

```
1
```

```
2
```

```
4
```

```
9
```

```
10
```

K =

```
-4.0000 -0.2564
```

```
-4.0000 -4.0000
```

fval =

```
-6.9313
```

```
-4.1588
```

```
-1.4099
```

attainfactor =

```
-0.3863
```

5.7 最小二乘最优问题

5.7.1 约束线性最小二乘

有约束线性最小二乘的标准形式为

$$\begin{aligned} \min_x \quad & \frac{1}{2} \|Cx - d\|_2^2 \\ \text{sub.to} \quad & A \cdot x \leq b \\ & A_{eq} \cdot x = b_{eq} \\ & lb \leq x \leq ub \end{aligned}$$

其中：C、A、Aeq 为矩阵；d、b、beq、lb、ub、x 是向量。

在 MATLAB5.x 中，约束线性最小二乘用函数 `conls` 求解。

函数 **lsqlin**

格式 `x = lsqlin(C,d,A,b)` % 求在约束条件 $A \cdot x \leq b$ 下，方程 $Cx = d$ 的最小二乘解 x 。
`x = lsqlin(C,d,A,b,Aeq,beq)` % Aeq、beq 满足等式约束 $Aeq \cdot x = beq$ ，若没有不等式约束，则设 $A=[]$ ， $b=[]$ 。
`x = lsqlin(C,d,A,b,Aeq,beq,lb,ub)` % lb、ub 满足 $lb \leq x \leq ub$ ，若没有等式约束，则 $Aeq=[]$ ， $beq=[]$ 。
`x = lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0)` % x0 为初始解向量，若 x 没有界，则 $lb=[]$ ， $ub=[]$ 。
`x = lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0,options)` % options 为指定优化参数
`[x,resnorm] = lsqlin(...)` % $resnorm = \text{norm}(C*x-d)^2$ ，即 2-范数。
`[x,resnorm,residual] = lsqlin(...)` % $residual = C*x-d$ ，即残差。
`[x,resnorm,residual,exitflag] = lsqlin(...)` % exitflag 为终止迭代的条件
`[x,resnorm,residual,exitflag,output] = lsqlin(...)` % output 表示输出优化信息
`[x,resnorm,residual,exitflag,output,lambd] = lsqlin(...)` % lambda 为解 x 的 Lagrange 乘子

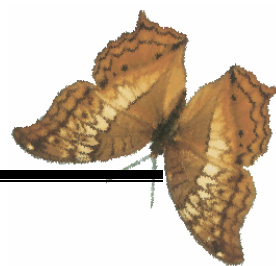
例 5-15 求解下面系统的最小二乘解

系统： $Cx=d$

约束： $A \cdot x \leq b$ ； $lb \leq x \leq ub$

先输入系统系数和 x 的上下界：

```
C = [0.9501    0.7620    0.6153    0.4057;...
      0.2311    0.4564    0.7919    0.9354;...
      0.6068    0.0185    0.9218    0.9169;...
      0.4859    0.8214    0.7382    0.4102;...
      0.8912    0.4447    0.1762    0.8936];
d = [ 0.0578; 0.3528; 0.8131; 0.0098; 0.1388];
A = [ 0.2027    0.2721    0.7467    0.4659;...
      0.1987    0.1988    0.4450    0.4186;...
      0.6037    0.0152    0.9318    0.8462];
b = [ 0.5251; 0.2026; 0.6721];
```



```

lb = -0.1*ones(4,1);
ub = 2*ones(4,1);
然后调用最小二乘命令：
[x,resnorm,residual,exitflag,output,lambda] = lsqlin(C,d,A,b,[],[],lb,ub);
结果为：
x =
    -0.1000
    -0.1000
     0.2152
     0.3502
resnorm =
     0.1672
residual =
     0.0455
     0.0764
    -0.3562
     0.1620
     0.0784
exitflag =
     1      %说明解 x 是收敛的
output =
    iterations: 4
    algorithm: 'medium-scale: active-set'
    firstorderopt: []
    cgiterations: []
lambda =
    lower: [4x1 double]
    upper: [4x1 double]
    eqlin: [0x1 double]
    ineqlin: [3x1 double]

```

通过 `lambda.ineqlin` 可查看非线性不等式约束是否有效。

5.7.2 非线性数据（曲线）拟合

非线性曲线拟合是已知输入向量 `xdata` 和输出向量 `ydata`，并且知道输入与输出的函数关系为 $ydata = F(x, xdata)$ ，但不知道系数向量 `x`。今进行曲线拟合，求 `x` 使得下式成立：

$$\min_x \frac{1}{2} \|F(x, xdata) - ydata\|_2^2 = \frac{1}{2} \sum_i (F(x, xdata_i) - ydata_i)^2$$

在 MATLAB5.x 中，使用函数 `curvefit` 解决这类问题。

函数 **lsqcurvefit**

格式

```

x = lsqcurvefit(fun,x0,xdata,ydata)
x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub)
x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub,options)
[x,resnorm] = lsqcurvefit(...)
[x,resnorm,residual] = lsqcurvefit(...)
[x,resnorm,residual,exitflag] = lsqcurvefit(...)
[x,resnorm,residual,exitflag,output] = lsqcurvefit(...)
[x,resnorm,residual,exitflag,output,lambda] = lsqcurvefit(...)

```

`[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqcurvefit(...)`

参数说明:

`x0` 为初始解向量; `xdata`, `ydata` 为满足关系 $ydata=F(x, xdata)$ 的数据;
`lb`、`ub` 为解向量的下界和上界 $lb \leq x \leq ub$, 若没有指定界, 则 `lb=[]`, `ub=[]`;
`options` 为指定的优化参数;
`fun` 为拟合函数, 其定义方式为: `x=lsqcurvefit(@myfun,x0,xdata,ydata)`,
 其中 `myfun` 已定义为 `function F=myfun(x,xdata)`
`F=...` % 计算 `x` 处拟合函数值 `fun` 的用法与前面相同;
`resnorm=sum((fun(x,xdata)-ydata).^2)`, 即在 `x` 处残差的平方和;
`residual=fun(x,xdata)-ydata`, 即在 `x` 处的残差;
`exitflag` 为终止迭代的条件;
`output` 为输出的优化信息;
`lambda` 为解 `x` 处的 Lagrange 乘子;
`jacobian` 为解 `x` 处拟合函数 `fun` 的 jacobian 矩阵。

例 5-16 求解如下最小二乘非线性拟合问题

已知输入向量 `xdata` 和输出向量 `ydata`, 且长度都是 `n`, 拟合函数为

$$ydata(i) = x(1) \cdot xdata(i)^2 + x(2) \cdot \sin(xdata(i)) + x(3) \cdot xdata(i)^3$$

即目标函数为 $\min_x \frac{1}{2} \sum_{i=1}^n (F(x, xdata_i) - ydata_i)^2$

其中: $F(x, xdata) = x(1) \cdot xdata^2 + x(2) \cdot \sin(xdata) + x(3) \cdot xdata^3$

初始解向量为 `x0=[0.3, 0.4, 0.1]`。

解: 先建立拟合函数文件, 并保存为 `myfun.m`

```
function F=myfun(x,xdata)
F=x(1)*xdata.^2+x(2)*sin(xdata)+x(3)*xdata.^3;
```

然后给出数据 `xdata` 和 `ydata`

```
>>xdata=[3.6 7.7 9.3 4.1 8.6 2.8 1.3 7.9 10.0 5.4];
>>ydata=[16.5 150.6 263.1 24.7 208.5 9.9 2.7 163.9 325.0 54.3];
>>x0=[10, 10, 10]; %初始估计值
>>[x,resnorm]=lsqcurvefit(@myfun,x0,xdata,ydata)
```

结果为:

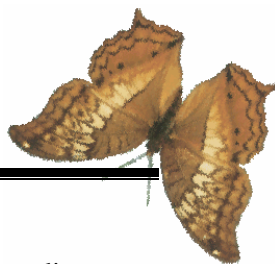
```
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
x =
0.2269    0.3385    0.3021
resnorm =
6.2950
```

5.7.3 非线性最小二乘

非线性最小二乘(非线性数据拟合)的标准形式为

$$\min_x f(x) = f_1(x)^2 + f_2(x)^2 + \cdots + f_m(x)^2 + L$$





其中: L 为常数

在 MATLAB5.x 中, 用函数 `leastsq` 解决这类问题, 在 6.0 版中使用函数 `lsqnonlin`。

$$\text{设 } F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

则目标函数可表达为 $\min_x \frac{1}{2} \|F(x)\|_2^2 = \frac{1}{2} \sum_i f_i(x)^2$

其中: x 为向量, $F(x)$ 为函数向量。

函数 `lsqnonlin`

格式 `x = lsqnonlin(fun,x0)` % x_0 为初始解向量; fun 为 $f_i(x)$, $i=1,2,\dots,m$, fun 返回向量值 F , 而不是平方和值, 平方和隐含在算法中, fun 的定义与前面相同。

`x = lsqnonlin(fun,x0,lb,ub)` % lb 、 ub 定义 x 的下界和上界: $lb \leq x \leq ub$ 。

`x = lsqnonlin(fun,x0,lb,ub,options)` % options 为指定优化参数, 若 x 没有界, 则 $lb=[]$, $ub=[]$ 。

`[x,resnorm] = lsqnonlin(...)` % $\text{resnorm} = \sum(\text{fun}(x).^2)$, 即解 x 处目标函数值。

`[x,resnorm,residual] = lsqnonlin(...)` % $\text{residual} = \text{fun}(x)$, 即解 x 处 fun 的值。

`[x,resnorm,residual,exitflag] = lsqnonlin(...)` % exitflag 为终止迭代条件。

`[x,resnorm,residual,exitflag,output] = lsqnonlin(...)` % output 输出优化信息。

`[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(...)` % λ 为 Lagrange 乘子。

`[x,resnorm,residual,exitflag,output,lambda,jacobian] = lsqnonlin(...)` % fun 在解 x 处的 Jacobian 矩阵。

例 5-17 求下面非线性最小二乘问题 $\sum_{k=1}^{10} (2 + 2k - e^{kx_1} - e^{kx_2})^2$ 初始解向量为 $x_0=[0.3, 0.4]$ 。

解: 先建立函数文件, 并保存为 `myfun.m`, 由于 `lsqnonlin` 中的 fun 为向量形式而不是平方和形式, 因此, `myfun` 函数应由 $f_i(x)$ 建立:

$$f_k(x) = 2 + 2k - e^{kx_1} - e^{kx_2} \quad k=1,2,\dots,10$$

`function F = myfun(x)`

`k = 1:10;`

`F = 2 + 2*k - exp(k*x(1)) - exp(k*x(2));`

然后调用优化程序:

`x0 = [0.3 0.4];`

`[x,resnorm] = lsqnonlin(@myfun,x0)`

结果为:

Optimization terminated successfully:

Norm of the current step is less than OPTIONS.TolX

$x =$

```

0.2578    0.2578
resnorm = %求目标函数值
124.3622

```

5.7.4 非负线性最小二乘

非负线性最小二乘的标准形式为：

$$\min_x \frac{1}{2} \|C x - d\|_2^2$$

sub.to $x \geq 0$

其中：矩阵 C 和向量 d 为目标函数的系数，向量 x 为非负独立变量。

在 MATLAB5.x 中，用函数 `nls` 求解这类问题，在 6.0 版中则用函数 `lsqnonneg`。

函数 lsqnonneg

格式 `x = lsqnonneg(C,d)` % C 为实矩阵， d 为实向量
`x = lsqnonneg(C,d,x0)` % $x0$ 为初始值且大于 0
`x = lsqnonneg(C,d,x0,options)` % `options` 为指定优化参数
`[x,resnorm] = lsqnonneg(...)` % `resnorm`= $\text{norm}(C*x-d)^2$
`[x,resnorm,residual] = lsqnonneg(...)` % `residual`= $C*x-d$
`[x,resnorm,residual,exitflag] = lsqnonneg(...)`
`[x,resnorm,residual,exitflag,output] = lsqnonneg(...)`
`[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)`

例 5-18 一个最小二乘问题的无约束与非负约束解法的比较。

先输入数据：

```

>>C = [ 0.0372  0.2869; 0.6861  0.7071; 0.6233  0.6245; 0.6344  0.6170];
>>d = [0.8587; 0.1781; 0.0747; 0.8405];
>> [C\d, lsqnonneg(C,d)]
ans =
-2.5627    0
 3.1108    0.6929

```

注意：1. 当问题为无约束线性最小二乘问题时，使用 MATLAB 下的 “\” 运算即可以解决。2. 对于非负最小二乘问题，调用 `lsqnonneg(C,d)` 求解。

5.8 非线性方程(组)求解

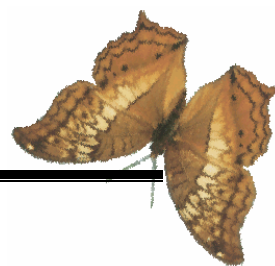
5.8.1 非线性方程的解

非线性方程的标准形式为 $f(x)=0$

函数 fzero

格式 `x = fzero(fun,x0)` % 用 `fun` 定义表达式 $f(x)$ ， $x0$ 为初始解。
`x = fzero(fun,x0,options)`
`[x,fval] = fzero(...)` % `fval`= $f(x)$
`[x,fval,exitflag] = fzero(...)`





`[x,fval,exitflag,output] = fzero(...)`

说明 该函数采用数值解求方程 $f(x)=0$ 的根。

例 5-19 求 $x^3 - 2x - 5 = 0$ 的根

解: `>> fun='x^3-2*x-5';`

`>> z=fzero(fun,2)` %初始估计值为 2

结果为

`z =`

`2.0946`

5.8.2 非线性方程组的解

非线性方程组的标准形式为: $F(x) = 0$

其中: x 为向量, $F(x)$ 为函数向量。

函数 **fsolve**

格式 `x = fsolve(fun,x0)` %用 `fun` 定义向量函数, 其定义方式为: 先定义方程函数

`function F = myfun(x)`。

`F=[表达式 1; 表达式 2; ...表达式 m]` %保存为 `myfun.m`, 并用下面方式调用:

`x = fsolve(@myfun,x0)`, x_0 为初始估计值。

`x = fsolve(fun,x0,options)`

`[x,fval] = fsolve(...)` %`fval=F(x)`, 即函数值向量

`[x,fval,exitflag] = fsolve(...)`

`[x,fval,exitflag,output] = fsolve(...)`

`[x,fval,exitflag,output,jacobian] = fsolve(...)` % `jacobian` 为解 x 处的 Jacobian 阵。

其余参数与前面参数相似。

例 5-20 求下列系统的根

$$2x_1 - x_2 = e^{-x_1}$$

$$-x_1 + 2x_2 = e^{-x_2}$$

解: 化为标准形式

$$2x_1 - x_2 - e^{-x_1} = 0$$

$$-x_1 + 2x_2 - e^{-x_2} = 0$$

设初值点为 $x_0 = [-5 \ -5]$ 。

先建立方程函数文件, 并保存为 `myfun.m`:

`function F = myfun(x)`

`F = [2*x(1) - x(2) - exp(-x(1));`

`-x(1) + 2*x(2) - exp(-x(2))];`

然后调用优化程序

`x0 = [-5; -5];` % 初始点

`options=optimset('Display','iter');` % 显示输出信息

`[x,fval] = fsolve(@myfun,x0,options)`

结果为

Iteration	Func-count	f(x)	Norm of step	First-order optimality	CG-iterations
1	4	47071.2	1	2.29e+004	0
2	7	6527.47	1.45207	3.09e+003	1
3	10	918.372	1.49186	418	1
4	13	127.74	1.55326	57.3	1
5	16	14.9153	1.57591	8.26	1
6	19	0.779051	1.27662	1.14	1
7	22	0.00372453	0.484658	0.0683	1
8	25	9.21617e-008	0.0385552	0.000336	1
9	28	5.66133e-017	0.000193707	8.34e-009	1

Optimization terminated successfully:

Relative function value changing by less than OPTIONS.TolFun

x =

0.5671

0.5671

fval =

1.0e-008 *

-0.5320

-0.5320

例 5-21 求矩阵 x 使其满足方程 $x * x * x = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, 并设初始解向量为 $x=[1, 1; 1, 1]$ 。

解: 先编写 M 文件:

```
function F = myfun(x)
```

```
F = x*x*x-[1,2;3,4];
```

然后调用优化程序求解:

```
>>x0 = ones(2,2); % 初始解向量
```

```
>>options = optimset('Display','off'); % 不显示优化信息
```

```
>>[x,Fval,exitflag] = fsolve(@myfun,x0,options)
```

则结果为

x =

-0.1291 0.8602

1.2903 1.1612

Fval =

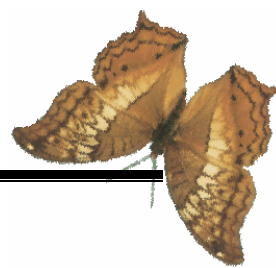
1.0e-003 *

0.1541 -0.1163

0.0109 -0.0243

exitflag =

1



第6章 模糊逻辑

6.1 隶属函数

6.1.1 高斯隶属函数

函数 **gaussmf**

格式 `y=gaussmf(x,[sig c])`

说明 高斯隶属函数的数学表达式为： $f(x;\sigma,c) = e^{-\frac{(x-c)^2}{2\sigma^2}}$ ，其中 σ, c 为参数， x 为自变量， sig 为数学表达式中的参数 σ 。

例 6-1

```
>>x=0:0.1:10;
>>y=gaussmf(x,[2 5]);
>>plot(x,y)
>>xlabel('gaussmf, P=[2 5]')
```

结果为图 6-1。

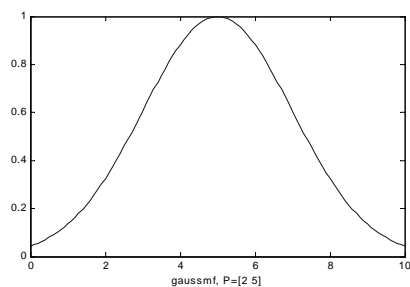


图 6-1

6.1.2 两边型高斯隶属函数

函数 **gauss2mf**

格式 `y = gauss2mf(x,[sig1 c1 sig2 c2])`

说明 sig1 、 $c1$ 、 sig2 、 $c2$ 为命令 1 中数学表达式中的两对参数

例 6-2

```
>>x = (0:0.1:10)';
>>y1 = gauss2mf(x, [2 4 1 8]);
>>y2 = gauss2mf(x, [2 5 1 7]);
>>y3 = gauss2mf(x, [2 6 1 6]);
>>y4 = gauss2mf(x, [2 7 1 5]);
>>y5 = gauss2mf(x, [2 8 1 4]);
```



```
>>plot(x, [y1 y2 y3 y4 y5]);
>>set(gcf, 'name', 'gauss2mf', 'numbertitle', 'off');
```

结果为图 6-2。

6.1.3 建立一般钟型隶属函数

函数 **gbellmf**

格式 $y = \text{gbellmf}(x, \text{params})$

说明 一般钟型隶属函数依靠函数表达式 $f(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}}$

这里 x 指定变量定义域范围, 参数 b 通常为正, 参数 c 位于曲线中心, 第二个参数变量 params 是一个各项分别为 a , b 和 c 的向量。

例 6-3

```
>>x=0:0.1:10;
>>y=gbellmf(x,[2 4 6]);
>>plot(x,y)
>>xlabel('gbellmf, P=[2 4 6]')
```

结果为图 6-3。

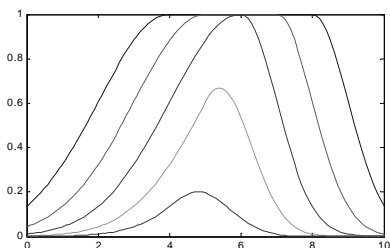


图 6-2

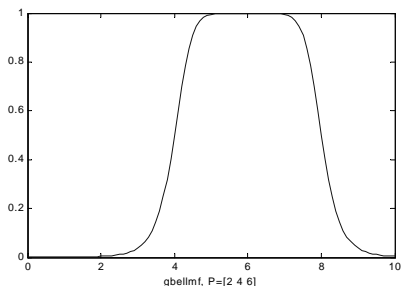


图 6-3

6.1.4 两个 sigmoid 型隶属函数之差组成的隶属函数

函数 **dsigmf**

格式 $y = \text{dsigmf}(x, [a_1 \ c_1 \ a_2 \ c_2])$

说明 这里 sigmoid 型隶属函数由下式给出 $f(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}$

x 是变量, a, c 是参数。dsigmf 使用四个参数 a_1 , c_1 , a_2 , c_2 , 并且是两个 sigmoid 型函数之差: $f_1(x; a_1, c_1) - f_2(x; a_2, c_2)$, 参数按顺序 $[a_1 \ c_1 \ a_2 \ c_2]$ 列出。

例 6-4

```
>>x=0:0.1:10;
>>y=dsigmf(x,[5 2 5 7]);
>>plot(x,y)
```

结果为图 6-4

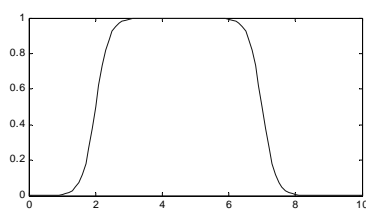
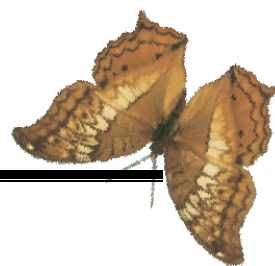


图 6-4

6.1.5 通用隶属函数计算

函数 **evalmf**

格式 $y = \text{evalmf}(x, \text{mfParams}, \text{mfType})$

说明 **evalmf** 可以计算任意隶属函数，这里 x 是变量定义域，**mfType** 是工具箱提供的一种隶属函数，**mfParams** 是此隶属函数的相应参数，如果你想创建自定义的隶属函数，**evalmf** 仍可以工作，因为它可以计算它不知道名字的任何隶属函数。

例 6-5

```
>>x=0:0.1:10;
>>mfparams = [2 4 6];
>>mfType = 'gbellmf';
>>y=evalmf(x,mfparams,mfType);
>>plot(x,y)
>>xlabel('gbellmf, P=[2 4 6]')
```

结果为图 6-5。

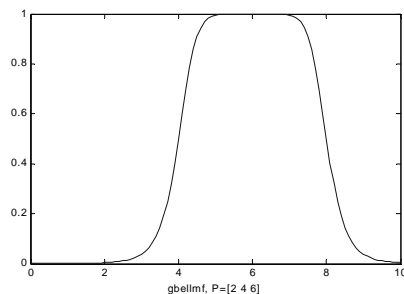


图 6-5

6.1.6 建立 II 型隶属函数

函数 **primf**

格式 $y = \text{primf}(x, [a \ b \ c \ d])$

说明 向量 x 指定函数自变量的定义域，该函数在向量 x 的指定点处进行计算，参数 $[a,b,c,d]$ 决定了函数的形状， a 和 d 分别对应曲线下部的左右两个拐点， b 和 c 分别对应曲线上部的左右两个拐点。

例 6-6



```
>>x=0:0.1:10;  
>>y=pimf(x,[1 4 5 10]);  
>>plot(x,y)  
>>xlabel('pimf, P=[1 4 5 10]')
```

结果为图 6-6。

6.1.7 通过两个 sigmoid 型隶属函数的乘积构造隶属函数

函数 **psigmf**

格式 $y = \text{psigmf}(x, [a_1 \ c_1 \ a_2 \ c_2])$

说明 这里 sigmoid 型隶属函数由下式给出 $f(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}$

x 是变量, a, c 是参数。psigmf 使用四个参数 a_1, c_1, a_2, c_2 , 并且是两个 sigmoid 型函数之积: $f_1(x; a_1, c_1) * f_2(x; a_2, c_2)$, 参数按顺序 $[a_1 \ c_1 \ a_2 \ c_2]$ 列出。

例 6-7

```
>>x=0:0.1:10;  
>>y=psigmf(x,[2 3 -5 8]);  
>>plot(x,y)  
>>xlabel('psigmf, P=[2 3 -5 8]')
```

结果为图 6-7。

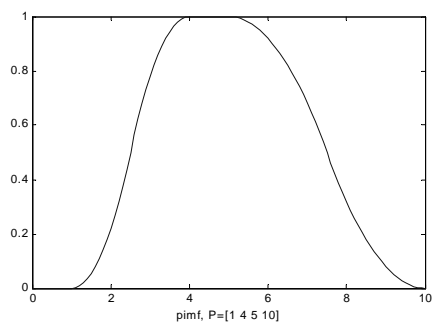


图 6-6

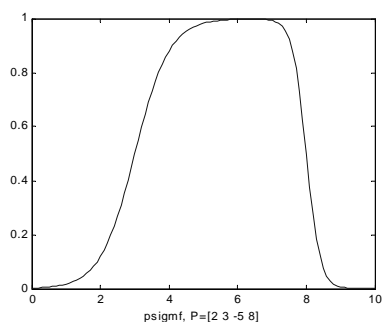


图 6-7

6.1.8 建立 Sigmoid 型隶属函数

函数 **sigmf**

格式 $y = \text{sigmf}(x, [a \ c])$

说明 $f(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}$, 定义域由向量 x 给出, 形状由参数 a 和 c 确定。

例 6-8

```
>>x=0:0.1:10;  
>>y=sigmf(x,[2 4]);  
>>plot(x,y)  
>>xlabel('sigmf, P=[2 4]')
```

结果为图 6-8。

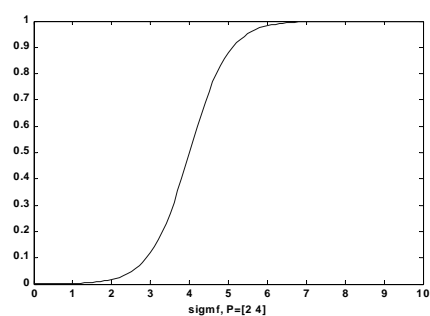
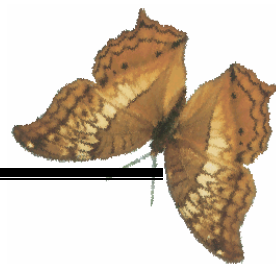


图 6-8

例 6-9

```
>>x = (0:0.2:10)';
>>y1 = sigmf(x,[-1 5]);
>>y2 = sigmf(x,[-3 5]);
>>y3 = sigmf(x,[4 5]);
>>y4 = sigmf(x,[8 5]);
>>subplot(2,1,1),plot(x,[y1 y2 y3 y4]);
>>y1 = sigmf(x,[5 2]);
>>y2 = sigmf(x,[5 4]);
>>y3 = sigmf(x,[5 6]);
>>y4 = sigmf(x,[5 8]);
>>subplot(2,1,2),plot(x,[y1 y2 y3 y4]);
```

结果为图 6-9。

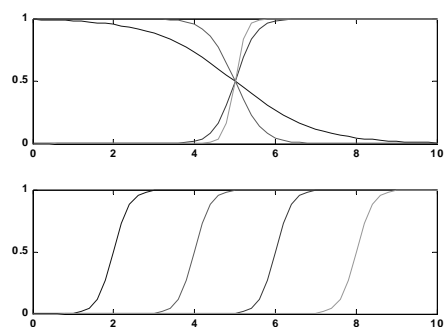


图 6-9

6.1.9 建立 S 型隶属函数

函数 **smf**

格式 $y = \text{smf}(x, [a \ b])$ % x 为变量, a 为 b 参数, 用于定位曲线的斜坡部分。

例 6-10

```
>>x=0:0.1:10;
>>y=smf(x,[1 8]);
>>plot(x,y)
```

结果为图 6-10。

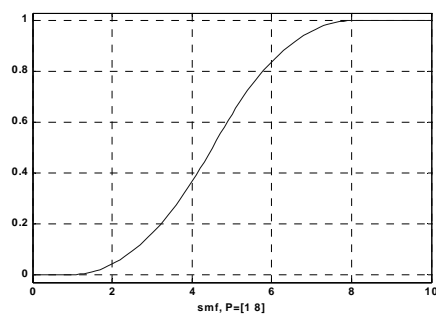


图 6-10

例 6-11

```
>>x = 0:0.1:10;
>>subplot(3,1,1);plot(x,smf(x,[2 8]));
>>subplot(3,1,2);plot(x,smf(x,[4 6]));
>>subplot(3,1,3);plot(x,smf(x,[6 4]));
```

结果为图 6-11。

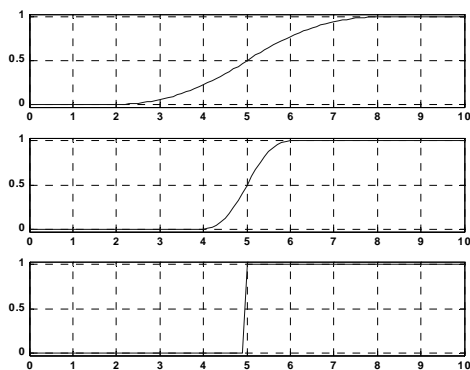


图 6-11

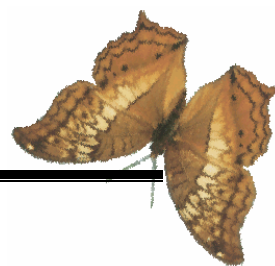
6.1.10 建立梯形隶属函数

函数 **trapmf**

格式 $y = \text{trapmf}(x,[a \ b \ c \ d])$

说明 这里梯形隶属函数表达式: $f(x;a,b,c,d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 0 & d \leq x \end{cases}$

或 $f(x;a,b,c,d) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0)$, 定义域由向量 x 确定, 曲线形状由参数 a, b, c, d 确定, 参数 a 和 d 对应梯形下部的左右两个拐点, 参数 b 和 c 对应梯形上部的左右两



个拐点。

例 6-12

```
>>x=0:0.1:10;
>>y=trapmf(x,[1 5 7 8]);
>>plot(x,y)
>>xlabel('trapmf, P=[1 5 7 8]')
```

结果为图 6-12。

例 6-13

```
>>x = (0:0.1:10)';
>>y1 = trapmf(x,[2 3 7 9]);
>>y2 = trapmf(x,[3 4 6 8]);
>>y3 = trapmf(x,[4 5 5 7]);
>>y4 = trapmf(x,[5 6 4 6]);
>>plot(x,[y1 y2 y3 y4]);
```

结果为图 6-13。

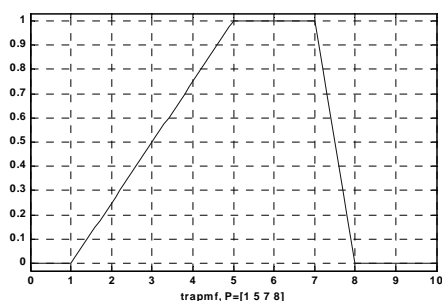


图 6-12

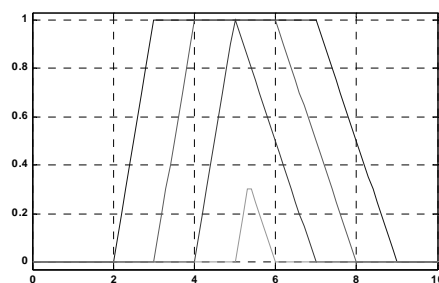


图 6-13

6.1.11 建立三角形隶属函数

函数 **trimf**

格式 $y = \text{trimf}(x, \text{params})$

$y = \text{trimf}(x, [a \ b \ c])$

说明 三角形隶属函数表达式: $f(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}$

或者 $f(x; a, b, c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0)$

定义域由向量 x 确定, 曲线形状由参数 a, b, c 确定, 参数 a 和 c 对应三角形下部的左右两个顶点, 参数 b 对应三角形上部的顶点, 这里要求 $a \leq b \leq c$, 生成的隶属函数总有一个统一的高度, 若想有一个高度小于统一高度的三角形隶属函数, 则使用 **trapmf** 函数。

例 6-14



```
>>x=0:0.1:10;
>>y=trimf(x,[3 6 8]);
>>plot(x,y)
>>xlabel('trimf, P=[3 6 8]')
```

结果为图 6-14。

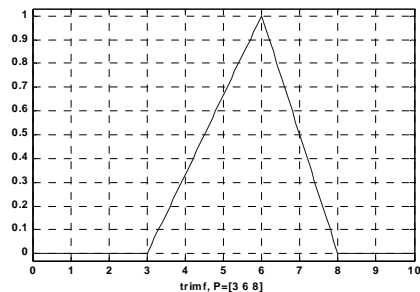


图 6-14

例 6-15

```
>>x = (0:0.2:10)';
>>y1 = trimf(x,[3 4 5]);
>>y2 = trimf(x,[2 4 7]);
>>y3 = trimf(x,[1 4 9]);
>>subplot(2,1,1),plot(x,[y1 y2 y3]);
>>y1 = trimf(x,[2 3 5]);
>>y2 = trimf(x,[3 4 7]);
>>y3 = trimf(x,[4 5 9]);
>>subplot(2,1,2),plot(x,[y1 y2 y3]);
```

结果为图 6-15。

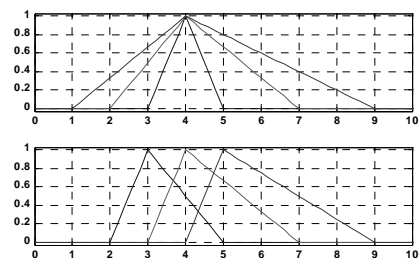


图 6-15

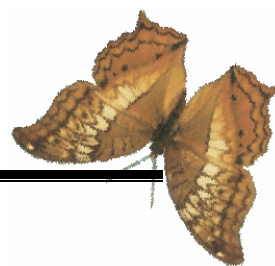
6.1.12 建立 Z 型隶属函数

函数 **zmf**

格式 $y = \text{zmf}(x, [a \ b])$ % x 为自变量, a 和 b 为参数, 确定曲线的形状。

例 6-16

```
>>x=0:0.1:10;
>>y=zmf(x,[3 7]);
>>plot(x,y)
```



```
>>xlabel('zmf, P=[3 7]')
```

结果为图 6-16。

例 6-17

```
>>x = 0:0.1:10;
>>subplot(3,1,1);plot(x,zmf(x,[2 8]));
>>subplot(3,1,2);plot(x,zmf(x,[4 6]));
>>subplot(3,1,3);plot(x,zmf(x,[6 4]));
```

结果为图 6-17。

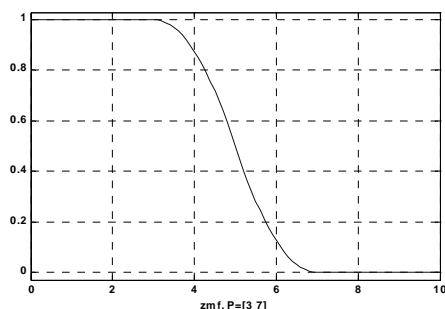


图 6-16

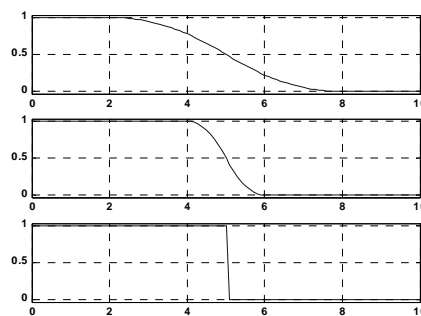


图 6-17

6.1.13 两个隶属函数之间转换参数

函数 **mf2mf**

格式 `outParams = mf2mf(inParams,inType,outType)`

说明 此函数根据参数集，将任意内建的隶属函数类型转换为另一种类型，`inParams` 为你要转换的隶属函数的参数，`inType` 为你要转换的隶属函数的类型的字符串名称，`outType` 为你要转换成的目标隶属函数的字符串名称。

例 6-18

```
>>x=0:0.1:5;
>>mfp1 = [1 2 3];
>>mfp2 = mf2mf(mfp1,'gbellmf','trimf');
>>plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

结果为图 6-18。

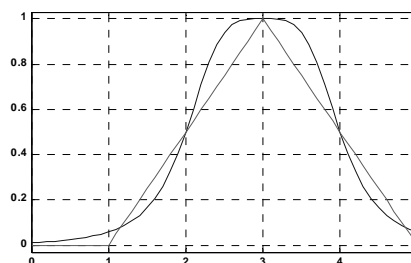


图 6-18

6.1.14 基本 FIS 编辑器

函数 **fuzzy**

格式 `fuzzy` %弹出未定义的基本 FIS 编辑器

`fuzzy(fismat)` %使用 `fuzzy('tipper')`，弹出下图 FIS 编辑器。

编辑器是任意模糊推理系统的高层显示，它允许你调用各种其它的编辑器来对其操作。此界面允许你方便地访问所有其它的编辑器，并以最灵活的方式与模糊系统进行交互。

方框图：窗口上方的方框图显示了输入、输出和它们中间的模糊规则处理器。单击任意一个变量框，使选中的方框成为当前变量，此时它变成红色高亮方框。双击任意一个变量，



弹出隶属度函数编辑器，双击模糊规则编辑器，弹出规则编辑器。

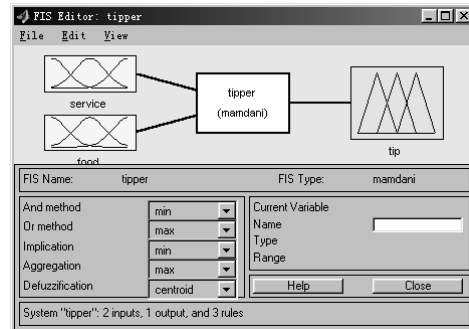


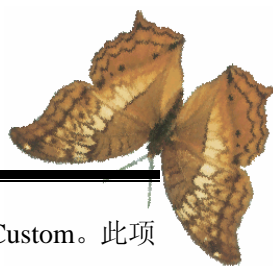
图 6-19

菜单项：FIS 编辑器的菜单棒允许你打开相应的工具，打开并保存系统。

- File 菜单包括：
 - New mamdani FIS ... 打开新 mamdani 型系统；
 - New Sugeno FIS ... 打开新 Sugeno 型系统；
 - Open from disk ... 从磁盘上打开指定的.fis 文件系统；
 - Save to disk 保存当前系统到磁盘上的一个.fis 文件上；
 - Save to disk as ... 重命名方式保存当前系统到磁盘上；
 - Open from workspace ... 从工作空间中指定的 FIS 结构变量装入一个系统；
 - Save to workspace ... 保存系统到工作空间中当前命名的 FIS 结构变量中；
 - Save to workspace as ... 保存系统到工作空间中指定的 FIS 结构变量中；
 - Close windows 关闭 GUI；
- Edit 菜单包括：
 - Add input 增加另一个输入到当前系统中；
 - Add output 增加另一个输出到当前系统中；
 - Remove variable 删除一个所选的变量；
 - Undo 恢复当前最近的改变；
- View 菜单包括：
 - Edit MFs ... 调用隶属度函数编辑器；
 - Edit rules ... 调用规则编辑器；
 - Edit anfis ... 只对单输出 Sugeno 型系统调用编辑器；
 - View rules ... 调用规则观察器；
 - View surface ... 调用曲面观察器。

弹出式菜单：用五个弹出式菜单来改变模糊蕴含过程中五个基本步骤的功能：

- And method: 为一个定制操作选择 min、prod 或 Custom；
- Or method: 为一个定制操作选择 max、probor（概率）或 Custom；
- Implication method: 为一个定制操作选择 min、prod 或 Custom；此项对 Sugeno 型模糊系统不可用。



- Aggregation method: 为一个定制操作选择 max、sum、probor 或 Custom。此项对 Sugeno 型模糊系统不可用。
- Defuzzification method: 对 Mamdani 型推理, 为一个定制操作选择 centroid (面积中心法)、bisector (面积平分法)、mom (平均最大隶属度法)、som (最大隶属度最小值法)、lom (最大隶属度最大值法) 或 Custom。对 Sugeno 型推理, 在 wtaver (加权平均) 或 wtsum (加权和) 之间选择。

6.1.15 隶属函数编辑器

函数 **mfedit**

格式 **mfedit('a')**

mfedit(a)

mfedit

说明 **mfedit('a')** 生成一个隶属函数编辑器, 他允许你检查和修改存储在文件 **a.fis** 中 FIS 结构的所有隶属函数。如图, **mfedit('tank')** 以这种方式打开隶属函数编辑器并装入 **tank.fis** 中存储的所有隶属函数。

mfedit(a) 对于 FIS 结构操作一个 MATLAB 工作空间变量 **a**。Mfedit 可单独弹出没有装入 FIS 的隶属函数编辑器

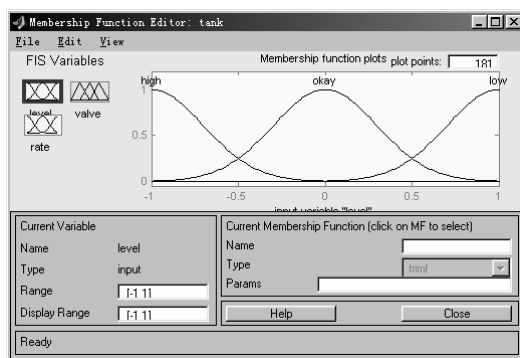


图 6-20

菜单项: 在 ANFIS 编辑器 GUI 上, 有一个菜单棒允许你打开相关的 GUI 工具、打开和保存系统等。File 菜单与 FIS 编辑器上的 File 菜单功能相同。

- Edit 菜单项包括:
 - Add MF... 为当前语言变量增加隶属度函数;
 - Add custom MF... 为当前语言变量增加定制的隶属度函数;
 - Remove current MF 删除当前的隶属度函数;
 - Remove all MFS 删除当前语言变量的所有隶属度函数;
 - Undo 恢复当前最近的改变。
- View 菜单项包括:
 - Edit FIS properties... 调用 FIS 编辑器;



Edit rules... 调用规则编辑器;
View rules... 调用规则观察器;
View surface... 调用曲面观察器。

6.2 模糊推理结构 FIS

6.2.1 不使用数据聚类方法从数据生成 FIS 结构

函数 **genfis1**

格式 **fismat = genfis1(data)**

fismat = genfis1(data,numMFs,inmftype, outmftype)

说明 **genfis1** 为 **anfis** 训练生成一个 **Sugeno** 型作为初始条件的 **FIS** 结构（初始隶属函数）。**genfis1(data,numMFs,inmftype, outmftype)**使用对数据的网格分割方法，从训练数据集生成一个 **FIS** 结构。**Data** 是训练数据矩阵，除最后一列表示单一输出数据外，它的其它各列表示输入数据。**NumMFs** 是一个向量，它的坐标指定与每一输入相关的隶属函数的数量。如果你想使用每个输入相关的相同数量的隶属函数，那么只须使 **numMFs** 成为一个数就足够了。**Inmftype** 是一个字符串数组，它的每行指定与每个输入相关的隶属函数类型。**outmftype** 是一个字符串数组，它的指定与每个输出相关的隶属函数类型

例 6-19

```
>>data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];  
>>numMFs = [3 7];  
>>mfType = str2mat('pimf','trimf');  
>>fismat = genfis1(data,numMFs,mfType);  
>> [x,mf] = plotmf(fismat,'input',1);  
>>subplot(2,1,1), plot(x,mf);  
>>xlabel('input 1 (pimf)');  
>>[x,mf] = plotmf(fismat,'input',2);  
>>subplot(2,1,2), plot(x,mf);  
>>xlabel('input 2 (trimf)');
```

结果为图 6-21。

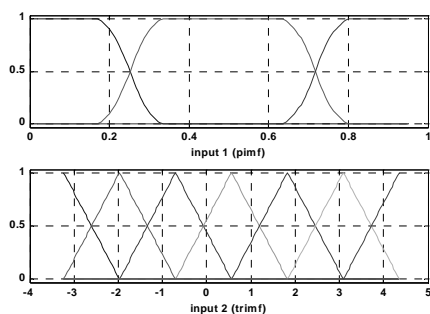
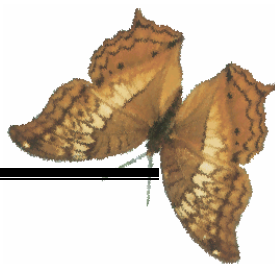


图 6-21



6.2.2 使用减法聚类方法从数据生成 FIS 结构

函数 **genfis2**

格式 `fismat = genfis2(Xin,Xout,radii)`

`fismat = genfis2(Xin,Xout,radii,xBounds)`

`fismat = genfis2(Xin,Xout,radii,xBounds,options)`

说明 `Xin` 是一个矩阵，它的每一行包含一个数据点的输入值；`Xout` 是一个矩阵，它的每一行包含一个数据点的输出值；`radii` 是一个向量，它指定一个聚类中心在一个数据维上作用的范围，这里假定数据位于一个单位超立方体内；`xBounds` 是一个 $2 \times N$ 可选矩阵，它用于指定如何将 `Xin` 和 `Xout` 中的数据映射到一个超立方体内，这里是数据的维数（行数）；`options` 是一个可选向量，它指定的值用于覆盖算法参数的缺省值。

例 6-20

```
fismat = genfis2(Xin,Xout,0.5)
```

这是使用此函数所需的最小变量数。这里对所有数据维指定 0.5 的作用范围。

```
fismat = genfis2(Xin,Xout,[0.5 0.25 0.3])
```

这里假定组合的维数是 3。假设 `Xin` 有两维、`Xout` 有一维，那么，0.5 和 0.25 是 `Xin` 数据维中每一维的作用范围，0.3 是 `Xout` 数据维的作用范围。

```
fismat = genfis2(Xin,Xout,0.5,[-10 -5 0; 10 5 20])
```

这里指定了如何将 `Xin` 和 `Xout` 中的数据规范化为 $[0 \ 1]$ 区间中的值来进行处理。假设 `Xin` 有两维、`Xout` 有一维，那么 `Xin` 第一列中的数据是从 $[-10 \ +10]$ 比例变换后的值，`Xin` 第二列中的数据是从 $[-5 \ +5]$ 比例变换后的值，`Xout` 中的数据是从 $[0 \ 20]$ 比例变换后的值。

6.2.3 生成一个 FIS 输出曲面

函数 **gensurf**

格式 `gensurf(fis)` % 使用前两个输入和第一个输出来生成给定模糊推理系统(fis)的输出曲面

`gensurf(fis,inputs,output)` % 使用分别由向量 `inputs` 和标量 `output` 给定的输入（一个或两个）和输出（只允许一个）来生成一个图形。

`gensurf(fis,inputs,output,grids)` % 指定 `X`（第一、水平）和 `Y`（第二、垂直）方向的网格数。如果是二元向量，`X` 和 `Y` 方向上的网格可以独立设置。

`gensurf(fis,inputs,output,grids,refinput)` % 用于多于两个的输入，`refinput` 向量的长度与输入相同：

- 将对应于要显示的输入的 `refinput` 项，设置为 NaN；
- 对其它输入的固定值设置为双精度实标量。

`[x,y,z]=gensurf(...)` % 返回定义输出曲面的变量并且删除自动绘图。

例 6-21

```
>>a = readfis('tipper');
```

```
>>gensurf(a)
```

结果为图 6-22。

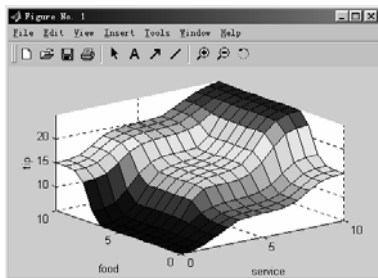


图 6-22

6.2.4 将 mamdan 型 FIS 转换为 Sugeno FIS

函数 **mam2sug**

格式 `sug_fis=mam2sug(mam_fis)`

说明 该函数将一个 mamdani 型 FIS 结构（不必是单输出）`mam_fis` 转化为一个 sugeno 型结构 `sug_fis`。返回的 sugeno 型系统具有常值输出隶属度函数。这些常值由原来 mamdani 型系统的后件的隶属度函数的面积中心法来确定。前件仍保持不变。

6.2.5 完成模糊推理计算

函数 **evalfis**

格式 `output=evalfis(input,fismat)`

`output=evalfis(input,fismat,numPts)`

`[output,IRR,ORR,ARR]=evalfis(input,fismat)`

`[output,IRR,ORR,ARR]=evalfis(input,fismat,numPts)`

说明 **input**: 指定输入值的一个数或一个矩阵，如果输入是一个 $M \times N$ 矩阵，其中 N 是输入变量数，那么 `evalfis` 使用 `input` 的每一行作为一个输入向量，并且为变量 `output` 返回 $M \times L$ 矩阵，该矩阵每一行是一个向量并且 L 是输出变量数；

fismat: 要计算的一个 FIS 结构；

numPts: 一个可选变量，它表示在输入或输出范围内的采样点数，在这些点上计算隶属函数，如果 不使用此变量，就使用 101 点的缺省值。

Evalfis 的值域如下：

Output: 大小为 ML 的输出矩阵，这里 M 表示前面指定的输入值的数量， L 表示 FIS 的输出变量数。

evalfis 的可选值域变量只有当 `input` 是一个行向量时才计算这些可选值域变量是：

IRR: 通过隶属函数计算的输入变量的结果，这是一个大小为 $\text{numRules}N$ 的矩阵，这里 numRules 是规则条数， N 是输入变量数。

ORR: 通过隶属函数计算的输出变量的结果，这是一个大小为 $\text{numPtsnumRules}L$ 的矩阵，这里 numRules 是规则条数， L 是输出变量数，此矩阵的第一组 numRules 列，对应于第一个输出，第二组



numRules 对应于第二个输出，依次类推。

ARR: 对每个输出，在输出值域中，numPts 处采样合成值的 numPtsL 矩阵，当只有一个值域变量调用时，该函数使用由结构 fismat 指定的模糊推理系统，由标量或矩阵 inout 指定的输入值计算输出向量 output。

例 6-22

```
>>fismat = readfis('tipper');
>>out = evalfis([2 1; 4 9],fismat)
```

结果为

```
out =
    7.0169
   19.6810
```

6.2.6 模糊 c 均值聚类

函数 **fcm**

格式 `[center,U,obj_fcn] = fcm(data,cluster_n)`

说明 对给定的数据集应用模糊 c 均值聚类方法进行聚类

data: 要聚类的数据集，每行是一个采样数据点；

cluster_n: 聚类中心的个数（大于 1）

center: 迭代后得到的聚类中心的矩阵，这里每行给出聚类中心的坐标；

U: 得到的所有点对聚类中心的模糊分类矩阵或隶属度函数矩阵；

Obj_fcn: 迭代过程中，目标函数的值；

fcm(data,cluster_n,options)使用可选的变量 options 控制聚类参数。包括停止准则，和/或设置迭代信息显示：

options(1): 分类矩阵 U 的指数，缺省值是 2.0；

options(2): 最大迭代次数，缺省值是 100；

options(3): 最小改进量，即迭代停止的误差准则，缺省值是 1e-5；

option(4): 迭代过程中显示信息，缺省值是 1。

如果任意一项为 NaN，这些选项就使用缺省值；当达到最大迭代次数时，或目标函数两次连续迭代的改进量小于指定的最小改进量，即满足停止误差准则时，聚类过程结束。

例 6-23

```
>>data = rand(100, 2);
>>[center,U,obj_fcn] = fcm(data, 2);
>>plot(data(:,1), data(:,2),'o');
>>maxU = max(U);
>>index1 = find(U(1,:) == maxU);
>>index2 = find(U(2,:) == maxU);
>>line(data(index1,1), data(index1, 2), 'linestyle', 'none',
'marker', '*', 'color', 'g');
>>line(data(index2,1), data(index2, 2), 'linestyle', 'none',
'marker', '*', 'color', 'r');
```

结果为图 6-23。

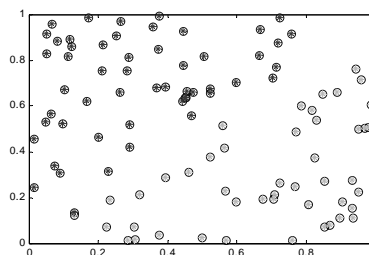


图 6-23

6.2.7 模糊均值和减法聚类

函数 **findcluster**



格式 `findcluster`

`findcluster('file.dat')`

说明 `findcluster` 产生一个 GUI 上的 Method 下的下拉式标签, 可以实现模糊 C 均值(fcm)或模糊减法聚类(subtractiv), 使用 Load Data 按钮输入数据, 刚进入 GUI 时, 对每种方法的选项都设置为缺省值。

此工具使用多维数据集, 但只显示这些维数中的两维。使用 X-axis 和 Y-axis 下的下拉式标签选择你想观察的数据维。例如你有一个五维数据集, 按照出现在数据集中的顺序, 此工具将数据标记为 data_1, data_2, data_3, data_4, data_5, Start 将完成聚类, Save Centre 将保存聚类中心。

当使用数据集 file.data 时, `findcluster(file.dat)` 自动装入数据集, 并且只绘制数据集中的前两维。产生 GUI 后, 你仍可以选择要聚类数据的那两维。

例 6-24

```
>>findcluster('clusterdemo.dat')
```

结果为图 6-24。

6.2.8 绘制一个 FIS

函数 `plotfis`

格式 `plotfis(fismat)`

说明 此函数显示由 fismat 指定的一个 FIS 的高层方框图, 输入和它们的隶属函数出现在结构特征图的左边, 同时输出和它们的隶属函数出现在结构特征图的右边。

例 6-25

```
>>a = readfis('tipper');
```

```
>>plotfis(a)
```

结果为图 6-25。

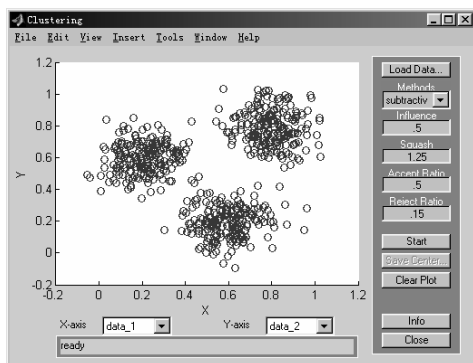


图 6-24

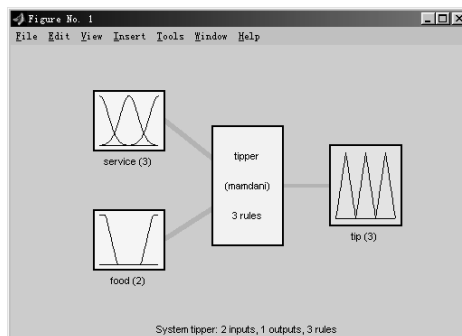


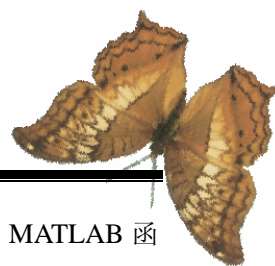
图 6-25

6.2.9 绘制给定变量的所有隶属的曲线

函数 `plotmf`

格式 `plotmf(fismat,varType,varIndex)`

说明 此函数绘制与给定变量相关的称为 fismat 的 FIS 中的所有隶属函数曲线, 变量的



类型和索引分别由 varType ('input' 或 'output') 和 varIndex 给出。此函数也可以与 MATLAB 函数 subplot 一起使用。

例 6-26

```
>>a = readfis('tipper');
>>plotmf(a,'input',1)
```

结果为图 6-26。

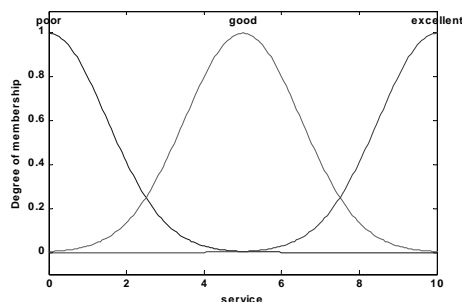


图 6-26

6.2.10 从磁盘装入一个 FIS

函数 readfis

格式 fismat = readfis('filename')

说明 从磁盘上的一个 .fis 文件(由 filename 命名)读出一个模糊推理系统, 并将产生的 FIS 装入当前的工作空间中。Fismat = readfis 不带输入变量, 即没有指定文件名时, 使用 uigetfile 命令打开一个对话框, 提示用户指定文件的名称和目录位置。

例 6-27

```
>>fismat = readfis('tipper');
>>getfis(fismat)
```

返回结果

```
getfis(fismat)
    Name      = tipper
    Type      = mamdani
    NumInputs = 2
    InLabels  =
        service
        food
    NumOutputs = 1
    OutLabels =
        tip
    NumRules = 3
    AndMethod = min
    OrMethod  = max
    ImpMethod = min
    AggMethod = max
    DefuzzMethod = centroid

ans =
tipper
```



6.2.11 从 FIS 中删除某一隶属函数

函数 **rmmf**

格式 `fis = rmmf(fis,'varType',varIndex,'mf',mfIndex)`

说明 从与工作空间 FIS 结构 `fis` 相关的模糊推理系统中删除变量类型为 `varType`，索引为 `varIndex` 的隶属函数 `mfIndex`。

字符串 `vartype` 必须是 'input' 或 'output'。

`varIndex` 是表示变量索引的一个整数，此索引表示列出变量的顺序；

变量 'mf' 是表示隶属函数的一个字符串；

`mfIndex` 是表示隶属函数索引的一个整数，此索引表示列出隶属函数的顺序。

例 6-28

```
>>a = newfis('mysys');
>>a = addvar(a,'input','temperature',[0 100]);
>>a = addmf(a,'input',1,'cold','trimf',[0 30 60]);
>>getfis(a,'input',1)
```

返回结果

```
      Name =      temperature
      NumMFs =      1
      MFLabels =
           cold
      Range =      [0 100]
ans =
     []
>>b = rmmf(a,'input',1,'mf',1);
>>getfis(b,'input',1)
```

返回

```
      Name =      temperature
      NumMFs =      0
      MFLabels =
      Range =      [0 100]
ans =
     []
```

6.2.12 从 FIS 中删除变量

函数 **rmvar**

格式 `[fis2,errorStr] = rmvar(fis,'varType',varIndex)`

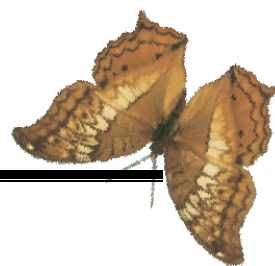
`fis2 = rmvar(fis,'varType',varIndex)`

说明 `fis2 = rmvar(fis,'varType',varIndex)` 从与工作空间 FIS 结构 `fis` 相关的模糊推理系统中删除索引为 `varIndex` 的语言变量 `mfIndex`，字符串 `vartype` 必须是 'input' 或 'output'。

`varIndex` 是表示变量索引的一个整数，此索引表示列出变量的顺序。

`[fis2,errorStr] = rmvar(fis,'varType',varIndex)` 将任何错误信息返回到字符串 `errorStr`。

此命令自动更新规则列表以保证列表尺寸与当前变量数保持一致，在删除语言变量之前，你必须从 FIS 删除任何包含要删除变量的规则，你无法删除在规则列表中正在使用的模



糊变量。

例 6-29

```
>>a = newfis('mysys');
>>a = addvar(a,'input','temperature',[0 100]);
>>getfis(a)
```

返回:

```
Name      = mysys
Type       = mamdani
NumInputs  = 1
InLabels   =
           temperature
NumOutputs = 0
OutLabels  =
NumRules   = 0
AndMethod  = min
OrMethod   = max
ImpMethod  = min
AggMethod  = max
DefuzzMethod = centroid
```

```
ans =
mysys
>>b = rmvar(a,'input',1);
>>getfis(b)
```

返回:

```
Name      = mysys
Type       = mamdani
NumInputs  = 0
InLabels   =
NumOutputs = 0
OutLabels  =
NumRules   = 0
AndMethod  = min
OrMethod   = max
ImpMethod  = min
AggMethod  = max
DefuzzMethod = centroid
```

```
ans =
mysys
```

6.2.13 设置模糊系统属性

函数 **setfis**

格式 **a = setfis(a,'fispropname','newfisprop')**

a = setfis(a,'vartype',varindex,'varpropname','newvarprop')

a = setfis(a,'vartype',varindex,'mf',mfindex, 'mfpropname','newmfprop');

说明 可以使用三个、五个或七个输入变量调用 **setfis** 命令，使用几个输入变量取决于是否设置整个结构的一个属性，是否设置属于该结构的一个特定变量，还是是否设置属于这些变量之一的一个特定隶属函数。这些变量是：

a: 工作空间中 **FIS** 的一个变量名称，

vartype: 表示变量类型的一个字符串：input 或 output；



varindex: 输入或输出变量的索引;
mf: 调用 `setfis` 时, 七个变量中的第四个变量所用的字符串, 用语指明此变量是一个隶属函数;
mfindex: 属于所选变量的隶属函数的索引;
fispropname: 表示你要设置 FIS 域属性的一个字符串: `name,type,andmethod,ormethod,impmethod,aggmethod,defuzzmethod`;
newfisprop: 你要设置的 FIS 的属性或方法名称的一个字符串;
varpropname: 你要设置的变量域名称的一个字符串: `name` 或 `range`;
newvarprop: 你要设置的变量名称的一个字符串 (对 `name`), 或变量范围的一个数组 (对 `range`), `mfpropname`—你要设置的隶属函数名称的一个字符串: `name,type` 或 `params`;
newmfprop: 你要设置的隶属函数名称或类型域的一个字符串 (对 `name` 或 `type`), 或者是参数范围的一个数组 (对 `params`)。

例 6-30 使用三个变量调用:

```
>>a = readfis('tipper');  
>>a2 = setfis(a, 'name', 'eating');  
>>getfis(a2, 'name');
```

结果为:

```
out =  
    eating
```

如果使用五个变量, `setfis` 将更新两个变量属性:

```
>>a2 = setfis(a,'input',1,'name','help');  
>>getfis(a2,'input',1,'name')
```

结果为:

```
ans =  
    help
```

如果使用七个变量, `setfis` 将更新七个隶属函数的任意属性:

```
>>a2 = setfis(a,'input',1,'mf',2,'name','wretched');  
>>getfis(a2,'input',1,'mf',2,'name')
```

结果为:

```
ans =  
    wretched
```

6.2.14 以分行形式显示 FIS 结构的所有属性

函数 **showfis**

格式 `showfis(fismat)`

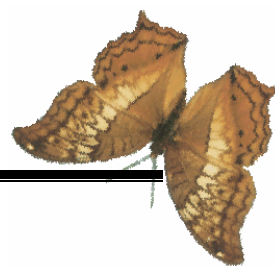
说明 以分行方式显示 MATLAB 工作空间 FIS 变量 `fismat`, 允许你查看结构的每个域的意义和内容。

例 6-31

```
>>a = readfis('tipper');  
>>showfis(a)
```

返回:

```
1. Name          tipper  
2. Type          mamdani
```



3. Inputs/Outputs	[2 1]
4. NumInputMFs	[3 2]
5. NumOutputMFs	3
6. NumRules	3
7. AndMethod	min
8. OrMethod	max
9. ImpMethod	min
10. AggMethod	max
11. DefuzzMethod	centroid
12. InLabels	service
13.	food
14. OutLabels	tip
15. InRange	[0 10]
16.	[0 10]
17. OutRange	[0 30]
18. InMFLabels	poor
19.	good
20.	excellent
21.	rancid
22.	delicious
23. OutMFLabels	cheap
24.	average
25.	generous
26. InMFTypes	gaussmf
27.	gaussmf
28.	gaussmf
29.	trapmf
30.	trapmf
31. OutMFTypes	trimf
32.	trimf
33.	trimf
34. InMFParams	[1.5 0 0 0]
35.	[1.5 5 0 0]
36.	[1.5 10 0 0]
37.	[0 0 1 3]
38.	[7 9 10 10]
39. OutMFParams	[0 5 10 0]
40.	[10 15 20 0]
41.	[20 25 30 0]
42. Rule Antecedent	[1 1]
43.	[2 0]
44.	[3 2]
42. Rule Consequent	1
43.	2
44.	3
42. Rule Weigth	1
43.	1
44.	1
42. Rule Connection	2
43.	1
44.	2

6.2.15 完成模糊运算

函数 fuzarith



格式 $C = \text{fuzarith}(X, A, B, \text{operator})$

说明 使用区间算法, $C = \text{fuzarith}(X, A, B, \text{operator})$ 返回一个模糊集 C 作为结果, 该算法使用由字符串 operator 表示的函数, 并在采样凸模糊集 A 和 B 上完成二进制运算; 元素 A 和 B 由采样值域变量 X 的凸函数产生;

A , B 和 X 是相同维数的向量;

operator 是下列串之一: 'sum', 'sub', 'prod', and 'div';

该函数返回的模糊集 C 是一个与 X 具有相同长度的列向量

例 6-32

```
>>point_n = 101;% this determines MF's resolution
>>min_x = -20; max_x = 20;% universe is [min_x, max_x]
>>x = linspace(min_x, max_x, point_n);
>>A = trapmf(x, [-10 -2 1 3]);% trapezoidal fuzzy set A
>>B = gaussmf(x, [2 5]);% Gaussian fuzzy set B
>>C1 = fuzarith(x, A, B, 'sum');
>>subplot(2,1,1);
>>plot(x, A, 'b--', x, B, 'm:', x, C1, 'c');
>>title('fuzzy addition A+B');
>>C2 = fuzarith(x, A, B, 'sub');
>>subplot(2,1,2);
>>plot(x, A, 'b--', x, B, 'm:', x, C2, 'c');
>>title('fuzzy subtraction A-B');
>>C3 = fuzarith(x, A, B, 'prod');
```

结果为图 6-27。

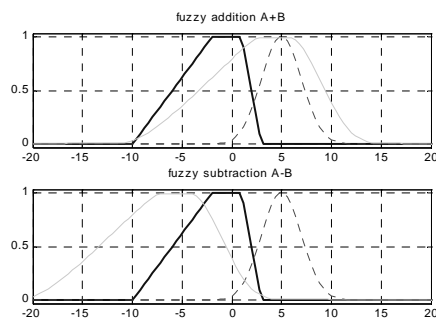


图 6-27

6.2.16 解析模糊规则

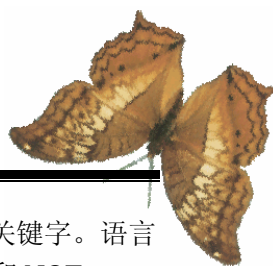
函数 **parsrule**

格式 $\text{fis2} = \text{parsrule}(\text{fis}, \text{txtRuleList})$

$\text{fis2} = \text{parsrule}(\text{fis}, \text{txtRuleList}, \text{ruleFormat})$

$\text{fis2} = \text{parsrule}(\text{fis}, \text{txtRuleList}, \text{ruleFormat}, \text{lang})$

说明 此函数为 MATLAB 工作空间 FIS 变量 fis 解析定义规则(txtRuleList)的文本, 并且返回添加了相应规则列表的一个 FIS 结构。如果原始输入 FIS 结构 fis 有任意初始规则, 他们将由新结构 fis2 替换。本函数支持三种不同的规则格式(由 ruleFormat 指定'verbose'(语言型)、'symbolic'(符号型)、'indexed'(索引型)。缺省格式是'verbose'(语言型)。当使用可选语



言变量 lang 时, 规则以语言型格式进行解析, 并采用语言变量 lang 中指定的关键字。语言必须是 'english'、'français' 或 'deutsch'。英语关键字是 if、then、is、AND、OR 和 NOT。

例 6-33

```
>>a = readfis('tipper');
>>ruleTxt = 'if service is poor then tip is generous';
>>a2 = parsrule(a,ruleTxt,'verbose');
>>showrule(a2)
```

结果为

```
ans =
    1. If (service is poor) then (tip is generous) (1)
```

6.2.17 规则编辑器和语法编辑器

函数 ruleedit

格式 ruleedit('a')

ruleedit(a)

说明 当使用 ruleedit('a')调用规则编辑器时, 可用于修改存储在文件 a.fis 中的一个 FIS 结构的规则。它也可用于检查模糊推理系统使用的规则。为使用编辑器创建规则, 你必须首先用 FIS 编辑器定义要使用的所有输入输出变量, 你可以使用列表框和检查框选择输入、输出变量, 连接操作和权重来创建新规则。如图所示, 用 ruleedit('tank')打开规则编辑器并装入 tank.fis 中存储的所有规则。

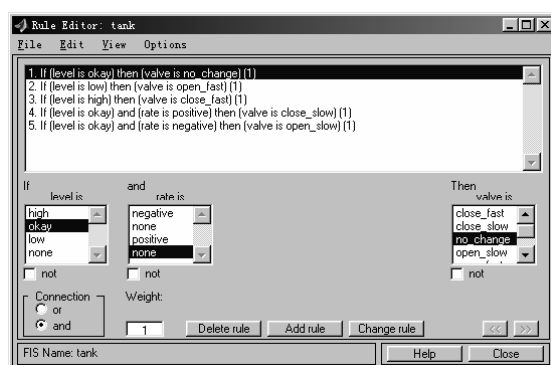


图 6-28

菜单项: 在规则编辑器 GUI 上, 有一个菜单棒允许你打开相关的 GUI 工具、打开和保存系统等。File 菜单与 FIS 编辑器上的 File 菜单功能相同。

- Edit 菜单项包括:

Undo 用于恢复最近的改变;

- View 菜单项包括:

Edit FIS properties... 调用 FIS 编辑器;

Edit membership functions... 调用隶属度函数编辑器;

Edit rules... 调用规则编辑器;

View surface... 调用曲面观察器。



- Options 菜单项包括：
 - Language 用于选择语言：English、Deutsch 和 Francais；
 - Format 用于选择格式
- Verbose 使用单词“if”、“then”、“AND”、“OR”等创建实际语句。
- Symbolic 用某些符号代替 Verbose 模式中使用的单词。例如：“if A AND B then C”成为“A&B=>C”。
- indexed 表示规则如何在 FIS 结构中存储。

6.2.18 规则观察器和模糊推理框图

函数 **ruleview**

格式 **ruleview('a')**

说明 使用 **ruleview('a')** 调用规则观察器时，将绘制在存储文件 a.fis 中的一个 FIS 的模糊推理框图。它用于观察从开始到结束整个蕴含过程。你可以移动对应输入的指示线，然后观察系统重新调节并计算新的输出。如图 6-29: **ruleview('tank')**

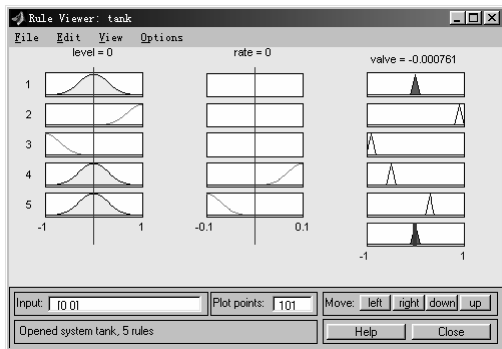


图 6-29

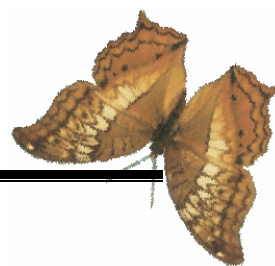
菜单项：在规则编辑器 GUI 上，有一个菜单棒允许你打开相关的 GUI 工具、打开和保存系统，等等。File 菜单与 FIS 编辑器上的 File 菜单功能相同。

- View 菜单项包括：
 - Edit FIS properties... 调用 FIS 编辑器；
 - Edit membership functions... 调用隶属度函数编辑器；
 - Edit rules... 调用规则编辑器；
 - View surface... 调用曲面观察器。
- Options 菜单项包括：
 - Rules display format 用于选择显示规则的格式。如果单击模糊推理方框图左边的规则序号，与该序号相关的规则出现在规则观察器底部的状态棒中。

6.2.19 保存 FIS 到磁盘上

函数 **writefis**

格式 **writefis(fismat)**



```
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
```

说明 writefis 将一个 MATLAB 工作空间 FIS 结构 fismat 用一个 .fis 文件形式保存到磁盘上；

writefis(fismat)产生一个对话框让用户输入文件的名称和存放文件的目录；

writefis(fismat,'filename')将对应于 FIS 结构 fismat 的一个 .fis 文件写到一个称为 filename.fis 的磁盘文件中，不使用对话框该文件被保存在当前目录中；

writefis(fismat,'filename','dialog')创建一个带有提供的缺省名为 filename.fis 的对话框；

若扩展名不存在，则只为 filename 添加 .fis 扩展名。

例 6-34

```
>>a = newfis('tipper');
>>a = addvar(a,'input','service',[0 10]);
>>a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
>>a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
>>a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
>>writefis(a,'my_file')
```

结果为 ans =
 my_file

6.2.20 显示 FIS 的规则

函数 showrule

格式 showrule(fis)
showrule(fis,indexList)
showrule(fis,indexList,format)
showrule(fis,indexList,format,Lang)

说明 此命令用于显示与给定系统相关的规则。

fis 是必须提供的变量，这是一个 FIS 结构在 MATLAB 工作空间中的变量名；

indexList 是你要显示的规则向量（可选项）；

format 是一个表示返回规则格式的字符串（可选项），showrule 可以用三种不同格式的任意一种返回规则：'verbose'（缺省模式，此处 English 是缺省语言），'symbolic'和'indexed'，它们用于隶属度函数的索引引用；

若要使用第四个参数 Lang，则 Lang 必须是 verbose（语言）型的，并且下面这种调用 showrule(fis,indexList,format,Lang)使用 Lang 给定的语言显示规则，它们必须是'english'，'français'或'deutsch'。

例 6-35

```
>>a = readfis('tipper');
>>showrule(a,1)
ans =
    1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
>>showrule(a,2)
ans =
    2. If (service is good) then (tip is average) (1)
```



```
>>showrule(a,[3 1],'symbolic')
ans =
    3. (service==excellent) | (food==delicious) => (tip=generous) (1)
    1. (service==poor) | (food==rancid) => (tip=cheap) (1)
>>showrule(a,1:3,'indexed')
ans =
    1 1, 1 (1) : 2
    2 0, 2 (1) : 1
    3 2, 3 (1) : 2
```

6.2.21 显示 FIS 结构的所有属性

函数 **showfis**

格式 **showfis(fismat)**

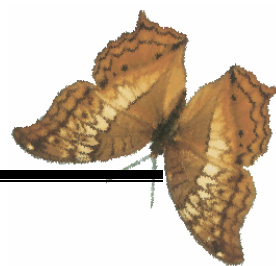
说明 以分行方式显示 MATLAB 工作空间 FIS 变量 **fismat**，允许你查看结构的每个域的意义和内容。

例 6-36

```
>>a = readfis('tipper');
>>showfis(a)
```

结果为

1. Name	tipper
2. Type	mamdani
3. Inputs/Outputs	[2 1]
4. NumInputMFs	[3 2]
5. NumOutputMFs	3
6. NumRules	3
7. AndMethod	min
8. OrMethod	max
9. ImpMethod	min
10. AggMethod	max
11. DefuzzMethod	centroid
12. InLabels	service
13.	food
14. OutLabels	tip
15. InRange	[0 10]
16.	[0 10]
17. OutRange	[0 30]
18. InMFLabels	poor
19.	good
20.	excellent
21.	rancid
22.	delicious
23. OutMFLabels	cheap
24.	average
25.	generous
26. InMFTypes	gaussmf
27.	gaussmf
28.	gaussmf
29.	trapmf
30.	trapmf
31. OutMFTypes	trimf
32.	trimf
33.	trimf



34. InMFParams	[1.5 0 0 0]
35.	[1.5 5 0 0]
36.	[1.5 10 0 0]
37.	[0 0 1 3]
38.	[7 9 10 10]
39. OutMFParams	[0 5 10 0]
40.	[10 15 20 0]
41.	[20 25 30 0]
42. Rule Antecedent	[1 1]
43.	[2 0]
44.	[3 2]
42. Rule Consequent	1
43.	2
44.	3
42. Rule Weigth	1
43.	1
44.	1
42. Rule Connection	2
43.	1
44.	2



第7章 绘图与图形处理

人们很难从一大堆原始的数据中发现它们的含义，而数据图形恰能使视觉感官直接感受到数据的许多内在本质，发现数据的内在联系。MATLAB 可以表达出数据的二维，三维，甚至四维的图形。通过图形的线型，立面，色彩，光线，视角等属性的控制，可把数据的内在特征表现得淋漓尽致。下面我们分别介绍图形的命令。

7.1 二维图形

7.1.1 基本平面图形命令

命令 1 plot

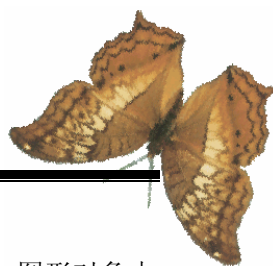
功能 线性二维图。在线条多于一条时，若用户没有指定使用颜色，则 plot 循环使用由当前坐标轴颜色顺序属性（current axes ColorOrder property）定义的颜色，以区别不同的线条。在用完上述属性值后，plot 又循环使用由坐标轴线型顺序属性（axes LineStyleOrder property）定义的线型，以区别不同的线条。

用法 plot(X,Y) 当 X,Y 均为实数向量，且为同维向量（可以不是同型向量）， $X=[x(i)]$ ， $Y=[y(i)]$ ，则 plot(X,Y) 先描出点 $(x(i), y(i))$ ，然后用直线依次相连；若 X, Y 为复数向量，则不考虑虚数部分。若 X, Y 均为同维同型实数矩阵， $X=[X(i)]$ ， $Y=[Y(i)]$ ，其中 $X(i), Y(i)$ 为列向量，则 plot(X,Y) 依次画出 plot(X(i),Y(i))，矩阵有几列就有几条线；若 X, Y 中一个为向量，另一个为矩阵，且向量的维数等于矩阵的行数或者列数，则矩阵按向量的方向分解成几个向量，再与向量配对分别画出，矩阵可分解成几个向量就有几条线；在上述的几种使用形式中，若有复数出现，则复数的虚数部分将不被考虑。

plot(Y) 若 Y 为实数向量，Y 的维数为 m，则 plot(Y) 等价于 plot(X,Y)，其中 $x=1:m$ ；若 y 为实数矩阵，则把 y 按列的方向分解成几个列向量，而 y 的行数为 n，则 plot(Y) 等价于 plot(X,Y) 其中 $x=[1;2;\dots;n]$ ；在上述的几种使用形式中，若有复数出现，则复数的虚数部分将不被考虑。

plot(X1,Y1,X2,Y2,...)，其中 X_i 与 Y_i 成对出现，plot(X1,Y1,X2,Y2,...) 将分别按顺序取两数据 X_i 与 Y_i 进行画图。若其中仅仅有 X_i 或 Y_i 是矩阵，其余的为向量，向量维数与矩阵的维数匹配，则按匹配的方向来分解矩阵，再分别将配对的向量画出。

plot(X1,Y1,LineStyle1,X2,Y2,LineStyle2...) 将按顺序分别画出由三参数定义 $X_i, Y_i, LineSpec_i$ 的线条。其中参数 LineSpec_i 指明了线条的类型，标记符号，和画线用的颜色。在 plot 命令中我们可以混合使用三参数和二参数的形式：



`plot(X1,Y1,LineStyle1,X2,Y2,X3,Y3,LineStyle3)`

`plot(...,'PropertyName',PropertyValue,...)` 对所有的用 `plot` 生成的 line 图形对象中指定的属性进行恰当的设置。

`h = plot(...)` 返回 line 图形对象句柄的一列向量，一线条对应一句柄值。

说明 参数 `LineStyle`

功能 定义线的属性。Matlab 允许用户对线条定义如下的特性：

1. 线型

表 7-1

定义符	-	--	:	-.
线型	实线（缺省值）	划线	点线	点划线

2. 线条宽度

指定线条的宽度，取值为整数（单位为像素点）

3. 颜色

表 7-2

定义符	R (red)	G(green)	b(blue)	c(cyan)
颜色	红色	绿色	蓝色	青色
定义符	M(magenta)	y(yellow)	k(black)	w(white)
颜色	品红	黄色	黑色	白色

4. 标记类型

表 7-3

定义符	+	o(字母)	*	.	x
标记类型	加号	小圆圈	星号	实点	交叉号
定义符	d	^	v	>	<
标记类型	菱形	向上三角形	向下三角形	向右三角形	向左三角形
定义符	s	h	P		
标记类型	正方形	正六角星	正五角星		

5. 标记大小

指定标记符号的大小尺寸，取值为整数（单位为像素）

6. 标记面填充颜色

指定用于填充标记符面的颜色。取值在上表。

7. 标记周边颜色

指定标记符颜色或者是标记符（小圆圈、正方形、菱形、正五角星、正六角星和四个方向的三角形）周边线条的颜色。取值在上表。

在所有的能产生线条的命令中，参数 `LineStyle` 可以定义线条的下面三个属性：线型、标记符号、颜色进行设置。对线条的上述属性的定义可用字符串来定义，如：`plot(x,y,'-or')`

结合 `x` 和 `y`，画出点划线（-），在数据点（`x`，`y`）处画出小圆圈（`o`），线和标记都用红色画出。其中定义符（即字符串）中的字母、符号可任意组合。若没有定义符，则画图命令 `plot` 自动用缺省值进行画图。若仅仅指定了标记符，而非线型，则 `plot` 只在数据点画出标记符。如：`plot(x,y,'d')`

例 7-1



```

>>t = 0:pi/20:2*pi;
>>plot(t,t.*cos(t),'-r*')
>>hold on
>>plot(exp(t/100).*sin(t-pi/2),'--mo')
>>plot(sin(t-pi),'bs')
>>hold off

```

图形结果为图 7-1。

例 7-2

```

>>plot(t,sin(2*t),'-mo', 'LineWidth',2,'MarkerEdgeColor','k',...
'MarkerFaceColor',[.49 1 .63],'MarkerSize',12)

```

图形结果为图 7-2。

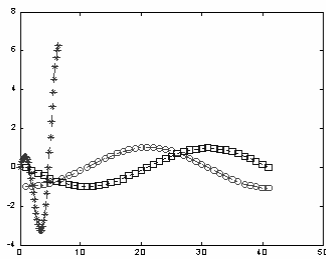


图 7-1 二维曲线图

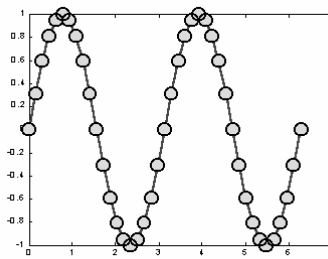


图 7-2 二维图形的绘制

命令 2 fplot

功能 在指定的范围 **limits** 内画出一元函数 $y=f(x)$ 的图形。其中向量 **x** 的分量分布在指定的范围内，**y** 是与 **x** 同型的向量，对应的分量有函数关系： $y(i)=f(x(i))$ 。若对应于 **x** 的值，**y** 返回多个值，则 **y** 是一个矩阵，其中每列对应一个 $f(x)$ 。例如， $f(x)$ 返回向量 $[f_1(x), f_2(x), f_3(x)]$ ，输入参数 $x=[x_1; x_2; x_3]$ ，则函数 $f(x)$ 返回矩阵

```

f1(x1)  f2(x1)  f3(x1)
f1(x2)  f2(x2)  f3(x2)
f1(x3)  f2(x3)  f3(x3)

```

注意一点的是，函数 **function** 必须是一个 **m**-文件函数或者是一个包含变量 **x**，且能用函数 **eval** 计算的字符串。例如：' $\sin(x)*\exp(2*x)$ '，' $[\sin(x), \cos(x)]$ '，'**hump(x)**'。

用法 **fplot('function',limits)** 在指定的范围 **limits** 内画出函数名为 **function** 的一元函数图形。其中 **limits** 是一个指定 **x**-轴范围的向量 $[x_{min} \ x_{max}]$ 或者是 **x** 轴和 **y** 轴的范围的向量 $[x_{min} \ x_{max} \ y_{min} \ y_{max}]$ 。

fplot('function',limits,LineSpec) 用指定的线型 **LineSpec** 画出函数 **function**。

fplot('function',limits,tol) 用相对误差值为 **tol** 画出函数 **function**。相对误差的缺省值为 $2e-3$ 。

fplot('function',limits,tol,LineSpec) 用指定的相对误差值 **tol** 和指定的线型 **LineSpec** 画出函数 **function** 的图形。

fplot('function',limits,n) 当 $n \geq 1$ ，则至少画出 $n+1$ 个点（即至少把范围 **limits** 分成 **n** 个小区间），最大步长不超过 $(x_{max}-x_{min})/n$ 。



`fplot('function',limits,...)` 允许可选参数 `tol`, `n` 和 `LineSpec` 以任意组合方式输入。
`[X,Y] = fplot('function',limits,...)` 返回横坐标与纵坐标的值给变量 `X` 和 `Y`, 此时 `fplot` 不画出图形。若想画出, 可用命令 `plot(X,Y)`。

`[...] = plot('function',limits,tol,n,LineSpec,P1,P2,...)` 允许用户直接给函数 `function` 输入参数 `P1`, `P2` 等, 其中函数 `functiond` 的定义形式为

$$y = \text{function}(x,P1,P2,\dots)$$

若想用缺省的 `tol`, `n` 或 `LineSpec` 值, 只需将空矩阵 `[]` 传递给函数即可。

注意: `fplot` 采用自适应步长控制来画出函数 `function` 的示意图, 在函数的变化激烈的区间, 采用小的步长, 否则采用大的步长。总之, 使计算量与时间最小, 图形尽可能精确。

例 7-3

```
>>fplot('tanh',[-2 2])
```

图形结果为图 7-3。

```
>>subplot(2,2,1);fplot('humps',[0 1])
>>subplot(2,2,2);fplot('abs(exp(-j*x*(0:9)))*ones(10,1)',[0 2*pi])
>>subplot(2,1,2);fplot('tan(x),sin(x),cos(x)',['2*pi*[-1 1 -1 1]'])
```

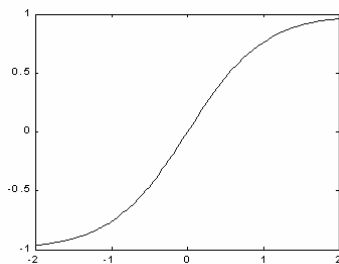


图 7-3 函数画图

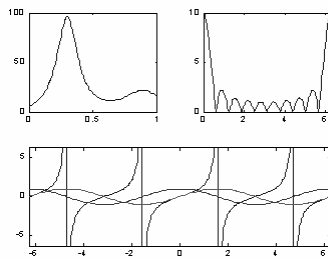


图 7-4

命令 3 loglog

功能 双对数图形。

用法 `loglog(Y)` 若 `y` 为实数向量或矩阵, 则结合 `y` 列向量的下标与 `y` 的列向量画出。
 若 `y` 为复数向量或矩阵, 则 `loglog(Y)` 等价于 `loglog(real(Y),imag(Y))`, 在 `loglog` 的其他使用形式中将忽略 `Y` 的虚数部分。

`loglog(X1,Y1,X2,Y2,...)` 结合 `Xn` 与 `Yn` 画出图形。若只有 `Xn` 或 `Yn` 为矩阵, 另一个为向量, 行向量维数等于矩阵的列数, 列向量的维数等于矩阵的行数, 则 `loglog` 把矩阵按向量的方向分解成向量, 再与向量结合分别画出图形。

`loglog(X1,Y1,LineSpec1,X2,Y2,LineSpec2,...)` 按顺序取三个参数 `Xn`, `Yn`, `LineSpecn` 画出线条, 其中 `LineSpecn` 指定线条的线型, 标记符号和颜色。
 用户可以混合使用二参数和三参数形式, 如:

$$\text{loglog}(X1,Y1,X2,Y2,\text{LineSpec2},X3,Y3)$$

`loglog(...,'PropertyName',PropertyValue,...)` 对所有由 `loglog` 命令生成的图形对象句柄的属性进行设置。



$h = \text{loglog}(\cdots)$ 返回 line 图形句柄向量，每条线对应一个句柄。

例 7-4

```
>>x = logspace(-1,2);
>>loglog(x,10*exp(x),'-s')
>>grid on
```

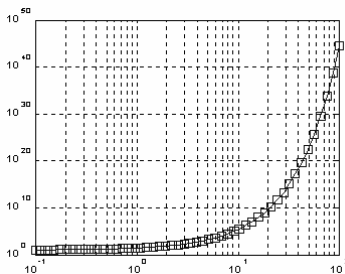


图 7-5

命令 4 semilogx

功能 x 轴对数图形。若没有指定使用的颜色，当所画线条较多时，semilogx 将自动使用由当前轴的 ColorOrder 和 LineStyleOrder 属性指定的颜色顺序和线型顺序来画线。

用法 $\text{semilogx}(Y)$ %对 x 轴的刻度求常用对数（以 10 为底），而 y 轴为线性刻度。

若 y 为实数向量或矩阵，则结合 y 列向量的下标与 y 的列向量画出线条；若 y 为复数向量或矩阵，则 $\text{semilogx}(Y)$ 等价于 $\text{semilogx}(\text{real}(Y), \text{imag}(Y))$ 。在 semilogx 的其他使用形式中，Y 的虚数部分将被忽略。

$\text{semilogx}(X1, Y1, X2, Y2, \cdots)$ %结合 X_n 和 Y_n 画出线条，若其中只有 x_n 或 y_n 为矩阵，另外一个为向量，行向量的维数等于矩阵的列数，列向量的维数等于矩阵的行数，则按向量的方向分解矩阵，再与向量结合，分别画出线条。

$\text{semilogx}(X1, Y1, \text{LineStyle1}, X2, Y2, \text{LineStyle2}, \cdots)$ %按顺序取三参数 $X_n, Y_n, \text{LineSpec}_n$ 画线，参数 LineSpec_n 指定使用的线型，标记符号和颜色。

用户可以混合使用二参数和三参数形式，如：

$\text{semilogx}(X1, Y1, X2, Y2, \text{LineStyle2}, X3, Y3)$

$\text{semilogx}(\cdots, \text{'PropertyName'}, \text{PropertyValue}, \cdots)$ %对所有由 semilogx 命令生成的图形对象句柄的属性进行设置

$h = \text{semilogx}(\cdots)$ %返回 line 图形句柄向量，每条线对应一个句柄。

例 7-5

```
>>x = 0:1:10;
>>semilogx(x,cos(10.^x))
```

图形结果为图 7-6。

命令 5 semilogy

用法：参见 semilogx 命令。

命令 6 fill

功能 用颜色填充二维多边形。

用法 $\text{fill}(X, Y, C)$ 用 x 和 y 中的数据生成多边形，

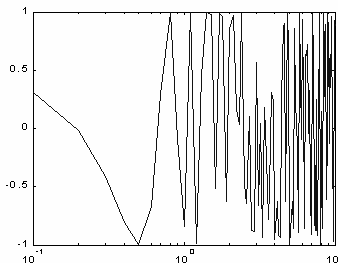
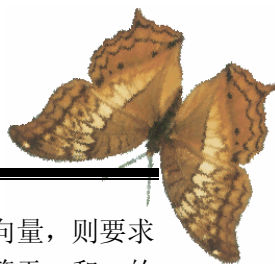


图 7-6



用 c 指定的颜色填充它。其中 c 为色图向量或矩阵。若 c 是行向量，则要求 c 的维数等于 x 和 y 的列数，若 c 为列向量，则要求 c 的维数等于 x 和 y 的行数。

`fill(X,Y,ColorSpec)` 用 `ColorSpec` 指定的颜色填充由 x 和 y 定义的多边形

`fill(X1,Y1,C1,X2,Y2,C2,...)` 指定多个要填充的二维区域

`fill(...,'PropertyName',PropertyValue)` 允许用户对一个 `patch` 图形对象的某个属性设定属性值。

`h = fill(...)` 返回 `patch` 图形对象句柄的向量，每一个 `patch` 对象对应一个句柄。

注意：

1. 若 x 或 y 是一矩阵，另一个是向量，向量应是维数与矩阵的行数相等的列向量或是维数等于矩阵列数的行向量时，函数 `fill` 将向量复制成与矩阵同型的矩阵。函数 `fill` 将矩阵 x 与 y 中列向量中的数据生成多边形的顶点。

2. 颜色阴影类型决定于用户在参数中列出的颜色，若用户用 `ColorSpec` 指定颜色，命令 `fill` 生成平坦阴影模式 (`flat-shaded`) 多边形，同时设置补片对象 (`patch`) 的 `FaceColor` 属性为相应的 **RGB** 颜色矩阵。

3. 若用户用参量 c 指定所用颜色，命令 `fill` 按坐标轴属性 `Clim` 的比例缩小 c 中的元素，之后， c 成为引用当前色图的下标矩阵。

4. 若 c 为行向量，命令 `fill` 生成平面阴影的多边形， c 的每一元素决定由矩阵 x , y 的每一列定义的多边形内的颜色，每一补片对象的 `FaceColor` 属性被设置为 `'flat'`， x , y 的每一行元素变成第 n 块补片对象的 `Cdata` 属性值，其中 n 为矩阵 x 或 y 中的相应的列。

5. 若 c 为一列向量或一矩阵，命令 `fill` 运用一线性插值法计算每一节点的颜色，以便用插值颜色填充多边形的内部。它设置补片对象的 `FaceColor` 属性为 `'interp'`，且在一列中的元素变成每一补片的 `Cdata` 属性值。若 c 为一列向量，命令 `fill` 用该向量复制成需要大小的尺寸。

例 7-6

```
>>t = (1/16:1/8:1)*2*pi;
>>x = exp(t).*sin(t);
>>y = t.*cos(t);
>>fill(x,y,'k')
>>grid on
```

图形结果为图 7-7。

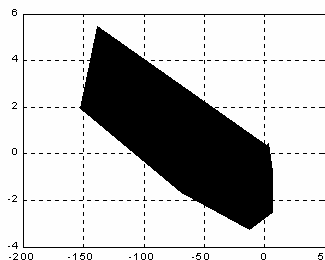


图 7-7

命令 7 zoom

功能 对二维图形进行放大或缩小。放大或缩小会改变坐标轴范围。

用法 `zoom on` 打开交互式的放大功能。当一个图形处于交互式的放大状态时，有两种方法来放大图形：

对于一键鼠标或二键，三键鼠标，单击坐标轴内的任意一点，可使图形放大一倍，这一操作可进行多次，直到 `matlab` 的最大显示为止；对于二键或三键的鼠标，在坐标轴内单击右键，可使图形缩小一倍，这一操作可进行多次，直到还原图形为止。对于一键鼠标，要想缩小图形，需要按住键盘上的 `Shift` 键，再单击鼠标键。

用鼠标拖出要放大的部分，系统将放大选定的区域。



zoom off 关闭交互式放大功能。

zoom out 将系统转回非放大状态，并将图形恢复原状。

zoom reset 系统将记住当前图形的放大状态，作为放大状态的设置值。以后使用 **zoom out** 或者是双击鼠标时，交互式放大状态打开，且图形并不是返回到原状，而是返回 **reset** 时的放大状态。

zoom 用于切换放大的状态：**on** 和 **off**。

zoom xon 只对 **x** 轴进行放大。

zoom yon 只对 **y** 轴进行放大。

zoom(factor) 用放大系数 **factor** 进行放大或缩小，而不影响交互式放大的状态。

若 $\text{factor} > 1$ ，系统将图形放大 **factor** 倍，若 $0 < \text{factor} \leq 1$ ，系统将图形放大 $1/\text{factor}$ 倍。

zoom(fig, option) 指定对窗口 **fig** 中（不一定为当前窗口）的二维图形进行放大，其中参数 **option** 为：**on**、**off**、**xon**、**yon**、**reset**、**factor** 等。

命令 8 meshgrid

功能 生成二元函数 $z = f(x,y)$ 中 **x-y** 平面上的矩形定义域中数据点矩阵 **X** 和 **Y**，或者是三元函数 $u = f(x,y,z)$ 中立方体定义域中的数据点矩阵 **X**、**Y** 和 **Z**。

用法 a: **[X,Y] = meshgrid(x,y)**

b: **[X,Y] = meshgrid(x)**

c: **[X,Y,Z] = meshgrid(x,y,z)**

说明 对于形式 **a**，输入向量 **x** 为 **x-y** 平面上矩形定义域的矩形分割线在 **x** 轴的值，向量 **y** 为 **x-y** 平面上矩形定义域的矩形分割线在 **y** 轴的值。输出向量 **X** 为 **x-y** 平面上矩形定义域的矩形分割点的横坐标值矩阵，输出向量 **Y** 为 **x-y** 平面上矩形定义域的矩形分割点的纵坐标值矩阵。

对于形式 **b**，等价于形式 **a**: **[X,Y] = meshgrid(x) = meshgrid(x,x)**。

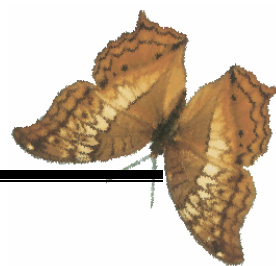
对于形式 **c**，输入向量 **x** 为立方体定义域的立方体分割平面在 **x** 轴上的值，输入向量 **y** 为立方体定义域的立方体分割平面在 **y** 轴上的值，输入向量 **z** 为立方体定义域的立方体分割平面在 **z** 轴上的值。输出向量 **X** 为立方体定义域中分割点的 **x** 轴坐标值，**Y** 为立方体定义域中分割点的 **y** 轴坐标值，**Z** 为立方体定义域中分割点的 **z** 轴坐标值。

例 7-7

```
>>x = [0.7 1.1]; y = [-2 3 1]; z = [2 5 3]; %分量不一定从小到大
>>[X_2d,Y_2d] = meshgrid(x,y)
>>[X_3d,Y_3d,Z_3d] = meshgrid(x,y,z)
```

计算结果为：

```
X_2d =
    0.7000    1.1000
    0.7000    1.1000
    0.7000    1.1000
Y_2d =
    -2    -2
     3     3
     1     1
X_3d(:,:,1) =
    0.7000    1.1000
```



```

0.7000    1.1000
0.7000    1.1000
X_3d(:,:,2) =
0.7000    1.1000
0.7000    1.1000
0.7000    1.1000
X_3d(:,:,3) =
0.7000    1.1000
0.7000    1.1000
0.7000    1.1000
Y_3d(:,:,1) =
-2    -2
3     3
1     1
Y_3d(:,:,2) =
-2    -2
3     3
1     1
Y_3d(:,:,3) =
-2    -2
3     3
1     1
Z_3d(:,:,1) =
2     2
2     2
2     2
Z_3d(:,:,2) =
5     5
5     5
5     5
Z_3d(:,:,3) =
3     3
3     3
3     3

```

7.1.2 特殊平面图形命令

命令 1 polar

功能 画极坐标图。该命令接受极坐标形式的函数 $\rho=f(\theta)$ ，在笛卡儿坐标系平面上画出该函数，且在平面上画出极坐标形式的格栅。

用法 `polar(theta,rho)` 用极角 θ 和极径 ρ 画出极坐标图形。极角 θ 为从 x 轴到半径的单位为弧度的向量，极径 ρ 为各数据点到极点的半径向量。

`polar(theta,rho,LineSpec)` 参量 `LineSpec` 指定极坐标图中线条的线型、标记符号和颜色等。

例 7-8

```

>>t = 0:0.01:2*pi;
>>polar(t,sin(3*t).*cos(2*t),'-r')

```

图形结果为图 7-8。

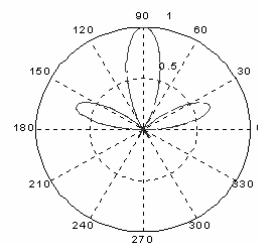


图 7-8

命令 2 bar

功能 二维垂直条形图。用垂直条形显示向量或矩阵中的值。

用法 `bar(Y)` 若 y 为向量，则分别显示每个分量的高度，横坐标为 1 到 `length(y)`；若 y 为矩阵，则 `bar` 把 y 分解成行向量，再分别画出，横坐标为 1 到 `size(y,1)`，即矩阵的行数。



`bar(x,Y)` 在指定的横坐标 x 上画出 y , 其中 x 为严格单增的向量。若 y 为矩阵, 则 `bar` 把矩阵分解成几个行向量, 在指定的横坐标处分别画出。

`bar(...,width)` 设置条形的相对宽度和控制在一组内条形的间距。缺省值为 0.8, 所以, 如果用户没有指定 x , 则同一组内的条形有很小的间距, 若设置 `width` 为 1, 则同一组内的条形相互接触。

`bar(...,'style')` 指定条形的排列类型。类型有 “group” 和 “stack”, 其中 “group” 为缺省的显示模式。

“group”: 若 y 为 $n*m$ 阶的矩阵, 则 `bar` 显示 n 组, 每组有 m 个垂直条形的条形图。

“stack”: 对矩阵 y 的每一个行向量显示在一个条形中, 条形的高度为该行向量中的分量和。其中同一条形中的每个分量用不同的颜色显示出来, 从而可以显示每个分量在向量中的分布。

`bar(...,LineStyle)` 用指定的颜色 `LineStyle` 显示所有的条形。

`[xb,yb] = bar(...)` 返回用户可用命令 `plot` 或命令 `patch` 画出条形图的参量 xb , yb 。这对用户控制一个图形的显示是有用的, 例如要在一个 `plot` 语句中加入装饰性的条形图等。

`h = bar(...)` 返回一个 `patch` 图形对象句柄的向量。每一条形对应一个句柄。

例 7-9

```
x = -2.9:0.2:2.9;  
bar(x,exp(x.*sin(x)))  
colormap gray
```

图形结果为图 7-9。

例 7-10

```
subplot(2,2,4)  
bar(Y,1.5)  
title 'Width = 1.5'
```

图形结果为图 7-10。

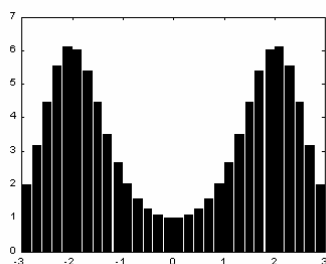


图 7-9

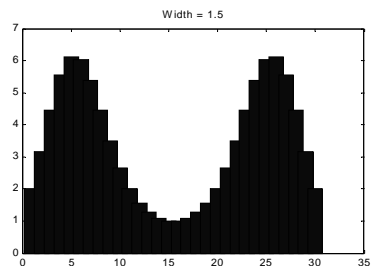


图 7-10

命令 3 barh

功能 二维水平条形图。用水平条形显示向量或矩阵中的值。

用法 `barh(Y)` 若 y 为向量, 则分别显示每个分量的高度, 纵坐标为 1 到 `length(y)`; 若 y 为矩阵, 则 `barh` 把 y 分解成行向量, 再分别画出, 纵坐标为 1 到 `size(y,1)`, 即矩阵的行数。

`barh(x,Y)` 在指定的纵坐标 x 上以水平方向画出 y , 其中 x 为严格单增的向量。若 y 为矩阵, 则 `barh` 把矩阵分解成几个行向量, 在指定的纵坐标处分别画出。

`barh(...,width)` 设置条形的相对宽度和控制在一组内条形的间距。缺省值为 0.8, 所以, 如果用户没有指定 x , 则同一组内的条形有很小的间距, 若设置 `width` 为 1, 则同一组内的条形相互接触。

`barh(...,'style')` 指定条形的排列类型。类型有 “group” 和 “stack”, 其中 “group” 为缺省的显示模式。

“group”: 若 y 为 $n*m$ 阶的矩阵, 则 `barh` 显示 n 组, 每组有 m 个水平条形的条形图。

“stack”: 对矩阵 y 的每一个行向量显示在一个条形中, 条形的高度为



该行向量中的分量和。其中同一条形中的每个分量用不同的颜色显示出来，从而可以显示每个分量在向量中的分布。

`barh(...,LineSpec)` 用指定的颜色 `LineSpec` 显示所有的条形。

`[xb,yb] = barh(...)` 返回用户可用命令 `plot` 或命令 `patch` 画出条形图的参量 `xb,yb`。

这给用户控制一个图形的显示是有用的，例如要在一个 `plot` 语句中加入装饰性的条形图等。

`h = barh(...)` 返回一个 `patch` 图形对象句柄的向量。每一条形对应一个句柄。

例 7-11

```
>>X = 1:5:5;
>>Y = exp(X).*sin(X);
>>barh(Y,'stack')
```

图形结果为图 7-11。

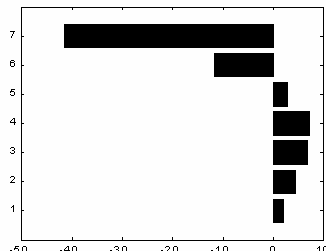


图 7-11

命令 4 compass

功能 从原点画箭头图。箭头图为一显示起点为笛卡儿坐标系中的原点的二维或三维方向或向量的图形，同时在坐标系中显示圆形的分隔线。

用法 `compass(X,Y)` 参量 `x` 与 `y` 为同型的 `n` 维向量，则命令显示 `n` 个箭头，箭头的起点为原点，箭头的位置为 `[X(i),Y(i)]`。

`compass(Z)` 参量 `z` 为 `n` 维复数向量，则命令显示 `n` 个箭头，箭头起点为原点，箭头的位置为 `[real(Z),imag(Z)]`。

`compass(...,LineSpec)` 用参量 `LineSpec` 指定箭头图的线型、标记符号、颜色等属性。

`h = compass(...)` 返回 `line` 对象的句柄给 `h`。

例 7-12

```
Z = magic(20).*randn(20);
compass(Z)
```

图形结果为图 7-12。

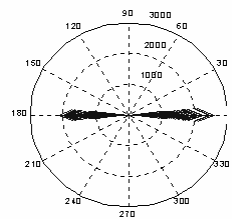


图 7-12

命令 5 comet

功能 二维彗星图。彗星图为彗星头（一个小圆圈）沿着数据点前进的动画，彗星体为跟在彗星头后面的痕迹，轨道为沿着整个函数的实线。我们要指出的是，由命令 `comet` 生成的轨迹是使用擦除模式（`EraseMode`）属性的值为 `none`，该属性使用户不能打印该图形（只能得到彗星头），且当用户改变窗口的大小时，动画将消失。

用法 `comet(y)` 彗星图动画显示向量 `y` 确定的路线。

`comet(x,y)` 彗星图动画显示向量 `x` 与 `y` 确定的路线。

`comet(x,y,p)` 指定彗星体的长度 `p*length(y)`，缺省的 `p` 值为 0.1。

例 7-13

```
>>t = 0:0.01:2*pi;
>>x = exp(sin(2*t)).*(cos(t).^2/3);
>>y = t.*(sin(t).^2);
>>comet(x,y);
```

图形结果为图 7-13。

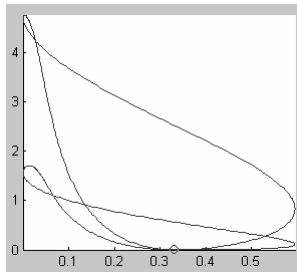


图 7-13



附：擦除模式(EraseMode)属性及属性值：{normal}|none|xor|background

该属性控制系统用于显示与擦除线条对象的技术。不同的擦除模式对于生成动画系列，即控制个别对象的重新显示方式，对于改进外在显示和获得理想的效果是很必要的。

表 7-4

属性值	含义
Normal (缺省值)	重新显示受影响的区域，在必要的时候，进行三维分析计算，以保证所有的对象的显示都是正确的。该模式下的图形显示是最精确的，不过也是最缓慢的，以下其他三种模式显示速度较快，不过没有执行一个完全的重显过程，因而，图形显示也不是很精确的。
none	当线条移动或改动时，该模式没有擦除线条，而是仍然显示于屏幕上。该模式下不能打印图形，因为系统没有存储前一图形的任何信息。
xor	使用异或运算(xor)计算线条颜色与当前位置下的颜色，用所得结果显示与擦除线条。该模式对于线条下面对象的颜色没有任何破坏，只是影响到线条的当前显示颜色而已。
Background	用当前坐标轴颜色重新显示线条的方式来擦除线条，若当前坐标轴颜色设置为 none，则用图形的背景色来代替坐标轴颜色。该模式对于处于擦除线条后面的对象来说是有损害的，不过当前线条的颜色总是最合适的。

命令 6 errorbar

功能 沿着一曲线画误差棒形图。误差棒为数据的置信水平或者为沿着曲线的偏差。在下列参数中，若为矩阵，则按列画出误差棒。

用法 errorbar(Y,E) 画出向量 y，同时显示在向量 y 的每一元素之上的误差棒。误差棒为 E(i)在曲线 y 上面与下面的距离，所以误差棒的长度为 2*E(i)。

errorbar(X,Y,E) X,Y,E 必须为同型参量。若同为向量，则画出带长度为 2*E(i)、对称误差棒于曲线点(X(i),Y(i))之处；若同为矩阵，则画出带长度为 E(i, j)、对称误差棒于曲面点(X(i,j),Y(i,j))之处，

errorbar(X,Y,L,U) X, Y, L, U 必须为同型参量。若同为向量，则在点(X(i),Y(i))处画出向下长为 L(i)，向上长为 U(i)的误差棒；若同为矩阵，则在点(X(i,j),Y(i,j))处画出向下长为 L(i,j),向上长为 U(i,j)的误差棒。

errorbar(...,LineStyle) 用 LineSpec 指定的线型、标记符、颜色等画出误差棒。

h = errorbar(...) 返回线图图形对象的句柄向量给 h。

例 7-14

```
>>X = 0:pi/10:pi;  
>>Y = exp(X).*sin(X);  
>>E = std(Y)*ones(size(X));  
>>errorbar(X,Y,E)
```

图形结果为图 7-14。

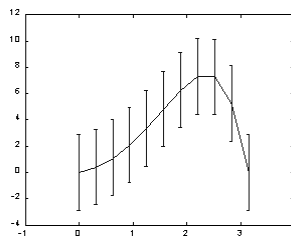


图 7-14

命令 7 feather

功能 画出速度向量图。一羽毛图在横坐标上等距地显示向量。用户要表示各个向量的、相对于原点的向量分量。

用法 feather(U,V) 显示由参量向量 u 与 v 确定的向量，其中 u 包含作为相对坐标系中的 x 成分，v 包含作为相对坐标系中的 y 成分。

feather(Z) 显示复数参量向量 z 确定的向量，等价于 feather(real(Z),imag(Z))。



`feather(...,LineStyle)` 用参量 `LineStyle` 指定的线型、标记符号、颜色等属性画出羽毛图。

例 7-15

```
>>th = (-90:10:90)*pi/180;
>>r = 4*ones(size(th));
>>[u,v] = pol2cart(th,r);
>>feather(u,v);
```

图形结果为图 7-15。

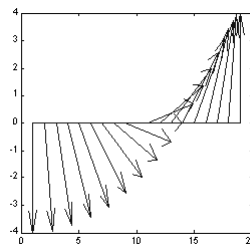


图 7-15

命令 8 hist

功能 二维条形直方图，可以显示出数据的分配情形。所有向量 y 中的元素或者是矩阵 y 中的列向量中的元素是根据它们的数值范围来分组的，每一组作为一个条形进行显示。条形直方图中的 x 轴反映了数据 y 中元素数值的范围，直方图的 y 轴显示出参量 y 中的元素落入该组的数目。所以 y 轴的范围从 0 到任一条形中包含元素最多的数字。直方图为一 `patch` 图形对象，若想改变图形的颜色，可以对 `patch` 对象的属性进行设置。缺省时，图形颜色是由当前色图进行控制，当前色图的第一个颜色为直方图的颜色。

用法 `n = hist(Y)` 把向量 y 中的元素放入等距的 10 个条形中，且返回每一个条形中的元素个数。若 y 为矩阵，则该命令按列对 y 进行处理。

`n = hist(Y,x)` 参量 x 为向量，把 y 中元素放到 m ($m=\text{length}(x)$) 个由 x 中元素指定的位置为中心的条形中。

`n = hist(Y,nbins)` 参量 `nbins` 为标量，用于指定条形的数目。

`[n,xout] = hist(...)` 返回向量 n 与包含频率计数与条形的位置向量 `xout`，用户可以用命令 `bar(xout,n)` 画出条形直方图。

例 7-16

```
>>x = -5:0.1:5;
>>y = randn(1000,1);
>>hist(y,x)
```

图形结果为图 7-16。

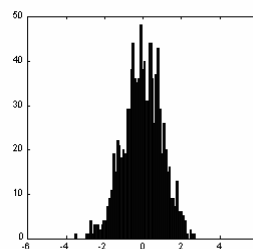


图 7-16

命令 9 histc

功能 直方图记数

用法 `n = histc(x,edges)` 统计向量 x 中、落入向量 `edges` (元

素必须为单调的非减的) 各个元素之间的元素个数。输出参量 n 为一与向量 `edges` 同维的向量。其中若有 `edges(k) >= x(i) >= edges(k+1)`，则 `n(k)` 增加 1。X 中超出向量 `edges` 规定的范围的元素将不被统计。参量 `edges` 中可使用 `-inf` 与 `inf`，用于包括向量 x 中非 NaN 的元素。若 x 为一矩阵，则对 x 的每一列进行上述操作。

`n = histc(x,edges,dim)` 对多维矩阵的第 `dim` 维进行统计。

`[n,bin] = histc(...)` n 结果同上，同时返回矩阵下标 `bin`。若 x 为向量，`n(k) = sum(bin == k)`。对于超出范围的数值，`bin` 为零值。

命令 10 rose

功能 画角度直方图。该直方图是一个显示所给数据的变化范围内数据的分布情形的极坐标图，所给数据分成不同的组。每一组作为一小扇形进行显示。



用法 `rose(theta)` 画一角度直方图，显示参数 `theta` 的数据在 20 个区间或更少的区间内的分布。向量 `theta` 中的角度单位为弧度，用于确定每一区间与原点角度。每一区间的长度反映出输入参量的元素落入一区间的个数。

`rose(theta,x)` 用参量 `x` 指定每一区间内的元素与区间的位置，`length(x)` 等于每一区间内元素的个数与每一区间位置角度的中间角度。例如，若 `x` 为一 5 维向量，`rose` 命令分配参量 `theta` 中的元素为 5 部分，每一部分的角度中线由 `x` 指定。

`rose(theta,nbins)` 于区间 $[0,2\pi]$ 内画出 `nbins` 个等距的小扇形。缺省值为 20。

`[tout,rout] = rose(...)` 返回向量 `tout` 与 `rout`，可以用 `polar(tout,rout)` 画出图形。该命令没有画任何的图形。

例 7-17

```
>>theta = 3*pi*randn(1,30);
>>rose(theta)
```

图形结果为图 7-17。

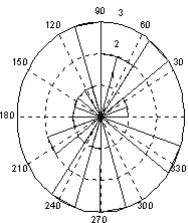


图 7-17

命令 11 stairs

功能 画二维阶梯图，这种图对与时间有关的数字样本系统的作图很有用处。

用法 `stairs(Y)` 用参量 `y` 的元素画一阶梯图。若 `y` 为向量，则横坐标 `x` 的范围从 1 到 `m=length(y)`，若 `y` 为矩阵，则对 `y` 的每一行画一阶梯图，其中 `x` 的范围从 1 到 `y` 的列数 `m`。

`stairs(X,Y)` 结合 `x` 与 `y` 画阶梯图。其中要求 `x` 与 `y` 为同型的向量或矩阵。此外，`x` 可以为行向量或为列向量，且 `y` 为有 `m=length(x)` 行的矩阵。

`stairs(...,LineStyle)` 用参数 `LineStyle` 指定的线型、标记符号和颜色画阶梯图。

`[xb,yb] = stairs(Y)` 该命令没有画图，而是返回可以用命令 `plot` 画出参量 `y` 的阶梯图的向量 `xb` 与 `yb`。

`[xb,yb] = stairs(X,Y)` 该命令没有画图，而是返回可以用命令 `plot` 画出参量 `x`，`y` 的阶梯图的向量 `xb` 与 `yb`。

例 7-18

```
>>x = 0:.25:10;
>>stairs(x,exp(sin(x.^2)))
```

图形结果为图 7-18。

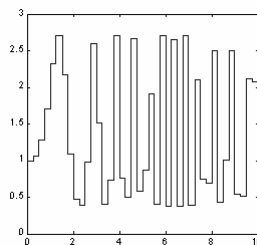


图 7-18

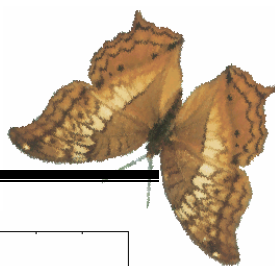
命令 12 stem

功能 画二维离散数据的柄形图。该图用线条显示数据点与 `x` 轴的距离，一小圆圈（缺省标记）或用指定的其他标记符号与线条相连，在 `y` 轴上标记数据点的值。

用法 `stem(Y)` 按 `y` 元素的顺序画出柄形图，在 `x` 轴上，柄与柄之间的距离相等；若 `y` 为矩阵，则把 `y` 分成几个行向量，在同一横坐标的位置上画出一个行向量的柄图。

`stem(X,Y)` 在横坐标 `x` 上画出列向量 `y` 的柄形图。其中 `x` 与 `y` 为同型的向量或矩阵，此外，`x` 可以为行向量或列向量，而 `y` 为有 `m=length(x)` 行的矩阵。

`stem(...,'fill')` 指定是否对柄形图末端的小圆圈填充颜色。



`stem(...,LineSpec)` 用参数 `LineSpec` 指定线型, 标记符号和柄图末端的小圆圈的颜色画柄图。

`h = stem(...)` 返回柄形图的 `line` 图形对象句柄向量。

例 7-19

```
>>x = linspace(0,2,10);
>>stem(exp(-x.^2),'fill','-o')
```

图形结果为图 7-19。

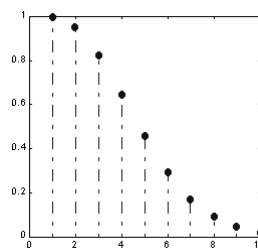


图 7-19

命令 13 stem3

功能 画三维离散数据的柄形图。该图用一线段显示数据离开 `xy` 平面的高度, 在线段的末端用一小圆圈 (缺省记号) 或其他的标记符号表示数据的高度。

格式 `stem3(Z)` 用柄形图显示 `z` 中数据与 `xy` 平面的高度。若 `z` 为一行向量, 则 `x` 与 `y` 将自动生成, `stem3` 将在与 `x` 轴平行的方向上等距的位置上画出 `z` 的元素; 若 `y` 为列向量, `stem3` 将在与 `y` 轴平行的方向上等距的位置上画出 `z` 的元素。
`stem3(X,Y,Z)` 在参数 `x` 与 `y` 指定的位置上画出 `z` 的元素, 其中 `x`, `y`, `z` 必须为同型的向量或矩阵。

`stem3(...,'fill')` 指定是否要填充柄形图末端小圆圈。

`stem3(...,LineSpec)` 指定线型, 标记符号和末端小圆圈的颜色。

`h = stem3(...)` 返回柄形图的 `line` 图形对象句柄。

例 7-20

```
[X,Y,Z] = peaks(20);
stem3(X,Y,Z,'r*')
```

图形结果为图 7-20。

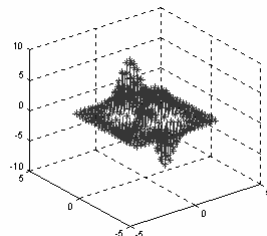


图 7-20

命令 14 pie

功能 饼形图

格式 `pie(X)` 用 `x` 中的数据画一饼形图, `x` 中的每一元素代表饼形图中的一部分。`X` 中元素 `X(i)` 所代表的扇形大小通过 `X(i)/sum(X)` 的大小来决定。若有 `sum(X)=1`, 则 `x` 中元素就直接指定了所在部分的大小; 若 `sum(X)<1`, 则画出一不完整的饼形图。

`pie(X,explode)` 从饼形图中分离出一部分, `explode` 为元素为零或非零的、与 `x` 相对应的向量或矩阵。与 `explode` 的非零值对应的部分将从饼形图中心分离出来。`explode` 必须与 `x` 同型。

`h = pie(...)` 返回一 `patch` 与 `text` 的图形对象句柄向量 `h`。

例 7-21

```
>>x = [1 3 0.5 2.5 2];
>>explode = [0 1 0 0 0];
>>pie(x,explode)
```

图形结果为图 7-21。

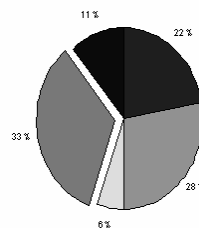


图 7-21

7.1.3 二维图形注释命令

命令 1 grid

功能 给二维或三维图形的坐标面增加分隔线。该命令会对当前坐标轴的 `Xgrid`, `Ygrid`,



Zgrid 的属性有影响。

用法 `grid on` 给当前的坐标轴增加分隔线。

`grid off` 从当前的坐标轴中去掉分隔线。

`grid` 转换分隔线的显示与否的状态。

`grid(axes_handle,on|off)` 对指定的坐标轴 `axes_handle` 是否显示分隔线。

命令 2 `gtext`

功能 在当前二维图形中用鼠标放置文字。当光标进入图形窗口时,会变成一个十字,表明系统正等待用户的动作。

用法 `gtext('string')` 当光标位于一个图形窗口内时,等待用户单击鼠标或键盘。若按下鼠标或键盘,则在光标的位置放置给定的文字“string”

`h = gtext('string')` 当用户在鼠标指定的位置放置文字“string”后,返回一个 `text` 图形对象句柄给 `h`。

命令 3 `legend`

功能 在图形上添加图例。该命令对有多种图形对象类型(线条图,条形图,饼形图等)的窗口中显示一个图例。对于每一线条,图例会在用户给定的文字标签旁显示线条的线型,标记符号和颜色等。当所画的是区域(patch 或 surface 对象)时,图例会在文字旁显示表面颜色。Matlab 在一个坐标轴中仅仅显示一个图例。图例的位置有几个因素决定,像遮挡的对象等,用户可以用鼠标拖动图例到恰当的位置,双击标签可以进入标签编辑状态。

用法 `legend('string1','string2',...)` 用指定的文字 `string` 在当前坐标轴中对所给数据的每一部分显示一个图例。

`legend(h,'string1','string2',...)` 用指定的文字 `string` 在一个包含于句柄向量 `h` 中的图形显示图例。用给定的数据对相应的图形对象加上图例。

`legend(string_matrix)` 用字符矩阵参量 `string_matrix` 的每一行字符串作为标签。

`legend(h,string_matrix)` 用字符矩阵参量 `string_matrix` 的每一行字符串作为标签给包含于句柄向量 `h` 中的相应的图形对象加标签。

`legend(axes_handle,...)` 给由句柄 `axes_handle` 指定的坐标轴显示图例。

`legend('off')` 从当前的坐标轴,或是由 `axes-handle` 指定的坐标轴中除掉图例。

`legend(axes_handle,'off')` 从由 `axes_handle` 指定的坐标轴中除掉图例。

`legend_handle = legend` 返回当前坐标轴中的图例句柄,若坐标轴中没有图例存在,则返回空向量。

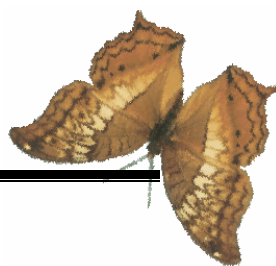
`legend` 对当前图形中所有的图例进行刷新。

`legend(legend_handle)` 对由句柄 `legend_handle` 指定的图例进行刷新。

`legend(...,pos)` 在指定的位置 `pos` 放置图例见表 7-5。

表 7-5

Pos 取值	pos=-1	pos=0	Pos=1
图例位置	坐标轴之外的右边	坐标轴之内,有可能遮挡部分图形	坐标轴的右上角(缺省位置)
Pos 取值	pos=2	pos=3	pos=4
图例位置	坐标轴的左上角	在坐标轴的左下角	坐标轴的右下角



`h = legend(...)` 返回图例的句柄向量。

`[legend_handle,object_handles] = legend(...)` 返回图例句柄, 该句柄为坐标轴定义于图例中的图形对象、line 对象、text 对象的句柄。这些句柄允许用户对每个对象进行详细的操作。

例 7-22

```
>>x = -pi:pi/20:pi;
>>plot(x,(cos(x)).^2,'rd',x,asin(x),'-b')
>>h = legend('cos2x','asin',2);
```

图形结果为图 7-22。

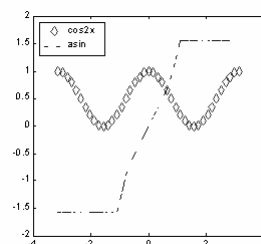


图 7-22

命令 4 title

功能 给当前轴加上标题。每个 axes 图形对象可以有一个标题。标题定位于 axes 的上方正中央。

用法 `title('string')` 在当前坐标轴上方正中央放置字符串 `string` 作为标题

`title(fname)` 先执行能返回字符串的函数 `fname`, 然后在当前轴上方正中央放置返回的字符串作为标题

`title(...,'PropertyName',PropertyValue,...)` 对由命令 `title` 生成的 text 图形对象的属性进行设置

`h = title(...)` 返回作为标题的 text 对象句柄。

命令 5 text

功能 在当前轴中创建 text 对象。函数 `text` 是创建 text 图形句柄的低级函数。可用该函数在图形中指定的位置上显示字符串。

用法 `text(x,y,'string')` 在图形中指定的位置(x,y)上显示字符串 `string`

`text(x,y,z,'string')` 在三维图形空间中的指定位置(x,y,z)上显示字符串 `string`

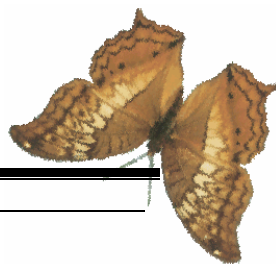
`text(x,y,z,'string','PropertyName',PropertyValue,...)` 对引号中的文字 `string` 定位于用坐标轴指定的位置, 且对指定的属性进行设置。表 7-6 给出文字属性名、含义及属性值。

表 7-6

属性名	属性说明	属性值
定义字符串		
Editing	能否对文字进行编辑	有效值: on、off 缺省值: off
Interpretation	TeX 字符是否可用	有效值: tex、none 缺省值: tex
String	字符串(包括 TeX 字符串)	有效值: 可见字符串
放置字符串		
Extent	text 对象的范围(位置与大小)	有效值: [left, bottom, width, height]
HorizontalAlignment	文字水平方向的对齐方式	有效值: left(文本外框左边对齐, 缺省对齐方式)、center(文本外框中间对齐)、right(文本外框右边对齐) 缺省值: left
Position	文字范围的位置	有效值: [x,y,z]直角坐标系 缺省值: [] (空矩阵)
Rotation	文字对象的方位角度	有效值: 标量(单位为度) 缺省值: 0



Units	文字范围与位置的单位	有效值: pixels (屏幕上的像素点)、normalized (把屏幕看成一个长、宽为 1 的矩形)、inches(英寸)、centimeters(厘米)、points (图象点)、data 缺省值: data
VerticalAlignment	文字垂直方向的对齐方式	有效值: top (文本外框顶上对齐)、cap(文本字符顶上对齐)、middle(文本外框中间对齐)、baseline(文本字符底线齐)、bottom(文本外框底线对齐) 缺省值: middle
指定文字字体		
FontAngle	设置斜体文字模式	有效值: normal(正常字体)、italic(斜体字)、oblique(斜角字) 缺省值: normal
FontName	设置文字字体名称	有效值: 用户系统支持的字体名或者字符串 FixedWidth。 缺省值为 Helvetica
FontSize	文字字体大小	有效值: 结合字体单位的数值 缺省值为: 10 points
FontUnits	设置属性 FontSize 的单位	有效值: points (1 点=1/72 英寸)、normalized(把父对象坐标轴作为一个单位长的一个整体;当改变坐标轴的尺寸时,系统会自动改变字体的大小)、inches (英寸)、Centimeters(厘米)、Pixels(像素) 缺省值: points
FontWeight	设置文字字体的粗细	有效值: light(细字体)、normal(正常字体)、demi(黑体字)、Bold(黑体字) 缺省值: normal
控制文字外观		
Clipping	设置坐标轴中矩形的剪辑模式	有效值: on、off on: 当文本超出坐标轴的矩形时,超出的部分不显示; off: 当文本超出坐标轴的矩形时,超出的部分显示。 缺省值: off
EraseMode	设置显示与擦除文字的模式。这些模式对生成动画系列与改进文字的显示效果很有好处。	有效值: normal、none、xor、background 缺省值: normal
SelectionHighlight	设置选中文字是否突出显示	有效值: on、off 缺省值: on
Visible	设置文字是否可见	有效值: on、off 缺省值: on
Color	设置文字颜色	有效的颜色值: ColorSpec
控制对文字对象的访问		
HandleVisibility	设置文字对象句柄对其他函数是否可见	有效值: on、callback、off 缺省值: on
HitTest	设置文字对象能否成为当前对象(见图形 CurrentObject 属性)	有效值: on、off 缺省值: on
文字对象的一般信息		
Children	文字对象的子对象(文字对象没有子对象)	有效值: [] (即空矩阵)
Parent	文字对象的父对象(通常为 axes 对象)	有效值: axes 的句柄
Seleted	设置文字是否显示出“选中”状态	有效值: on、off 缺省值: off



Tag	设置用户指定的标签	有效值: 任何字符串 缺省值: '' (即空字符串)
Type	设置图形对象的类型 (只读类型)	有效值: 字符串 'text'
UserData	设置用户指定数据	有效值: 任何矩阵 缺省值: [] (即空矩阵)
控制回调例行执行程序		
BusyAction	设置如何处理对文字回调过程中断的句柄	有效值: cancel、queue 缺省值: queue
ButtonDownFcn	设置当鼠标在文字上单击时, 程序做出的反应 (即执行回调程序)	有效值: 字符串 缺省值: '' (空字符串)
CreateFcn	设置当文字被创建时, 程序做出的反应 (即执行的回调程序)	有效值: 字符串 缺省值: '' (空字符串)
DeleteFcn	设置当文字被删除 (通过关闭或删除操作) 时, 程序做出的反应 (即执行的回调程序)	有效值: 字符串 缺省值: '' (空字符串)
Interruptible	设置回调过程是否可中断	有效值: on、off 缺省值: on (能中断)
UIContextMenu	设置与文字相关的菜单项	有效值: 用户相关菜单句柄

`h = text(...)` 返回文字对象句柄的列向量, 每一对象对应一句柄。该命令的其他使用形式中, 将随意地返回这个输出参量。

例 7-23

```
>>plot(0:pi/20:2*pi,sin(0:pi/20:2*pi))
>>text(pi,0,'Zeros Point')
>>grid on
```

图形结果为图 7-23。

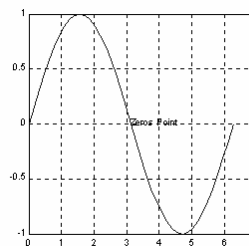


图 7-23

命令 6 xlabel、ylabel

功能 给 x、y 轴贴上标签

用法 `xlabel('string')、ylabel('string')` 给当前轴对象中的

x、y 轴贴标签; 注意: 若再次执行 `xlabel` 或 `ylabel` 命令, 则新的标签会覆盖旧的标签。

`xlabel(fname)、ylabel(fname)` 先执行函数 `fname`, 其返回一个字符串, 然后在 x、y 轴旁边显示出来;

`xlabel(..., 'PropertyName', PropertyValue, ...)`、`ylabel(..., 'PropertyName', PropertyValue)` 指定轴对象中的要控制的属性名和要改变的属性值, 这些都是由 `xlabel` 或 `ylabel` 创建的 `text` 图形对象的成对值;

`h = xlabel(...)`、`h = ylabel(...)` 返回作为标签的 `text` 对象的句柄。

7.2 三维图形

7.2.1 三维曲线、面填色命令

命令 1 comet3

功能 三维空间中的彗星图。彗星图为一个三维的动画图像, 彗星头 (一个小圆圈) 沿



着数据指定的轨道前进，彗星体为跟在彗星头后面的一段痕迹，彗星轨道为整个函数所画的实曲线。注意一点的是，该彗星轨迹的显示模式 `EraseMode` 为 `none`，所以用户不能打印出彗星轨迹（只能得到一个小圆圈），且若用户调整窗口大小，则彗星会消失。

用法 `comet3(z)` 用向量 `z` 中的数据显示一个三维彗星
`comet3(x,y,z)` 显示一个彗星通过数据 `x`, `y`, `z` 确定的三维曲线。

`comet3(x,y,z,p)` 指定彗星体的长度为：`p*length(y)`。

例 7-24

```
>>t = -20*pi:pi/50:20*pi;
>>comet3((cos(2*t).^2).*sin(t),(sin(2*t).^2).*cos(t),t);
```

图形的结果为图 7-24。

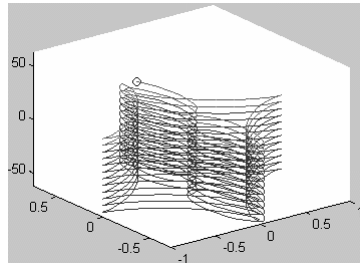


图 7-24

命令 2 fill3

功能 用指定的颜色填充三维多边形。阴影类型为平面型和 Gouraud 型。

用法 `fill3(X,Y,Z,C)` 填充由参数 `x`, `y` 和 `z` 确定多边形。若 `x`, `y` 或 `z` 为矩阵，`fill3` 生成 `n` 个多边形，其中 `n` 为矩阵的列数。在必要的时候，`fill3` 会自动连接最后一个节点和第一个节点。以便能形成封闭的多边形。参数 `c` 指定颜色，这儿 `c` 为引用当前色图的下标向量或矩阵。若 `c` 为行向量，则 `c` 的维数必须等于 `x` 的列数和 `y` 的列数，若 `c` 为列向量，则 `c` 的维数必须等于矩阵 `x` 的行数和 `y` 的行数。

`fill3(X,Y,Z,ColorSpec)` 用指定的颜色 `ColorSpec` 填充由 `x`, `y` 和 `z` 确定的多边形。

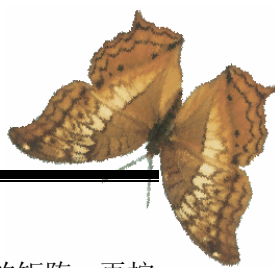
`fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...)` 对多边形的不同区域用不同的颜色进行填充。

`fill3(...,'PropertyName',PropertyValue)` 允许用户对特定的 `patch` 属性进行设置。

`h = fill3(...)` 返回 `patch` 图形对象的句柄向量，每一块 (`patch`) 对应一个句柄。

运算规则：

1. 若 `X`, `Y`, `Z` 为同型的矩阵，`fill3` 生成 `X`, `Y`, `Z` 中相同位置的元素确定的顶点，每一列生成一个多边形。
2. 若只有 `X`, `Y` 或 `Z` 为矩阵，则 `fill3` 由列向量参数生成可用的同型矩阵。
3. 若用户对填充的颜色指定为 `ColorSpec`，则 `fill3` 生成阴影类型为 `flat-shaded` 的多边形，且设置块 (`patch`) 的属性 `FaceColor` 为 RGB 颜色形式的矩阵。
4. 若用户用矩阵 `C` 指定颜色，命令 `fill3` 通过坐标轴属性 `Clim` 来调整 `C` 中的元素，在引用当前色图之前，用于指定颜色坐标轴的参数比例。
5. 若参数 `C` 为一行向量，命令 `fill3` 生成带平面阴影 (`flat-shaded`) 的多边形，同时设置补片对象的面颜色 (`FaceColor`) 属性为 `flat`。向量 `c` 中的每一元素成为每一补片对象的颜色数据 (`CData`) 属性的值。
6. 若参数 `C` 为一矩阵，命令 `fill3` 生成带内插颜色的多边形，同时设置多边形补片对象的 `FaceColor` 属性为 `interp`。命令 `fill3` 采用对多边形顶点色图的下标指定的颜色采用线性内插算法，同时多边形的颜色采用对顶点颜色用内插算法得到的颜色。矩阵 `C` 的每一列元素变



成对应补片对象的 Cdata 属性值。

7. 若参数 C 为一列向量, 命令 fill3 先复制 C 的元素, 使之成为所需维数的矩阵, 再按上面的方法 6 进行计算。

例 7-25

```
>>X = 10*rand(4);Y=10*rand(4);Z=10*rand(4);
>>C = rand(4);
>>fill3(X,Y,Z,C)
```

图形结果可能为图 7-25。

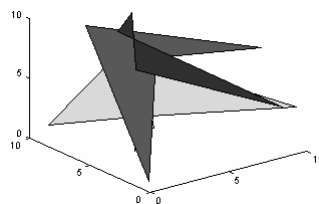


图 7-25

7.2.2 三维图形等高线

命令 1 contour

功能 曲面的等高线图

用法 contour(z) 把矩阵 z 中的值作为一个二维函数的值, 等高曲线是一个平面的曲线, 平面的高度 v 是 Matlab 自动取的;

contour(x,y,z) (x,y)是平面 z=0 上点的坐标矩阵, z 为相应点的高度值矩阵。效果同上;

contour(z,n) 画出 n 条等高线;

contour(x,y,z,n) 画出 n 条等高线;

contour(z,v) 在指定的高度 v 上画出等高线;

contour(x,y,z,v) 同上;

[c,h] = contour(...) 返回如同 contourc 命令描述的等高矩阵 c 和线句柄或块句柄列向量 h, 这些可作为 clabel 命令的输入参量, 每条线对应一个句柄, 句柄中的 userdata 属性包含每条等高线的高度值;

contour(...,'linespec') 因为等高线是以当前的色图中的颜色画的, 且是作为块对象处理的, 即等高线是一般的线条, 我们可象画普通线条一样, 可以指定等高线的颜色或者线形。

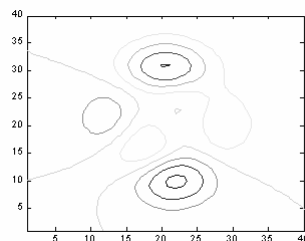


图 7-26

例 7-26

```
>>contour(peaks(40))
```

图形结果为图 7-26。

命令 2 clabel

功能 在二维等高线图中添加高度标签。在下列形式中, 若有 h 出现, 则会对标签进行恰当的旋转, 否则标签会竖直放置, 且在恰当的位置显示一个“+”号。

用法 clabel(C,h) 把标签旋转 to 恰当的角度, 再插入到等高线中。只有等高线之间有足够的空间时才加入, 当然这决定于等高线的尺度。

clabel(C,h,v) 在指定的高度 v 上显示标签 h, 当然要对标签做恰当的处理。

clabel(C,h,'manual') 手动设置标签。用户用鼠标左键或空格键在最接近指定的位置上放置标签, 用键盘上的回车键结束该操作。当然会对标签做恰当的处理。

clabel(C) 在从命令 contour 生成的等高线结构 c 的位置上添加标签。此时标签的放置的位置是随机的。



`clabel(C,v)` 在给定的位置 v 上显示标签

`clabel(C,'manual')` 允许用户通过鼠标来给等高线

贴标签

例 7-27

```
>>[x,y] = meshgrid(-2:2:2);
>>z = x.*y.*exp(-x.^2-y.^2);
>>[C,h] = contour(x,y,z);
>>clabel(C,h);
```

图形结果为图 7-27。

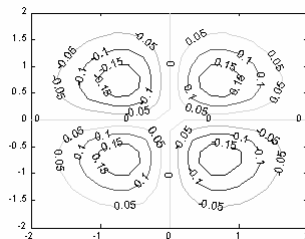


图 7-27

命令 3 `contourc`

功能 低级等高线图形计算命令。该命令计算等高线矩阵 c ，该矩阵可用于命令 `contour`，`contour3` 和 `contourf` 等。矩阵 z 中的数值确定平面上的等高线高度值，等高线的计算结果用由矩阵 z 维数决定的间隔的宽度。

用法 `C = contourc(Z)` 从矩阵 z 中计算等高矩阵，其中 z 的维数至少为 2×2 阶，等高线为矩阵 z 中数值相等的单元。等高线的数目和相应的高度值是自动选择的。

`C = contourc(Z,n)` 在矩阵 z 中计算出 n 个高度的等高线。

`C = contourc(Z,v)` 在矩阵 z 中计算出给定高度向量 v 上计算等高线，当然向量 v 的维数决定了等高线的数目。若只要计算一条高度为 a 的等高线，输入：

```
contourc(Z,[a,a]);
```

`C = contourc(x,y,Z)` 在矩阵 z 中，参量 x, y 确定的坐标轴范围内计算等高线；

`C = contourc(x,y,Z,n)` 从矩阵 Z 中，参量 x 与 y 确定的坐标范围内画出 n 条等高线；

`C = contourc(x,y,Z,v)` 从矩阵 Z 中，参量 x 与 y 确定的坐标范围内，画在 v 指定的高度上指定的等高线。

命令 4 `contour3`

功能 三维空间等高线图。该命令生成一个定义在矩形格栅上曲面的三维等高线图。

用法 `contour3(Z)` 画出三维空间角度观看矩阵 z 的等高线图，其中 z 的元素被认为是距离 xy 平面的高度，矩阵 z 至少为 2×2 阶的。等高线的条数与高度是自动选择的。若 $[m, n] = \text{size}(z)$ ，则 x 轴的范围为 $[1:n]$ ， y 轴的范围为 $[1:m]$ 。

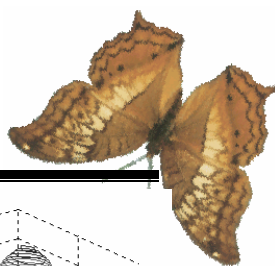
`contour3(Z,n)` 画出由矩阵 z 确定的 n 条等高线的三维图。

`contour3(Z,v)` 在参量 v 指定的高度上画出三维等高线，当然等高线条数与向量 v 的维数相同；若想只画一条高度为 h 的等高线，输入：`contour3(Z,[h,h])`

`contour3(X,Y,Z)`、`contour3(X,Y,Z,n)`、`contour3(X,Y,Z,v)` 用 X 与 Y 定义 x -轴与 y -轴的范围。若 X 为矩阵，则 $X(1,:)$ 定义 x -轴的范围；若 Y 为矩阵，则 $Y(:,1)$ 定义 y -轴的范围；若 X 与 Y 同时为矩阵，则它们必须同型。不论为哪种使用形式，所起的作用与命令 `surf` 相同。若 X 或 Y 有不规则的间距，`contour3` 还是使用规则的间距计算等高线，然后将数据转变给 X 或 Y 。

`contour3(...,LineStyle)` 用参量 `LineStyle` 指定的线型与颜色画等高线。

`[C,h] = contour3(...)` 画出图形，同时返回与命令 `contourc` 中相同的等高线矩阵 C ，包含所有图形对象的句柄向量 h ；除非没有指定 `LineStyle` 参数，`contour3` 将生成 `patch` 图形对象，且当前的 `colormap` 属性与 `caxis` 属性将控制颜色的显示。不



论使用何种形式,该命令都生成line图形对象。

例 7-28

```
>>[X,Y] = meshgrid([-2:.25:2]);
>>Z = X.*exp(-X.^2-Y.^2);
>>contour3(X,Y,Z,30)
```

图形结果为图 7-28。

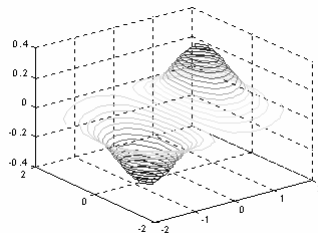


图 7-28

命令 5 **contourf**

功能 填充二维等高线图。即先画出不同等高线,然后相

邻的等高线之间用同一颜色进行填充。填充用的颜色决定于当前的色图颜色。

用法 `contourf(Z)` 矩阵 z 的等高线图,其中 z 理解成距平面的高度。 Z 至少为 2×2 阶的。等高线的条数与高度是自动选择的。

`contourf(Z,n)` 画出矩阵 z 的 n 条高度不同的等高线。

`contourf(Z,v)` 画出矩阵 z 的、由 v 指定的高度的等高线图。

`contourf(X,Y,Z)`、`contourf(X,Y,Z,n)`、`contourf(X,Y,Z,v)` 画出矩阵 z 的等高线图,其中 X 与 Y 用于指定 x -轴与 y -轴的范围。若 X 与 Y 为矩阵,则必须与 Z 同型。若 X 或 Y 有不规则的间距,`contour3` 还是使用规则的间距计算等高线,然后将数据转变给 X 或 Y 。

`[C,h,CF] = contourf(...)` 画出图形,同时返回与命令 `contourc` 中相同的等高线矩阵 C , C 也可被命令 `clabel` 使用;返回包含 `patch` 图形对象的句柄向量 h ;返回一用于填充用的矩阵 CF 。

例 7-29

```
>>contourf(peaks(30),20);
>>colormap gray
```

图形结果为图 7-29。

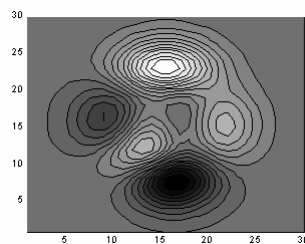


图 7-29

命令 6 **pie3**

功能 三维饼形图

用法 `pie3(X)` 用 x 中的数据画一个三维饼形图。 X

中的每一个元素代表三维饼形图中的一部分。

`pie3(X,explode)` x 中的某一部分可以从三维饼形图中分离出来。`explode` 是一个与 x 同型的向量或矩阵,`explode` 中非零的元素对应 x 中从饼形图中分离出来的分量。

`h = pie3(...)` 返回一个分量为 `patch`, `surface` 和 `text` 图形句柄对象的向量。即每一块对应一个句柄。

注意: 命令 `pie3` 将 x 的每一个元素在所有元素的总和中所占的比例表达出来。若 x 中的分量和小于 1 (则所有元素小于 1), 则认为 x 中的值指明三维饼形图的每一部分的大小。

例 7-30

```
>>x = [1 3 0.5 2.5 2]
>>ex = [0 1 0 0 0]
>>pie3(x,ex)
```

图形结果为图 7-30。

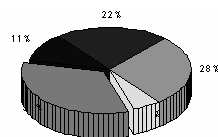


图 7-30



7.2.3 曲面与网格图命令

命令 1 mesh

功能 生成由 X、Y 和 Z 指定的网线面，由 C 指定的颜色的三维网格图。网格图是作为视点由 view(3) 设定的 surface 图形对象。曲面的颜色与背景颜色相同（当要动画显示不透明曲面时，这时可用命令 hidden 控制），或者当画一个标准的可透视的网线图时，曲面的颜色就没有（命令 shading 控制渲染模式）。当前的色图决定线的颜色。

用法 mesh(X,Y,Z) 画出颜色由 c 指定的三维网格图，所以和曲面的高度相匹配，

1. 若 X 与 Y 均为向量， $\text{length}(X)=n$ ， $\text{length}(Y)=m$ ，而 $[m, n]=\text{size}(Z)$ ，空间中的点 $(X(j), Y(I), Z(I,j))$ 为所画曲面网线的交点，分别地，X 对应于 z 的列，Y 对应于 z 的行。
2. 若 X 与 Y 均为矩阵，则空间中的点 $(X(I,j), Y(I,j), Z(I,j))$ 为所画曲面的网线的交点。

mesh(Z) 由 $[n, m]=\text{size}(Z)$ 得， $X=1:n$ 与 $Y=1:m$ ，其中 z 为定义在矩形划分区域上的单值函数。

mesh(...,C) 用由矩阵 c 指定的颜色画网线网格图。Matlab 对矩阵 c 中的数据进行线性处理，以便从当前色图中获得有用的颜色。

mesh(...,PropertyName',PropertyValue, ...) 对指定的属性 PropertyName 设置属性值 PropertyValue，可以在同一语句中对多个属性进行设置。

h = mesh(...) 返回 surface 图形对象句柄。

运算规则：

1. 数据 X、Y 和 z 的范围，或者是对当前轴的 XLimMode、YLimMode 和 ZLimMode 属性的设置决定坐标轴的范围。命令 axis 可对这些属性进行设置。

2. 参量 c 的范围，或者是对当前轴的 Clim 和 ClimMode 属性的设置（可用命令 caxis 进行设置），决定颜色的刻度化程度。刻度化颜色值作为引用当前色图的下标。

3. 网格图显示命令生成由于把 z 的数据值用当前色图表现出来的颜色值。Matlab 会自动用最大值与最小值计算颜色的范围（可用命令 caxis auto 进行设置），最小值用色图中的第一个颜色表现，最大值用色图中的最后一个颜色表现。Matlab 会对数据的中间值执行一个线性变换，使数据能在当前的范围内显示出来。

例 7-31

```
>>[X,Y] = meshgrid(-3:1.25:3);  
>>Z = peaks(X,Y);  
>>mesh(X,Y,Z);
```

图形结果为图 7-31。

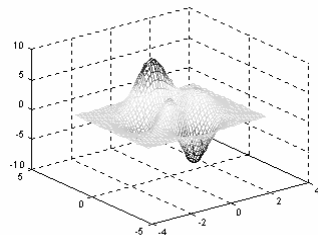
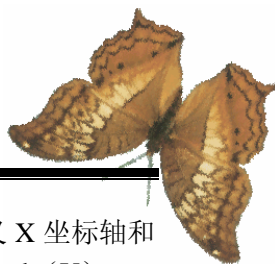


图 7-31

命令 2 surf

功能 在矩形区域内显示三维带阴影曲面图。

用法 surf(Z) 生成一个由矩阵 z 确定的三维带阴影的曲面图，其中 $[m, n]=\text{size}(Z)$ ，而 $X=1:n$ ， $Y=1:m$ 。高度 z 为定义在一个几何矩形区域内的单值函数，z 同时指定曲面高度数据的颜色，所以颜色对于曲面高度是恰当的。



`surf(X,Y,Z)` 数据 z 同时为曲面高度，也是颜色数据。 X 和 Y 为定义 X 坐标轴和 Y 坐标轴的曲面数据。若 X 与 Y 均为向量， $\text{length}(X)=n$ ， $\text{length}(Y)=m$ ，而 $[m,n]=\text{size}(Z)$ ，在这种情况下，空间曲面上的节点为 $(X(I),Y(j),Z(I,j))$ 。
`surf(X,Y,Z,C)` 用指定的颜色 c 画出三维网格图。Matlab 会自动对矩阵 c 中的数据进行线性变换，以获得当前色图中可用的颜色。

`surf(...,'PropertyName',PropertyValue)` 对指定的属性 `PropertyName` 设置为属性值 `PropertyValue`

`h = surf(...)` 返回一个 `surface` 图形对象句柄给变量 h 。

运算规则：

1. 严格地讲，一个参数曲面是由两个独立的变量 I, j 来定义的，它们在一个矩形区域上连续变化。例如， $a \leq I \leq b, c \leq j \leq d$ ，三个变量 X, Y, Z 确定了曲面。曲面颜色由第四参数矩阵 C 确定。

2. 矩形定义域上的点有如下关系：

$$\begin{array}{c} A(I-1,j) \\ | \\ B(I,j-1) \text{ ---- } C(I,j) \text{ ---- } D(I,j+1) \\ | \\ E(I+1,j) \end{array}$$

这个矩形坐标方格对应于曲面上的有四条边的块，在空间的点的坐标为 $[X(\odot), Y(\odot), Z]$ ，每个矩形内部的点根据矩形的下标和相邻的四个点连接；曲面上的点只有相邻的三个点，曲面上四个角上的点只有两个相邻点，上面这些定义了一个四边形的网格图。

3. 曲面颜色可以有两种方法来指定：指定每个节点的颜色或者是每一块的中心点颜色。在这种一般的设置中，曲面不一定为变量 X 和 Y 的单值函数，进一步而言，有四边的曲面块不一定为平面的，而可以用极坐标，柱面坐标和球面坐标定义曲面。

4. 命令 `shading` 设置阴影模式。若模式为 `interp`， C 必须与 X, Y, Z 同型；它指定了每个节点的颜色，曲面块内的颜色由附近几个点的颜色用双线性函数计算出来的。若模式为 `faced`（缺省模式）或 `flat`， $c(I,j)$ 指定曲面块中的颜色：

$$\begin{array}{ccc} A(I,j) & \text{-----} & B(I,j+1) \\ | & C(I,j) & | \\ C(I+1,j) & \text{-----} & D(I+1,j) \end{array}$$

在这种情形下， C 可以与 X, Y ，和 Z 同型，且它的最后一行和最后一列将被忽略，换句话说，就是 C 的行数和列数可以比 X, Y, Z 少 1。

5. 命令 `surf` 将指定图形视角为 `view(3)`。

6. 数据 X, Y, Z 的范围或者通过对坐标轴的属性 `XlimMode`，`YlimMode` 和 `ZlimMode` 的当前设置（可以通过命令 `axis` 来设置），将决定坐标轴的标签。

7. 参数 C 的范围或者通过对坐标轴的属性 `Clim` 和 `ClimMode` 的设置（可以通过命令 `caxis` 来设置），将决定颜色刻度化。刻度化的颜色值将作为引用当前色图的下标。

例 7-32

```
>>[X,Y,Z] = peaks(30);
>>surf(X,Y,Z)
>>colormap hsv
```

结果图形为图 7-32。

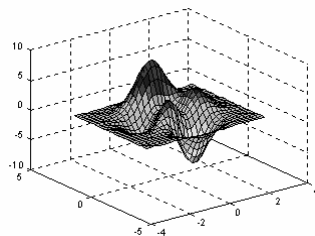


图 7-32

**命令 3 surf**

功能 在矩形区域内显示三维带阴影曲面图，且在曲面下面画出等高线。

用法 `surf(Z)`、`surf(X,Y,Z)`、`surf(X,Y,Z,C)`、
`surf(...,'PropertyName',PropertyValue)`、
`surf(...)`、`h = surf(...)`

上面各个使用形式的曲面效果与命令 `surf` 的相同，只不过是曲面下面增加了曲面的等高线而已。

例 7-33

```
>>[X,Y,Z] = peaks(30);  
>>surf(X,Y,Z)  
>>colormap hsv
```

图形结果为图 7-33。

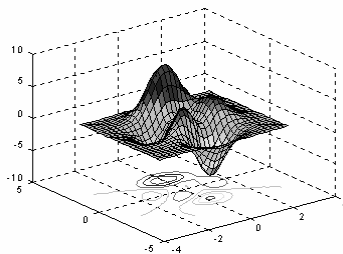


图 7-33

命令 4 surf

功能 画带光照模式的三维曲面图。该命令显示一个带阴影的曲面，结合了周围的，散射的和镜面反射的光照模式。想获得较平滑的颜色过度，要使用有线性强度变化的色图（如：`gray`, `copper`, `bone`, `pink` 等）。参数 `X`, `Y`, `Z` 确定的点定义了参数曲面的“里面”和“外面”，若用户想曲面的“里面”有光照模式，只要使用：

`surf(X',Y',Z')`

用法 `surf(Z)` 以向量 `z` 的元素生成一个三维的带阴影的曲面，其中阴影模式中的光源的方位、光照系数为缺省值（见下面）。

`surf(X,Y,Z)` 以矩阵 `X`, `Y`, `Z` 生成的一个三维的带阴影的曲面，其中阴影模式中的光源的方位、光照系数为缺省值（见下面）。

`surf(...,'light')` 用一个 `matlab` 光照对象（`light object`）生成一个带颜色、带光照的曲面，这与用缺省光照模式产生的效果不同。

`surf(...,'cdata')` 改变曲面颜色数据（`color data`），使曲面成为可反光的曲面。

`surf(...,s)` 指定光源与曲面之间的方位 `s`，其中 `s` 为一个二维向量 [`azimuth`, `elevation`]，或者三维向量 [`sx`, `sy`, `sz`]。缺省光源方位为从当前视角开始，逆时针 45°（度）。

`surf(X,Y,Z,s,k)` 指定反射系数 `k`，其中 `k` 为一个定义环境光（`ambient light`）系数（ $0 \leq ka \leq 1$ ）、漫反射（`diffuse reflection`）系数（ $0 \leq kb \leq 1$ ）、镜面反射（`specular reflection`）系数（ $0 \leq ks \leq 1$ ）与镜面反射亮度（以相素为单位）等的四维向量 [`ka`, `kd`, `ks`, `shine`]，缺省值为 `k=[0.55 0.6 0.4 10]`。

`h = surf(...)` 返回一个曲面图形句柄向量 `h`。

例 7-34

```
>>[X,Y] = meshgrid(-3:1/8:3);  
>>Z = peaks(X,Y);  
>>surf(X,Y,Z);  
>>shading interp  
>>colormap(gray);
```

图形结果为图 7-34。

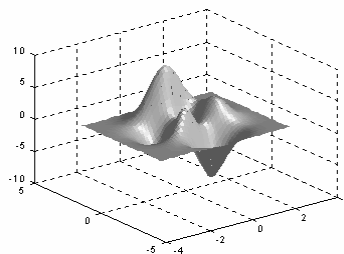
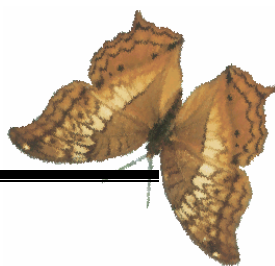


图 7-34

**命令 5 waterfall****功能** 瀑布图

用法 `waterfall(X,Y,Z)` 用所给参数 X 、 Y 与 Z 的数据画一“瀑布”效果图。若 X 与 Y 都是向量，则 X 与 Z 的列相对应， Y 与 Z 的行相对应，即 $\text{length}(X)=Z$ 的列数， $\text{length}(Y)=Z$ 的行数。参数 X 与 Y 定义了 x -轴与 y -轴， Z 定义了 z -轴的高度， Z 同时确定了颜色，所以颜色能恰当地反映曲面的高度。若想研究数据的列，可以输入：`waterfall(Z')`或 `waterfall(X',Y',Z')`

`waterfall(Z)` 画出一瀑布图，其中缺省地有： $X=1:Z$ 的列数， $Y=1:Z$ 的行数，且 Z 同时确定颜色，所以颜色能恰当地反映曲面高度。

`waterfall(...,C)` 用比例化的颜色值从当前色图中获得颜色，参量 C 决定颜色的比例，为此，必须与 Z 同型。系统使用一线性变换，从当前色图中获得颜色。

`h = waterfall(...)` 返回 `patch` 图形对象的句柄 h ，可用于画出图形。

例 7-35

```
>>[X,Y,Z] = peaks(30);
>>waterfall(X,Y,Z)
```

图形结果为图 7-35。

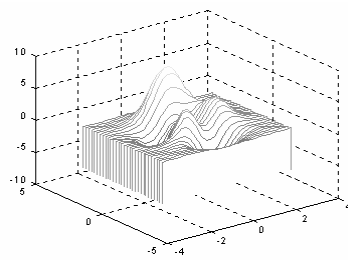


图 7-35

命令 6 cylinder

功能 生成圆柱图形。该命令生成一单位圆柱体的 x -， y -， z -轴的坐标值。用户可以用命令 `surf` 或命令 `mesh` 画出圆柱形对象，或者用没有输出参量的形式而立即画出图形。

用法 `[X,Y,Z] = cylinder` 返回一半径为 1、高度为 1 的圆柱体的 x -， y -， z -轴的坐标值，圆柱体的圆周有 20 个距离相同的点。

`[X,Y,Z] = cylinder®` 返回一半径为 r 、高度为 1 的圆柱体的 x -， y -， z -轴的坐标值，圆柱体的圆周有 20 个距离相同的点。

`[X,Y,Z] = cylinder(r,n)` 返回一半径为 r 、高度为 1 的圆柱体的 x -， y -， z -轴的坐标值，圆柱体的圆周有指定的 n 个距离相同的点。

`cylinder(...)` 没有任何的输出参量，直接画出圆柱体。

例 7-36

```
>>t = 0:pi/10:2*pi;
>>[X,Y,Z] = cylinder(2+(cos(t)).^2);
>>surf(X,Y,Z); axis square
```

图形结果为图 7-36。

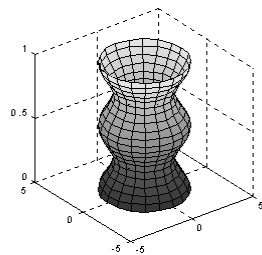


图 7-36

命令 7 sphere**功能** 生成球体

用法 `sphere` 生成三维直角坐标系中的单位球体。该单位球体由 $20*20$ 个面。

`sphere(n)` 在当前坐标系中画出有 $n*n$ 个面的球体

`[X,Y,Z] = sphere(n)` 返回三个阶数为 $(n+1)*(n+1)$ 的，直角坐标系中的坐标矩阵。

该命令没有画图，只是返回矩阵。用户可以用命令 `surf(X,Y,Z)` 或 `mesh`



(X, Y, Z) 画出球体。

例 7-37

```
>>[X,Y,Z]=sphere;
>>mesh(X,Y,Z)
>>hidden off
```

图形结果为图 7-37。

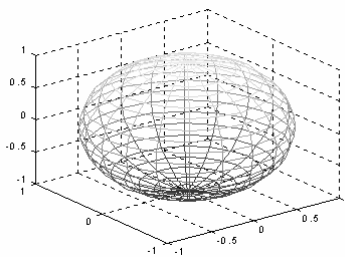


图 7-37

7.2.4 三维数据的其他表现形式命令

命令 1 pcolor

功能 伪彩色图。该图为一矩形单元的、由参数 c 定义了颜色的阵列，系统通过 c 中的每相邻的四点定义的曲面补片而生成一伪彩色图。是从上面向下观看的“平面”曲面图。若用户使用命令 `shading faceted` 或 `shading flat`，则每一单元的固定颜色是与之相连的角的颜色有关的。所以， $C(i,j)$ 定义了单元的地 i 行与地 j 列的颜色。 C 中的最后一行与最后一列都没有用上。若用户使用命令 `shading interp`，则每一单元的颜色是对它的四个顶点的颜色进行一双线性插值后的颜色，这时 c 的所有元素都参加了运算。

用法 `pcolor(C)` 画一伪彩色图。 C 中的元素都线性地映射于当前色图下标。从 C 映射到当前的色图是由命令 `colormap` 和 `caxis` 定义的。

`pcolor(X,Y,C)` 在参数 x 和 y 指定的位置上画一由 C 确定的彩色图。该图为一逻辑上为矩形、带二维格栅的、顶点在 $[X(i,j), Y(i,j)]$ 的图形（若 X 和 Y 为矩阵时）。参量 X 与 Y 为指定格栅线的向量或矩阵。若 X 与 Y 为向量，则 X 对应于 C 的列，而 y 对应于 C 的行；若 X 与 Y 同为矩阵，则必须为同型矩阵。该命令等价于命令：`surf(X,Y,0,C)`，观察角度为：`view([0,90])`。

$h = \text{pcolor}(\dots)$ 返回一 surface 图形对象句柄于 h

例 7-38

```
>>pcolor(magic(20))
>>colormap(gray(2))
>>axis ij;axis square
```

图形结果为图 7-38。

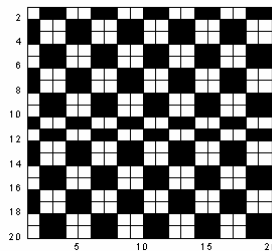


图 7-38

命令 2 quiver

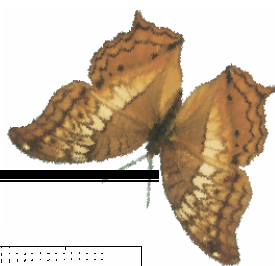
功能 矢量图或速度图

用法 `quiver(U,V)` 在范围为 $x = 1:n$ 和 $y = 1:m$ 的坐标系中显示由 U 和 V 定义的向量，而 $[m,n] = \text{size}(U) = \text{size}(V)$ ，这种形式是在一个几何矩形中画出 U 和 V 的，`quiver` 命令本身会自动地画出这些向量，使之不会重叠。

`quiver(X,Y,U,V)` 由向量 X 和 Y 中的分量的任意组合而成的向量与。若 X 与 Y 都是向量 $\text{length}(X) = n$ ，而 $\text{length}(Y) = m$ ，而 $[m, n] = \text{size}(U) = \text{size}(V)$ ，向量 X 对应于矩阵 U 、 V 的列向量，而向量 Y 对应于矩阵 U 、 V 的行向量。

`quiver(\dots, scale)` 自动对向量的长度进行处理。使之不会重叠，当然可以对 `scale` 进行取值，若 `scale=2`，则向量长度伸长 2 倍，若 `scale=0`，则如实画出向量图。

`quiver(\dots, LineSpec)` 可以指定画矢量图用的线型，符号，颜色，`quiver` 命令会在



原来的向量图上画出记号。

`quiver(...,LineStyle,'filled')` 对用 `LineStyle` 指定的记号进行填充

`h = quiver(...)` 返回每个向量图的句柄

例 7-39

```
>>[z,x,y]=peaks(30);
>>[Dx,Dy]=gradient(z,0.1,0.1);
>>quiver(x,y,Dx,Dy)
```

图形结果为图 7-39。

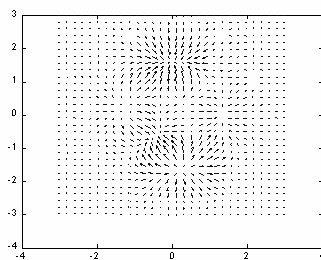


图 7-39

命令 3 slice

功能 立体切片图。该命令显示通过立体图形的矩形切片图。

用法 `slice(X,Y,Z,V,sx,sy,sz)` 显示三元函数 $V=V(X,Y,Z)$ 确定的超立体形在 x -轴、 y -轴与 z -轴方向上的若干点（对应若干平面。即若函数 $V=V(X,Y,Z)$ 中有一变量如 X 取一定值 X_0 ，则函数 $V=V(X_0,Y,Z)$ 变成一立体曲面（只不过是该曲面通过颜色表示高度 V ，从而显示于一平面而已。）的切片图，各点的坐标由参量向量 sx 、 sy 与 sz 指定。参量 X 、参量 Y 与参量 Z 为三维数组，用于指定立方体 V 的坐标。参量 X 、 Y 与 Z 必须有单调的、正交的间隔（如同用命令 `meshgrid` 生成的一样）。在每一点上的颜色由对超立体 V 的三维内插值确定。

`slice(V,sx,sy,sz)` 显示三元函数 $V=V(X,Y,Z)$ 确定的超立体形在 x -轴、 y -轴与 z -轴方向上的若干点（对应若干平面）的切片图，各点的坐标由数量向量 sx 、 sy 与 sz 指定。其中 V 为三维数组（阶数为 $m \times n \times p$ ），缺省地有： $X = 1:m$ 、 $Y = 1:n$ 、 $Z = 1:p$ 。

`slice(V,XI,YI,ZI)` 显示参量矩阵 XI 、 YI 与 ZI 确定的、超立体图形的切面图。参量 XI 、 YI 与 ZI 定义了一曲面，同时会在曲面的点上计算超立体 V 的值。参量 XI 、 YI 与 ZI 必须为同型矩阵。

`slice(X,Y,Z,V,XI,YI,ZI)` 沿着由矩阵 XI 、 YI 与 ZI 定义的曲面画穿过超立体图形 V 的切片。

`slice(...,'method')` 指定内插值的方法。‘method’ 为如下方法之一：‘linear’、‘cubic’、‘nearest’：

‘linear’ —— 指定使用三次线性内插值法（该状态为缺省的）；

‘cubic’ —— 指定使用三次立方内插值法；

‘nearest’ —— 指定使用最近点内插值法。

`h = slice(...)` 返回一曲面图形对象的句柄向量 h 。

命令 4 axis

功能 坐标轴的刻度与外在显示

用法 `axis([xmin xmax ymin ymax])` 设置当前坐标轴的 x -轴与 y -轴的范围。

`axis([xmin xmax ymin ymax zmin zmax cmin cmax])` 设置当前坐标轴的 x -轴、 y -轴与 z -轴的范围，当前颜色刻度范围。该命令也同时设置当前坐标轴的属性 $Xlim$ 、 $Ylim$ 与 $Zlim$ 为所给参数列表中的最大值和最小值。另外，坐标轴属



性 XlimMode、YlimMode 与 ZlimMode 设置为 ‘manual’。

`v = axis` 返回一包含 x-轴、y-轴与 z-轴的刻度因子的行向量，其中 `v` 为一四维或六维向量，这取决于当前坐标为二维还是三维的。返回的值包含当前坐标轴的 Xlim、Ylim 与 Zlim 属性值。

`axis auto` 设置系统到它的缺省动作——自动计算当前轴的范围，这取决于输入参数 `x`、`y` 与 `z` 的数据中的最大值与最小值。同时将当前坐标轴的属性 XlimMode、YlimMode 与 ZlimMode 设置为 ‘auto’ 用户可以指定对某一坐标轴进行自动操作。例如：

`axis 'auto x'` 将自动计算 x-轴的范围；

`axis 'auto yz'` 将自动计算 y-轴与 z-轴的范围。

`axis manual`、`axis(axis)` 把坐标固定在当前的范围，这样，若保持状态(hold)为 on，后面的图形仍用相同界限。该命令设置了属性 XlimMode、属性 YlimMode 与属性 ZlimMode 为 manual。

`axis tight` 把坐标轴的范围定为数据的范围，即坐标轴中没有多余的部分。

`axis fill` 该命令用于将坐标轴的取值范围分别设置为绘图所用数据在相应方向上的最大、最小值。

`axis ij` 使用矩阵坐标系：坐标原点在左上角、横坐标（j-轴）的值从左到右增加，纵坐标（i-轴）的值从上到下增加。

`axis xy` 使用笛卡儿坐标系（缺省）：坐标原点在左下角、横坐标（x-轴）的值从左到右增加，纵坐标（y-轴）的值从下到上增加。

`axis equal` 设置坐标轴的纵横比，使在每个方向的数据单位都相同。其中 x-轴、y-轴与 z-轴将根据所给数据在各个方向的数据单位自动调整其纵横比。

`axis image` 效果与命令 `axis equal` 相同，只是图形区域刚好紧紧包围图象数据。

`axis square` 设置当前图形为正方形（或立方体形），系统将调整 x-轴、y-轴与 z-轴，使它们有相同的长度，同时相应地自动调整数据单位之间的增加量。

`axis normal` 自动调整坐标轴的纵横比，还有用于填充图形区域的、显示于坐标轴上的数据单位的纵横比。

表 7-7 显示由上面三个命令设置的坐标轴属性。

表 7-7

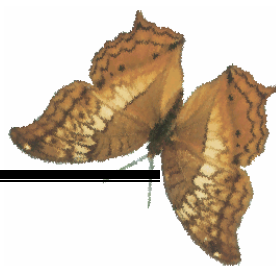
命令 坐标轴属性	axis equal	axis normal	axis square	axis tightequal
DataAspectRatioMode	[1 1 1]	没有设置	没有设置	[1 1 1]
PlotBoxAspectRatio	manual	auto	auto	Manual
PlotBoxAspectRatioMode	[3 4 4]	没有设置	[1 1 1]	Auto
Stretch-to-fill	禁止	可行	禁止	禁止

`axis vis3d` 该命令将冻结坐标系此时的状态，以便进行旋转。

`axis off` 关闭所用坐标轴上的标记、格栅和单位标记。但保留由 `text` 和 `gtext` 设置的对象。

`axis on` 显示坐标轴上的标记、单位和格栅。

`[mode,visibility,direction] = axis('state')` 返回表明当前坐标轴的设置属性的三个字



字符串，见表 7-8。

表 7-8

输出参量	返回字符串	说明
Mode	'auto'或 'manual'	若 XLimMode、YlimMode 与 ZlimMode 都设置为 auto, 则 mode 为 auto; 若 XLimMode、YlimMode 或者 ZlimMode 都设置为 manual, 则 mode 为 manual
Visibility	'on'或'off'	
Direction	'xy'或'ij'	

例 7-40

```
>>x = 0:.025:pi/2;
>>plot(x,exp(x).sin(2*x),'-m<')
>>axis([0 pi/2 0 5])
```

图形结果为图 7-40。

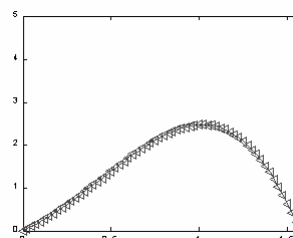


图 7-40

命令 5 hidden

功能 在网格图中显示隐含线条。隐含线条的显示，实际上是显示那些从观察角度观看没有被其他物体遮住的线条。

用法 `hidden on` 对当前图形打开隐含线条的显示状态，

使网格图后面的线条被前面的线条遮住。设置曲面图形对象的属性 `FaceColor` 为坐标轴背景颜色。这是系统的缺省操作。

`hidden off` 对当前图形关闭隐含线条的显示

`hidden` 在两种状态 `on` 与 `off` 之间切换

例 7-41

```
>>mesh(peaks)
>>hidden off
```

图形结果为图 7-41。

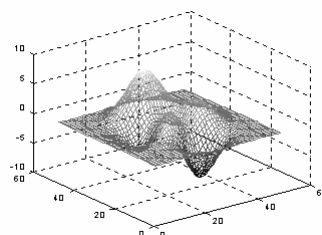


图 7-41

命令 6 shading

功能 设置颜色色调属性。该命令控制曲面与补片等的图形对象的颜色色调。同时设置当前坐标轴中的所有曲面与补片图形对象的属性 `EdgeColor` 与 `FaceColor`。命令 `shading` 设置恰当的属性值，这取决于曲面或补片对象是表现网格图或实曲面。

用法 `shading flat` 使网格图上的每一线段与每一小面有一相同颜色，该颜色由线段的末端的端点颜色确定；或由小面的、有小型的下标或索引的四个角的颜色确定。

`shading faceted` 带重叠的黑色网格线的平面色调模式。这是缺省的色调模式。

`shading interp` 在每一线段与曲面上显示不同的颜色，该颜色为通过在每一线段两边的、或者为不同小曲面之间的色图的索引或真颜色进行内插值得到的颜色。

例 7-42

```
>>sphere(16)
>>axis square
>>shading flat
>>title('Flat Shading')
```

图形结果为图 7-42。

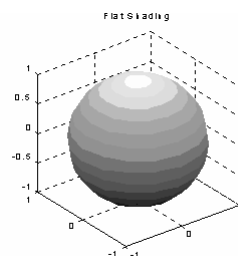


图 7-42

命令 7 caxis

功能 颜色坐标轴刻度。命令 `caxis` 控制着对应色图的数据值的



映射图。它影响下面对象之一的、用带索引的颜色数据（CData）与颜色数据映射（CDataMapping）控制的刻度的图形对象 `surface`、`patches` 与 `images`；它没有影响带用颜色数据（CData）或颜色数据映射（CDataMapping）直接设置的颜色的图形对象 `surface`、`images` 或 `patches`。该命令还改变坐标轴图形对象的属性 `Clim` 与 `ClimMode`。

用法 `caxis([cmin cmax])` 用指定的最大值与最小值设置颜色范围。数据值中小于 `cmin` 或大于 `cmax` 的，将分别地映射于 `cmin` 与 `cmax`；处于 `cmin` 与 `cmax` 之间的数据将线性地映射于当前色图。

`caxis auto` 让系统自动地计算数据的最大值与最小值对应的颜色范围。这是系统的缺省动作。数据中的正无穷大（`Inf`）对应于最大颜色值；负无穷大（`-Inf`）对应于最小颜色值；带颜色值设置为 `NaN` 的面或者边界将不显示。

`caxis manual`、`caxis(caxis)` 冻结当前颜色坐标轴的刻度范围。这样，当 `hold` 设置为 `on` 时，可使后面的图形命令使用相同的颜色范围。

`v = caxis` 返回一包含当前正在使用的颜色范围的二维向量 `v=[cmin cmax]`。

`caxis(axis_handle,...)` 使由参量 `axis_handle` 指定的坐标轴，而非当前坐标轴。

颜色坐标轴刻度工作原理：

使用带索引的颜色数据(Cdata)与颜色数据映射(CdataMapping)的图形对象 `surface`、`patch` 与 `image` 将设置成刻度化的，在每次图形渲染时，将映射颜色数据值为当前图形的颜色。当颜色数据值等于或小于 `cmin` 时，将它映射为当前色图中的第一个颜色；当颜色数据值等于或大于 `cmax` 时，将它映射为当前色图中的最后一个颜色；对于处于 `cmin` 与 `cmax` 之间的颜色数据（例如 `c`），系统将执行下列线性转换，以获得对应当前色图（它的长度为 `m`）中的颜色的索引（当前色图的行指标 `index`）：

$$\text{index} = \text{fix}((C-\text{min})/(\text{cmax}-\text{cmin})*m)+1$$

例 7-43

```
>>[X,Y,Z] = sphere;
>>C = Z;surf(X,Y,Z,C)
>>caxis([-1 3])
```

图形结果为图 7-43。

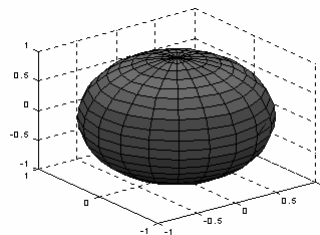


图 7-43

命令 8 view

功能 指定立体图形的观察点。观察者（观察点）的位置决定了坐标轴的方向。用户可以用方位角（`azimuth`）和仰角（`elevation`）一起，或者用空间中的一点来确定观察点的位置。

用法 `view(az,el)`、`view([az,el])` 给三维空间图形设置观察点的方位角。方位角 `az` 与仰角 `el` 为这两个旋转角度：做一通过视点与 `z`-轴的平面，与 `xy` 平面有一交线，该交线与 `y`-轴的反方向的、按逆时针方向（从 `z`-轴的方向观察）计算的、单位为度的夹角，就是观察点的方位角 `az`。若角度为负值，则按顺时针方向计算；在通过视点与 `z`-轴的平面上，用一直线连接视点与坐标原点，该直线与 `xy` 平面的夹角就是观察点的仰角 `el`。若仰角为负值，则观察点转移到曲面下面。

`view([x,y,z])` 在笛卡儿坐标系中于点 `(x,y,z)` 设置视点。注意：输入参量只能是方括号的向量形式，而非数学中的点的形式。



view(2) 设置缺省的二维形式视点。其中 $az=0$, $el=90$, 即从 z -轴上方观看。

view(3) 设置缺省的三维形式视点。其中 $az=-37.5$, $el=30$ 。

view(T) 根据转换矩阵 T 设置视点。其中 T 为 4×4 阶的矩阵, 如同用命令 **viewmtx** 生成的透视转换矩阵一样。

[az,el] = view 返回当前的方位角 az 与仰角 el 。

T = view 返回当前的 4×4 阶的转换矩阵 T 。

例 7-44

```
>>peaks;
>>az = 0;el = 90;
>>view(az, el)
```

图形结果为图 7-44。

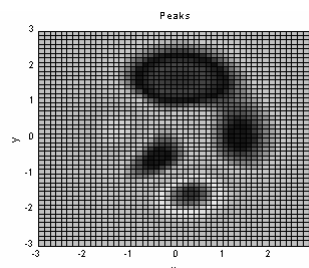


图 7-44

命令 9 viewmtx

功能 视点转换矩阵。计算一个 4×4 阶的正交的或透视的转换矩阵, 该矩阵将一四维的、齐次的向量转换到一个二维的视平面上 (如计算机平面上)。

用法 **T = viewmtx(az,el)** 返回一与视点的方位角 az 与仰角 el (单位都为度) 对应的正交矩阵, 并没有改变当前视点。

T = viewmtx(az,el,phi) 返回一透视的转换矩阵, 其中参量 ϕ 是单位为度的透视角度, 为标准化立方体 (单位为度) 的对像视角角度与透视扭曲程度。

表 7-9

Phi 的值	说明
0 度	正交投影
10 度	类似以远距离投影
25 度	类似以普通投影
60 度	类似以广角投影

用户可以通过使用返回的矩阵, 用命令 **view(T)** 改变视点的位置。该 4×4 阶的矩阵将变换四维的、同次的向量成形式为 (x,y,z,w) 的非标准化的向量, 其中 w 不等于 1。正交化的 x -元素与 y -元素组成的向量 $(x/w,y/w,z/w,1)$ 为我们所需的二维向量。(注: 一四维同次向量为在对应的三维向量后面增加一个 1。例如: $[x,y,z,1]$ 为对应于三维空间中的点 $[x,y,z]$ 的四维向量。)

T = viewmtx(az,el,phi,xc) 返回以在标准化的图形立方体中的点 xc 为目标点的透视矩阵 (就像相机正对着点 xc 一样), 目标点 xc 为视角的中心点。用户可以用一三维向量 $xc=[xc,yc,zc]$ 指定该中心点, 每一分量都在区间 $[0, 1]$ 上。缺省值为 $xc=[0\ 0\ 0]$ 。

命令 10 surfnorm

功能 计算与显示三维曲面的法线。该命令计算用户命令 **surf** 中的曲面法线。

用法 **surfnorm(Z)**、**surfnorm(X,Y,Z)** 画出一曲面与它的法线图。其中矩阵 Z 用于指定曲面的高度值; X 与 Y 为向量或矩阵, 用于定义曲面的 x 与 y 部分。

[Nx,Ny,Nz] = surfnorm(...) 返回组成曲面的法线在三个坐标轴上的投影分量 Nx,Ny 与 Nz 。

例 7-45

```
>>[x,y,z] = cylinder(1:10);
>>surfnorm(y,x,z)
```



```
>>axis([-12 12 -12 12 -0.1 1])
```

图形结果为图 7-45。

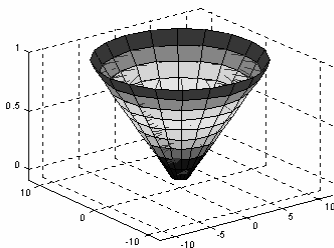


图 7-45 曲面法线图

7.3 通用图形函数命令

7.3.1 图形对象句柄命令

命令 1 figure

功能 创建一个新的图形对象。图形对象为在屏幕上单独的窗口，在窗口中可以输出图形。

用法 figure 用缺省的属性值创建一个新的图形对象。

figure('PropertyName',PropertyValue,...) 对指定的属性 PropertyName 用指定的属性值 PropertyValue（属性名与属性值成对出现）创建一个新的图形窗口，对于那些没有指定的属性，则用缺省值。属性名与有效的属性值见下表。

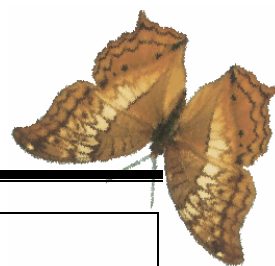
figure(h) 1. 若 h 为一个已经存在的图形的句柄，则 figure(h)使由 h 标记的图形成为当前图形，使它可见，且在屏幕上把它显示到所有图形之前。当前图形为图像输出的地方。

2. 若 h 不是已经存在图形的句柄，但是为一整数，则该命令生成一图形窗口，同时把该窗口的句柄赋值为 h；若 h 不是一图形窗口的句柄，也不是一整数，则返回一错误信息。

h = figure(...) 返回图形窗口对象的句柄给 h。

表 7-10

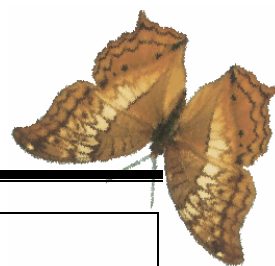
属性名	属性说明	有效属性值
窗口位置		
Position	图形窗口的位置与大小	有效值：四维向量[left,bottom,width,height] 缺省值：决定于显示
Units	用于解释属性 Position 的单位	有效值：inches(英寸) centimeters(厘米) normalized(标准化单位，认为窗口为一长宽都是 1) points(点) pixels(像素)



		characters(字符) 缺省值: pixels
指定类型与外在显示		
Color	窗口的背景颜色	有效值: ColorSpec (有效的颜色 参数) 缺省值: 取决于颜色表(参见命令 colordef)
Menubar	转换图形窗口菜单条的“开”与“关”	有效值: none、figure 缺省值: figure
Name	显示图形窗口的标题	有效值: 任意字符串 缺省值: '' (空字符串)
NumberTitle	标题栏中是否显示'Figure No. n',其中 n 为图形窗口的编号	有效值: on、off 缺省值: on
Resize	指定图形窗口是否可以通过鼠标改变大小	有效值: on、off 缺省值: on
SelectionHighlight	当图形窗口被选中时,是否突出显示	有效值: on、off 缺省值: on
Visible	确定图形窗口是否可见	有效值: on、off 缺省值: on
WindowStyle	指定窗口为标准窗口还是典型窗口	有效值: normal (标准窗口)、modal (典型窗口) 缺省值: normal
控制色图		
Colormap	图形窗口的色图	有效值: m*3 阶的 RGB 颜色矩阵 缺省值: jet 色图
Dithermap	用于真颜色数据以伪颜色显示的色图	有效值: m*3 阶的 RGB 颜色矩阵 缺省值: 有所有颜色的色图
DithermapMode	是否使用系统生成的抖动色图	有效值: auto、manual 缺省值: manual
FixedColors	不是从色图中获得的颜色	有效值: m*3 阶的 RGB 颜色矩阵 缺省值: 无 (只读模式)
MinColormap	系统颜色表中能使用的最少颜色数	有效值: 任一标量 缺省值: 64
ShareColors	允许 MATLAB 共享系统颜色表中的颜色	有效值: on、off 缺省值: on
指定透明度		
Alphamap	图形窗口的 α 色图,用于设定透明度。	有效值: m*1 维向量,每一分量在[0 1]之间 缺省值: 64*1 维向量
指定渲染模式		
BackingStore	打开或关闭屏幕像素缓冲区	有效值: on、off 缺省值: on
DoubleBuffer	对于简单的动画渲染是否使用快速缓冲	有效值: on、off 缺省值: off
Renderer	用于屏幕和圖片的渲染模式	有效值: painters、zbuffer、OpenGL 缺省值: 系统自动选择
关于图形窗口的一般信息		
Children	显示于图形窗口中的任意对象句柄	有效值: 句柄向量
FileName	命令 guide 使用的文件名	有效值: 字符串
Parent	图形窗口的父对象: 根屏幕	有效值: 总是 0 (即根屏幕)
Selected	是否显示窗口的“选中”状态	有效值: on、off 缺省值: on
Tag	用户指定的图形窗口	有效值: 任意字符串



	标签	缺省值: ' ' (空字符串)
Type	图形对象的类型 (只读类型)	有效值: 'figure'
UserData	用户指定的数据	有效值: 任一矩阵 缺省值: [] (空矩阵)
RendererMode	缺省的或用户指定的渲染程序	有效值: auto、manual 缺省值: auto
关于当前状态的信息		
CurrentAxes	在图形窗口中的当前坐标轴的句柄	有效值: 坐标轴句柄
CurrentCharacter	在图形窗口中最后一个输入的字符	有效值: 单个字符
CurrentObject	图形窗口中的当前对象的句柄	有效值: 图形对象句柄
CurrentPoint	图形窗口中最后单击的按钮的位置	有效值: 二维向量[x-coord, y-coord]
SelectionType	鼠标选取类型	有效值: normal、extended、alt、open
回调程序的执行		
BusyAction	指定如何处理中断调用程序	有效值: cancel、queue 缺省值: queue
ButtonDownFcn	当在窗口中空闲点按下鼠标按钮时, 执行的回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
CloseRequestFcn	当执行命令关闭时, 定义一回调程序	有效值: 字符串 缺省值: closereq
CreateFcn	当打开一图形窗口时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
DeleteFcn	当删除一图形窗口时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
Interruptible	定义一回调程序是否可中断	有效值: on、off 缺省值: on (可以中断)
KeyPressFcn	当在图形窗口中按下一键时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
ResizeFcn	当图形窗口改变大小时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
UIContextMenu	定义与图形窗口相关的菜单	有效值: 属性 UIContextmenu 的句柄
WindowButtonDownFcn	当在图形窗口中按下鼠标时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
WindowButtonMotionFcn	当将鼠标移进图形窗口中时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
WindowButtonUpFcn	当在图形窗口中松开按钮时, 定义一回调程序	有效值: 字符串 缺省值: ' ' (空字符串)
访问对象的控制		
IntegerHandle	指定使用整数或非整数图形句柄	有效值: on、off 缺省值: on (整数句柄)
HandleVisiblity	指定图形窗口句柄是否可见	有效值: on、callback、off 缺省值: on
HitTest	定义图形窗口是否能变成当前对象(参见图	有效值: on、off 缺省值: on



	形 窗 口 属 性 CurrentObject)	
NextPlot	在图形窗口中定义如何显示另外的图形	有效值: replacechildren、add、replace 缺省值: add
定义鼠标指针		
Pointer	选取鼠标记号	有效值: crosshair、arrow、topr、watch、topl、botl、botr、circle、cross、fleur、left、right、top、fullcrosshair、bottom、ibeam、custom 缺省值: arrow
PointerShapeCData	定义鼠标外形的数据	有效值: 16*16 阶矩阵 缺省值: 将鼠标设置为'custom'且可见
PointerShapeHotSpot	设置鼠标活跃的点	有效值: 二维向量[row, column] 缺省值: [1 1]

例 7-46

```
>>scrsz = get(0,'ScreenSize');
>>figure('Position',[1 scrsz(4)/2 scrsz(3)/2 scrsz(4)/2])
```

执行上面的语句, 会在屏幕的左上角生成一没有任何符号的窗口。

命令 2 line

功能 生成线(line)对象。命令 line 在当前坐标轴中生成一个线对象。用户可以指定线的颜色, 宽度, 类型和标记符号等其他特性。

命令 line 有两种形式:

1. 自动循环使用颜色和类型。当用户用非正式语法来指定矩阵坐标数据: line(X,Y,Z), Matlab 将循环使用由坐标轴 ColorOrder 和 LineStyle 指定的颜色顺序和类型顺序。
2. 纯粹低级操作。当用户用属性名和属性值调用命令 line:

```
line('XData',x,'YData',y,'ZData',z)
```

Matlab 将在当前用缺省的颜色(参见命令 colordef 的使用)画出线对象。注意一点的是, 用户不能在命令 line 的低级形式中使用矩阵数据。

用法 line(X,Y) 在当前的坐标轴中画出由向量 x 和 y 定义的线条。若 x 与 y 为同型的矩阵, 则对于 x, y 的每一列画出一线条。

line(X,Y,Z) 在三维空间中画出由 x, y, z 定义的线条。

line(X,Y,Z,'PropertyName',PropertyValue,...) 画出由参数 x, y, z 确定的线条, 其中对指定属性 PropertyName 设置为 PropertyValue, 其他没有指定属性用缺省值。属性 LineStyle 和 Marker 参见命令 plot。

line('PropertyName',PropertyValue,...) 对属性用相应的输入参数来设置而画出线条。这是命令 line 的低级使用形式, 此时不接受矩阵参数。除了该情形, 其他形式都接受矩阵参数。

h = line(...) 返回每一条线的线对象对应的句柄向量。

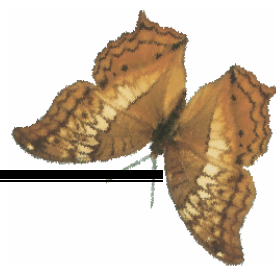
表 7-11

属性名	说明	有效属性值
定义对象的数据		
Xdata	定义线条的 x-轴坐标参量	有效值: 向量或矩阵 缺省值: [0 1]
Ydata	定义线条的 y-轴坐标参量	有效值: 向量或矩阵 缺省值: [0 1]



Zdata	定义线条的 z-轴坐标参量	有效值: 向量或矩阵 缺省值: [0 1]
定义线型与数据点标记符		
LineStyle	定义线条的类型	有效值: -, --, : , -., none 缺省值: - (实线)
LineWidth	定义线条的宽度 (以磅为单位)	有效值: 一标量 缺省值: 0.5 磅
Marker	定义标记数据点的标记符号	有效值: 13 种类型之一 缺省值: none
MarkerEdgeColor	定义标记颜色或可填充标记的边界颜色	有效值: auto、none、ColorSpec 缺省值: auto
MarkerFaceColor	定义封闭形标记的填充颜色	
MarkerSize	定义标记大小	有效值: 标量 (磅) 缺省值: 6 (磅)
控制线条的显示		
Clipping	坐标轴矩形区域是否可剪辑	有效值: on、off 缺省值: on
EraseMode	定义显示与擦除线条的方法 (对于动画显示)	有效值: normal、none、xor、background 缺省值: normal
SelectionHighlight	当线条被选中时, 是否突出显示	有效值: on、off 缺省值: on
Visible	定义线条是否可见	有效值: on、off 缺省值: on
Color	定义线条颜色	有效值: ColorSpec
对象访问的控制		
HandleVisibility	定义线条句柄对其他函数是否可见	有效值: on、off、callback 缺省值: on
HitTest	定义线条能否成为当前对象	有效值: on、off 缺省值: on
关于线条的一般信息		
Children	线条没有子对象	有效值: [] (空矩阵)
Parent	线条对象的父对象为坐标轴对象	有效值: 坐标轴句柄
Selected	是否显示线条的“选中”状态	有效值: on、off 缺省值: on
Tag	用户定义的标签	有效值: 任一字符串 缺省值: '' (空字符串)
Type	图形对象的类型 (只读类型)	有效值: 'line'
UserData	用户定义的数据	有效值: 任一矩阵 缺省值: [] (空矩阵)
与回调程序执行有关的属性		
BusyAction	定义如何处理回调中断程序	有效值: cancel、queue 缺省值: queue
ButtonDownFcn	当在线条上按下鼠标时, 定义一回调程序	有效值: 字符串 缺省值: '' (空字符串)
CreateFcn	当生成线条时, 定义一回调程序	有效值: 字符串 缺省值: '' (空字符串)
DeleteFcn	当删除线条时, 定义一回调程序	有效值: 字符串 缺省值: '' (空字符串)
Interruptible	定义回调程序是否可中断	有效值: on、off 缺省值: on (可中断)
UIContextMenu	定义与线条相关的菜单	有效值: UIContextMenu 的句柄

例 7-47



```
>>t = 0:pi/20:2*pi;
>>hline1 = plot(t,exp(t).*sin(t),'k');
>>hline2 = line(t+.06,exp(t).*sin(t),'LineWidth',4,'Color',[.8 .8 .8]);
>>set(gca,'Children',[hline1 hline2])
```

生成图形为图 7-46。

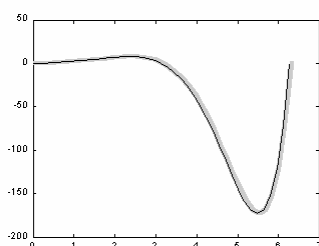


图 7-46 命令 line 画的函数图

例 7-48

生成随机直线图：

```
>>line(rand(4,2),rand(4,2),rand(4,1))
>>line(rand(1,4),rand(1,4),rand(1,4))
>>line(rand(4,1),rand(4,1),rand(4,1))
>>line(rand(2,4),rand(2,4),rand(1,4))
>>line(rand(4,2),rand(4,2),rand(4,1))
```

生成图形为图 7-47。

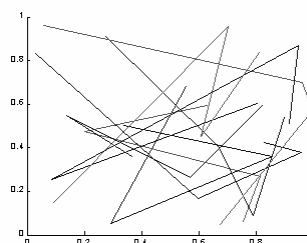


图 7-47 随机直线图

命令 3 patch

功能 生成补片图形对象。该命令为生成补片图形对象的低级图形函数。补片为一个或多个多边形，多边形的顶点为坐标中的点。用户可以指定补片的颜色与光照模式。

用法 `patch(X,Y,C)` 在当前坐标轴中增加二维带填充模式的补片。参量 `X,Y` 确定顶点的位置。若 `X,Y` 为矩阵（同型或不同型），系统按列生成多个多边形。若 `x, y` 没有定义一封闭的多边形，则命令自动地将多边形封闭。参量 `x` 与 `y` 可以定义凹的或自身交叉的多边形。可是，一个不可分隔的补片的边界自身交叉，则不能完整填充。在这种情况下，最好是将多边形分解成几个小的、自身没有交叉的多边形。参量 `c` 指定每一补片的颜色，它可以为简单的 `ColorSpec`，每面一颜色或每一顶点一颜色。若 `c` 为三维列向量，它将被认为是一直接指定的 RGB 颜色。

`patch(X,Y,Z,C)` 生成三维的补片对象。

`patch(FV)` 用结构 `FV` 生成一补片。结构 `FV` 包含这些域名 `vertices`，`faces` 和可选的 `facevertexcdata`，这些域名对应于补片的 `Vertices` 属性、`Faces` 属性、`FaceVertexCData` 属性。

`patch(...,C,'PropertyName',PropertyValue...)` 在二维(X,Y)或三维(X,Y,Z)空间中对补片指定的属性 `PropertyName` 设置为 `PropertyValue`。

`patch('PropertyName',PropertyValue...)` 对所有指定的多个属性 `PropertyName` 设置



为相应的值 **PropertyValue**。该命令形式可以使用户免除颜色的指定，因为系统将使用缺省的面颜色和边界颜色，除非用户准确地对属性 **FaceColor** 与 **EdgeColor** 进行设置。该命令形式也允许用户通过对属性 **Faces** 与 **Vertices** 的设置来代替 **x**-, **y**-与 **z**-轴的输入。

handle = patch(...) 返回命令 **patch** 生成的补片对象句柄。

说明 函数 **patch** 不象其他的高级的区域生成函数，例如函数 **fill** 或 **area**，它没有检测图形窗口与坐标轴的属性 **NextPlot** 的设置情形。它只是简单地在当前坐标轴中添加补片对象而已。

有两种指定颜色的补片属性名：

(1) **Cdata**——当指定 **x**-, **y**-与 **z**-轴坐标(**XData**,**YData**,**ZData**)时使用；

(2) **FaceVertexCData**——当指定多边形的顶点与连接矩阵时使用。

以上两个属性接受颜色数据作为索引颜色或者是真颜色(**RGB**)。其中索引颜色数据 能代表当前色图的直接索引或者代表映射到整个色图的线性数据的比例数值。

命令 4 **surface**

功能 生成面对象。该命令是生成面图形对象的低级函数。面对象为由矩阵元素的 **A(I, j)** 所在的行下标 **I** 为 **x**-坐标，所在的列下标 **j** 为 **y**-坐标，元素值为 **z**-坐标确定的点生成的空间多边形。

用法 **surface(Z)** 画出由矩阵 **z** 确定的曲面，其中 **z** 为定义在一几何矩形区域上的单值函数。

surface(Z,C) 画出颜色由 **c** 指定的、面由 **z** 指定的空间曲面。

surface(X,Y,Z) 曲面由参数 **x**, **y**, **z** 确定，颜色参数 **c=z**，因此颜色能恰当地反映曲面的高度。

surface(X,Y,Z,C) 曲面由参数 **x**, **y**, **z** 确定，颜色由参数 **c** 确定。

Surface(x,y,Z) 参数 **x** 与 **y** 为向量，若 **[m,n]=size(z)**，则要求 **length(x)=n**，**length(y)=m**，面上的点由 **(x(j),y(i),z(I,j))** 确定。

Surface(x,y,Z,C) 曲面确定如上情形，颜色由参数 **c** 确定。

surface(...'PropertyName',PropertyValue,...) 对指定的曲面属性 **PropertyName** 指定为 **PropertyValue**，对曲面进行细微控制。

h = surface(...) 返回生成面对象的句柄。

命令 5 **image**

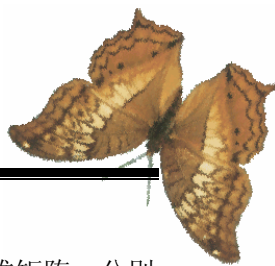
功能 显示图片对象。该命令通过对矩阵 **c** 中每一个元素（每一元素作为引用图形色图下标或直接给出 **RGB** 值）的解释而生成一个图片对象。**Image** 命令有两种使用格式：

1. 一个调用命令 **newplot** 的高级函数，可以确定在何处放置图片与坐标轴的范围为刚好围住图片；使刚生成的图片放置在坐标轴的刻度线与格栅线之上；属性 **Ydir** 设置为 **reverse**；属性 **View** 为 **[0 90]**。

2. 一个增加图片到当前坐标轴的低级命令，而没有调用命令 **newplot**，在低级使用形式中，只能对指定属性进行设置操作。

用户在命令的输入参量中可以输入属性名/属性值，结构数组，细胞数组等。

用法 **image(C)** 把 **C** 作为一图片进行显示。**C** 中的每一个元素指定了一个“图片”矩



形中的相应部分的颜色。

`image(x,y,C)` 在 (x,y) 确定的位置上画 C 的元素。其中 x , y 都为 2 维矩阵, 分别指定 x 轴与 y 轴的范围, 其效果与 `image(C)` 相同, 只不过是进行了恰当的比例缩放。

`image(x,y,C,'PropertyName',PropertyValue,...)` 该形式为指定属性名/属性值的高级使用形式, 在执行该命令之前, 先执行命令 `newplot`。

`image('PropertyName',PropertyValue,...)` 该形式为低级使用形式, 它只接受属性名/属性值的输入。

`handle = image(...)` 返回刚生成的图片对象的句柄。用户可以从上面的任何形式的调用后获得图片句柄。

例 7-49

```
>>load clown
>>image(X,'CDataMapping','scaled')
>>colormap(map)
```

图形结果为图 7-48。

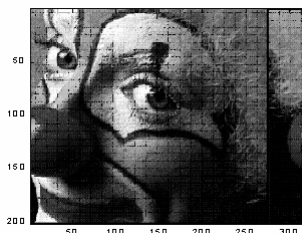


图 7-48

命令 6 uicontrol

功能 生成用户控制图形对象 (用户界面控制)。也通过该命令运行图形用户界面。当对象被选中时, 一般会执行相应的操作。系统支持多种控件, 每一种都有不同的作用:

- 校验框——当单击检验框时, 会执行一操作。该组件对于提供用户多个独立的选择是很有用的。要激活一校验框, 只需用鼠标单击该组件即可, 且选中的状态在组件上显示出来。
- 可编辑文本框——允许用户输入与修改文本文字的区域。当用户想把文字作为输入时, 可使用该组件。若一可编辑文本框有焦点, 则单击文本框的菜单栏不会执行任何操作。因此, 在单击菜单条后, 语句 `get(edit_handle,'String')` 并没有返回当前编辑框中的内容。因为系统必须执行回调函数来改变属性 `string` 的值, 即使屏幕上显示的文字已经改变。
- 框架——该组件为一封闭的、可见的、图形窗口区域。框架能使一用户图形界面中相关的控制组件能容易理解。框架没有相关的回调程序。只有控制组件能在框架中显示。框架不是透明的, 因此用户定义的组件先后顺序决定了组件是否被框架遮住或可见。属性 `Stacking order` 决定了控制组件的显示顺序: 第一个定义的组件最先显示, 后面定义的控制组件则覆盖已经存在的组件。若用户要用一框架包围一些组件, 则必须第一个定义框架。
- 列表框——显示一些项目的列表 (用命令 `string` 设置), 且允许用户选择一个或多个项目。属性 `Min` 与 `Max` 控制着选择的模式。属性 `Value` 显示可选择的项目与包含着字符串列表中项目的索引; 对于选择了多个项目则用向量表示。在任何的能改变属性 `Value` 值的、鼠标松开的操作之后, 系统 MATLAB 将马上执行列表框的回调函数。因此, 用户有必要增加一“Done”按钮, 用于推迟当要多次选择项目时的操作。在执行列表框回调函数 `Callback` 属性之前, 列表框中项目的选择有单击或双击之分, 对应于将图形窗口属性 `SelectionType` 设置为 `normal` 或 `open`。



- 弹出菜单——当组件被按下时，打开且显示一选择列表（用命令 `string` 设置）。当没有打开时，该组件显示当前的选择项。该组件对于用户想给其他用户提供一系列的互斥的选择项，又不想占用太多的区域。
- 普通按钮——当该组件被按下时，将执行一操作。要激活一按钮，只需在按钮上按下鼠标按钮。
- 单选按钮——该组件与校验框相类似，但它包含几个互斥的、而且相关的选项（例如在任意时刻，只能选择一个状态）。要激活某一单选按钮，只需在该组件上按下鼠标即可。被选中的组件同时显示出来。
- 滑块——该组件允许用户通过移动某一范围之内的滑块来输入一指定的数值。用户要移动一滑块，只需在滑块上按下鼠标不放，且在滑块方向上移动；或者是在滑槽内单击鼠标；或者是单击滑块条上的箭头。当松开鼠标后，滑块所在位置将与一数值对应。用户可以设置滑块的最大值、最小值与当前值等。
- 静态文本框——显示文本行。静态文本经常作为其他控制对象标签，以提供其他用户相关信息，或者是显示一滑块的数值。其他用户不能交互地改变静态文本，因此对于静态文本，没有相关的回调函数。
- 触发按钮——当该组件被单击且显示出它们的状态（on 或者 off）时，控制是否执行回调函数。

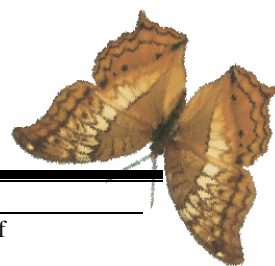
用法 `handle = uicontrol(parent)` 在父对象 `parent` 上生成一用户图形控制界面。用户界面控制对象都是图形窗口的子对象，所以当窗口中没有坐标轴时，同样可以放置控制组件于图形窗口中。

`handle = uicontrol(...,'PropertyName',PropertyValue,...)` 参量 `PropertyName` 为属性名，参量 `PropertyValue` 可为结构数组或者为单元数组，同时随意地返回刚生成的对象的句柄。当然用户可以通过命令 `set` 与 `get` 来设置与询问生成对象的属性值。

附：表 7-12 列出所有的用于命令 `uicontrol` 对象的属性名。每一属性名当作一描述该属性的链接。

表 7-12

属性名	属性名含义	属性值
控件类型与显示		
<code>BackgroundColor</code>	对象的背景颜色	有效值: <code>ColorSpec</code> 缺省值: 与系统有关
<code>Cdata</code>	显示于对象之上的真颜色图片	有效值: 矩阵
<code>ForegroundColor</code>	文本字体的颜色	有效值: <code>ColorSpec</code> 缺省值: <code>[0 0 0]</code> （黑色）
<code>SelectionHighlight</code>	当对象被选中时突出显示	有效值: <code>on</code> 、 <code>off</code> 缺省值: <code>on</code>
<code>String</code>	用户控制界面的标签，也是列表框与弹出菜单中的项目	有效值: 任意有效的字符串
<code>Visible</code>	用户界面控制是否可见	有效值: <code>on</code> 、 <code>off</code> 缺省值: <code>on</code>
关于控件对象的一般信息		
<code>Children</code>	用户界面控制界面没有子对象	



Enable	用户界面控制是否可用	有效值: on、inactive、off 缺省值: on
Parent	用户界面控制对象的父对象	有效值: 图形窗口标量句柄
Selected	对象是否为选中状态	有效值: on、off 缺省值: off
SliderStep	滑块步长尺度	有效值: 二维向量 缺省值: [0.001 0.1]
Style	用户界面控制对象的类型	有效值: pushbutton、edit、togglebutton、slider、text、radiobutton、popupmenu、listbox、frame 缺省值: pushbutton
Tag	由用户指定的对象的标记符	有效值: 任意有效字符串
TooltipString	对象的工具提示	有效值: 任意有效字符串
Type	图形对象的类型	有效值: 字符串 (只读) 缺省值: uicontrol
UserData	用户指定的数据	有效值: 矩阵
控制控件对象的位置		
Position	用户界面控制对象的大小与位置	有效值: 位置矩形 缺省值: [20 20 60 20]
Units	解释属性 position 向量的单位	有效值: pixels、inches、character、normalized、points、centimeters 缺省值: pixels
控制字体与标签		
FontAngle	字符的倾斜度	有效值: normal、italic、oblique 缺省值: normal
FontName	字体系列名称	有效值: 字符串 缺省值: 与系统有关
FontSize	字体大小	有效值: 一标量 缺省值: 与系统有关
FontUnits	字体大小单位	有效值: pixels、normalized、inches、centimeters、points 缺省值: points
FontWeight	文本字体的磅值	有效值: light、normal、demi、bold 缺省值: normal
HorizontalAlignment	标签字符串的对齐方式	有效值: left、center、right 缺省值: 决定于用户界面控制的对象
String	用户控制界面的标签, 也是列表框与弹出菜单中的项目	有效值: 字符串
控制回调函数的执行		
BusyAction	回调函数中断方式	有效值: cancel、queue 缺省值: queue
ButtonDownFcn	当按钮按下时执行的回调函数	有效值: 字符串
Callback	控制操作	有效值: 字符串
CreateFcn	在对象生成过程中执行的回调函数	有效值: 字符串
DeleteFcn	在对象删除过程中执行的回调函数	有效值: 字符串
Interruptible	回调函数中断的模式	有效值: on、off 缺省值: on
UIContextMenu	与界面控制中的对象相关的菜单 (如按下鼠标右键)	有效值: 句柄
关于当前状态的一般信息		
ListboxTop	第一个显示于列表框中的项目的索引	有效值: 标量 缺省值: [1]
Max	最大值 (与用户界面控制对象有关)	有效值: 标量



		缺省值: 与系统有关
Min	最小值 (与用户界面控制对象有关)	有效值: 标量 缺省值: 与系统有关
Value	用户界面控制对象的当前值	有效值: 标量或向量 缺省值: 与系统有关
控制组件的访问		
HandleVisibility	句柄是否可从命令窗口中与 GUIs 中访问	有效值: on、callback、off 缺省值: on
HitTest	组件是否可由鼠标单击选中	有效值: on、off 缺省值: on

命令 7 uimenu

功能 生成图形窗口的菜单中的层次的菜单与下一级子菜单。即增加新的菜单于已经存在的菜单后面, 当一菜单项被选中时, 该菜单项与它的下一级菜单也将显示。也可用该命令生成与组件相关的菜单。

用法 `handle = uimenu('PropertyName',PropertyValue,...)` 在当前图形窗口菜单条上用指定的属性 `PropertyName` 与相应的属性值 `PropertyValue` 创建一菜单, 同时将该菜单的句柄赋给 `handle`。其中两个输入参量可以是结构数组或者是单元数组。用户界面菜单的回调函数属性定义了当用户激活菜单项时, 进行的响应操作。

`uimenu('PropertyName',PropertyValue,...)` 效果同上, 但不返回句柄值。

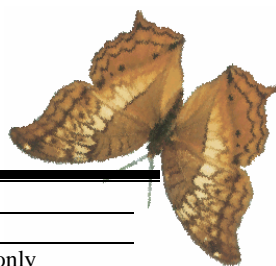
`handle = uimenu(parent,'PropertyName',PropertyValue,...)` 生成一父菜单的子菜单, 或者是生成由 `parent` 指定的相关菜单中的菜单项目。若 `parent` 不是另外的用户界面菜单对象或用户界面相关菜单对象, 而是一图形窗口, 则系统将生成该图形窗口菜单条上的新的菜单。同时将生成的菜单赋值给句柄 `handle`。

`uimenu(parent,'PropertyName',PropertyValue,...)` 效果同上, 但不返回菜单的句柄。

附: 表 7-13 列出了所有对 `uimenu` 对象有用的属性, 分别按功能进行了分类。每一属性名作为该属性描述的索引。

表 7-13

属性名	属性名描述	属性值
控制控件类型与显示		
Checked	菜单检查记号	有效值: on、off 缺省值: off
ForegroundColor	文本的颜色	有效值: ColorSpec 缺省值: 黑色[0 0 0]
Label	菜单标签	有效值: 任何字符串
Separator	分隔线模式	有效值: on、off 缺省值: off
SelectionHighlight	对象选中时是否突出显示	有效值: on、off 缺省值: on
Visible	用户界面菜单是否可见	有效值: on、off 缺省值: on
关于对象的一般信息		
Acceleratro	键盘等价字符	有效值: 任何的字符
Children	子菜单的句柄	有效值: 句柄向量
Enable	用户界面菜单是否可用	有效值: on、off 缺省值: on
Parent	用户界面菜单的父对象	有效值: 句柄



Tag	用户指定的对象标记符	有效值: 任何字符串
Type	图形对象类型	有效值: 字符串 read-only 缺省值: uimenu
UserData	用户指定数据	有效值: 任何矩阵
控制对象的位置		
Position	用户界面菜单的相对位置	有效值: 标量 缺省值: [1]
控制回调程序的执行		
BusyAction	回调程序的中断	有效值: cancel、queue 缺省值: queue
ButtonDownFcn	按钮按下回调程序	有效值: 字符串
Callback	控制操作	有效值: 字符串
CreateFcn	在对象生成期间执行的回调程序	有效值: 字符串
DeleteFcn	在对象删除期间执行的回调程序	有效值: 字符串
Interruptible	回调程序中断模式	有效值: on、off 缺省值: on
控制对象的访问		
HandleVisibility	是否可从命令行上访问图形用户界面	有效值: on、callback、off 缺省值: on
HitTest	是否可用鼠标选择	有效值: on、off 缺省值: on

7.3.2 轴的产生和控制命令

命令 1 axes

功能 创建坐标轴图形对象。该命令是创建坐标轴图形对象的低级函数命令。

用法 axes 在当前图形窗口中用缺省的属性值创建一坐标轴图形对象。

axes('PropertyName',Property Value,...) 用参量'PropertyName'指定的属性名与用参量 Property Value 指定的属性值创建一坐标轴。对于没有指定的属性名,系统则使用缺省的属性值。

axes(h) 使已经存在的坐标轴 h 成为当前的坐标轴。同时使坐标轴 h 为图形窗口中的所有子对象属性 (Children property) 的第一坐标轴,也使图形窗口的 CurrentAxes 属性为 h。当前坐标轴是图形函数 image、line、patch、surface 与 text 等命令输出图形对象的目的。

h = axes(...) 返回已经创建的坐标轴对象的句柄。

命令 2 cla

功能 清除当前坐标轴。该命令在命令窗口中执行与在回调程序中执行效果是一样的,即它不能区别由 callback 设置的属性 HandleVisibility,也就是说,当它从一回调程序中执行时,命令 cla 仅仅删除属性 HandleVisibility 为 on 的图形对象。

用法 cla 清除当前坐标轴中所有句柄为不隐藏(例如,图形对象属性 HandleVisibility 设置为 on)的图形对象。

cla reset 无条件地清除当前坐标轴中所有图形对象,且重新设置坐标轴的属性,(除了属性 Position 和 Units)。

命令 3 gca

功能 获取当前坐标轴句柄。

用法 h=gca 返回当前图形窗口中的坐标轴句柄。若坐标轴不存在,系统则生成一坐标



轴同时返回它的句柄。用户不想得到上面的结果，可以输入 `get(gcf,'CurrentAxes')`。

当前坐标轴为用户创建坐标轴以下子对象的目的。有许多图形命令可以在当前坐标轴中画出图形对象，如：`plot`，`text`，`surf` 等。改变了当前窗口，相应地改变了当前坐标轴。

7.3.3 图形句柄操作命令

命令 1 `gco`

功能 返回当前对象的句柄。“当前对象”为最后用鼠标单击的对象，除了命令 `uimenu` 之外。若鼠标没有单击到一图形对象之下的子对象，则该图形对象为“当前对象”。系统会把当前图形对象的句柄存放于图形的属性 `CurrentObject` 之中。当前图形窗口中的当前对象并非总是那些它们的回调函数，而是正在执行的对象。其他函数的回调中断函数可以改变当前对象或者甚至是当前图形窗口。一些回调函数，如生成命令 `CreateFcn`、删除命令 `DeleteFcn` 与用户界面菜单命令 `Callback` 等就没有改变当前图形窗口或者当前对象。

用法 `h = gco` 返回当前对象的句柄给 `h`。

`h = gco(figure_handle)` 返回指定窗口 `figure_handle` 中的当前对象的值。

命令 2 `get`

功能 获取对象属性。

用法 `get(h)` 返回由句柄 `h` 指定的图形对象的所有属性与相应的当前属性值；

`get(h,'PropertyName')` 返回由句柄 `h` 指定的图形对象的指定属性 `PropertyName` 的属性值。

`<m-by-n value cell array> = get(H,<property cell array>)` 返回由 `m` 个图形对象的 `n` 个属性值组成的 `m*n` 阶的细胞数组，其中 `m=length(H)`，且 `n` 为指定的属性细胞数组 `<property cell of array>` 中包含的属性名个数。

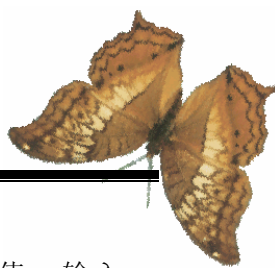
`a = get(h)` 返回一结构，其中该结构的域名为该对象的属性名，结构的域名值为相应属性的当前值。`H` 必须为标量。若用户没有指定输出参量，则系统将信息显示于屏幕之上。

`a = get(0,'Factory')` 返回所有能由用户设置的属性的缺省定义值。输出参量 `a` 为一结构数组，该结构的域名为对象的属性名，域名值为相应属性的当前值。若用户没有指定输出参量，则系统将信息显示于屏幕之上。

`a = get(0,'FactoryObjectTypePropertyName')` 返回指定对象类型的指定的属性的缺省属性值。输入参量 `FactoryObjectTypePropertyName` 为一关键字，由字符 `Factory` 与对象类型（如：`Figure`）还有属性名（如：`Color`）组成：
`FactoryFigureColor`

`a = get(h,'Default')` 返回由句柄 `h` 指定的对象的所有缺省属性值。输出参量 `a` 为一结构，该结构的域名为缺省值对应的属性名。若用户没有指定输出参量，则系统将该结构信息显示于屏幕。

`a = get(h,'DefaultObjectTypePropertyName')` 返回对象类型的指定属性的缺省属性值。输入参量 `DefaultObjectTypePropertyName` 为一关键字，该字由字符 `Default` 与对象类型名（例如：`Figure`）还有具体的属性名（例如：`Color`）组成：
`DefaultFigureColor`

**例 7-50**

若想获得定义于屏幕之上的图形对象属性 `LineWidth` 的缺省属性值，输入：
`get(0,'DefaultLineLineWidth')`

命令 3 set

功能 设置对象的属性。

用法 `set(H,'PropertyName',PropertyValue,...)` 用属性值 'PropertyValue' 设置关于用参量 `H` 标志的对象（一个或多个）的属性名 'PropertyName'（一个或多个）。`H` 可以为一句柄的向量。在这种情形下，命令 `set` 可以设置所有对象的属性值。

`set(H,a)` 用指定的属性值设置由 `H` 标志的对象的属性。其中 `a` 为一结构数组，该结构数组的域名为对象的属性名，域名值为相应属性名的属性值。

`set(H,pn,pv,...)` 对由 `H` 指定的所有对象中指定的细胞数组属性名 `pn` 设置为相应的细胞数组属性值 `pv`。

`set(H,pn,<m-by-n cell array>)` 对于每 `m` 个图形对象设置 `n` 个属性值，其中 `m=length(H)`，`n` 为包含属性名的细胞数组 `pn` 中包含的属性名个数。即允许用户对每一对象的指定的属性设置不同的属性值。

`a= set(h)` 返回句柄 `h` 中允许用户设置的属性名与可能的属性值。输出参量 `a` 为一结构数组，其域名为对象的属性名，域名值为相应的属性名对应的属性值。若没有指定输出参量 `a`，则系统自动将信息显示于屏幕，`h` 必须为标量。

`a= set(0,'Factory')` 返回那些用户可以设置缺省值的所有对象的属性名，同时显示可能的属性值，输出参量 `a` 为一结构数组，其域名为对象的属性名，域名值为相应的属性名对应的属性值，若没有指定输出参量 `a`，则系统自动将信息显示于屏幕。

`a= set(0,'FactoryObjectTypePropertyName')` 返回指定根对象 (0) 类型中指定的属性名 `ObjectTypePropertyName` 的所有可能的属性值。输入参量是由固定的关键字 `Factory`、对象类型（如 `axes`）与属性名（如 `position` 等）组成。

`a= set(h,'Default')` 返回由 `h` 标记的对象上缺省设置的值，其中 `h` 必须是标量。

`a= set(h,'DefaultObjectTypePropertyName')` 返回指定对象 `h` 的类型中指定的属性名 `ObjectTypePropertyName` 的所有可能的属性值。输入参量是由固定的关键字 `Factory`、对象类型（如 `axes`）与属性名（如 `position` 等）组成。

命令 4 reset

功能 重新设置图形对象的属性为它们的缺省值。

用法 `reset(h)` 重新设置由句柄 `h` 指定的图形对象的属性为系统为它们设置的初始值。

若 `h` 为一图形 `figure`，该命令不能重新设置属性 `Position`，`Units`，`PaperPosition` 和 `PaperUnits`；

若 `h` 为一坐标轴 `axes`，该命令不能重新设置属性 `Position` 和 `Units`。

例 7-51

`reset(gca)` % 重新设置当前坐标轴的属性。

`reset(gcf)` % 重新设置当前图形的属性。

命令 5 delete

功能 删除文件或图形对象。作为一可供选择的函数，用户可从当前目录浏览器(Current



Directory browser)中删除文件。要打开该浏览器,从 MATLAB 桌面上的 View 菜单中选择 Current Directory 命令。

用法 `delete filename` 从磁盘上删除指定的文件 `filename`。参量 `filename` 可以是绝对路径或与当前路径相关的路径名。其中可以包括通配符 (*)。

`delete(h)` 删除由句柄 `h` 指定的图形对象。该命令无条件地、直接地删除对象,甚至是图形窗口。

`delete('filename')` 这是第一种情形的函数形式。当文件名包含于字符串 `filename` 中时,使用函数形式。

例:

`delete('D:\MATLABR12\work*.m')` % 将删除指定目录上的所有.m 文件。

命令 6 findobj

功能 定位图形对象且返回它们的句柄。用户可用特定的属性值与沿着指定的层次分支来限定搜索条件。

用法 `h = findobj` 返回根对象与它的所有的子孙对象句柄。

`h = findobj('PropertyName',PropertyValue,...)` 返回属性名 `PropertyName` 具有属性值 `PropertyValue` 的所有图形对象。用户可指定一对或多对 PN 与 PV 值,对此, `findobj` 返回满足所有条件的那些对象。

`h = findobj(objhandles,...)` 限定搜索的对象为列表于 `objhandles` 中的对象与它们子孙对象。

`h = findobj(objhandles,'flat','PropertyName',PropertyValue,...)` 限定搜索对象为 `objhandles` 中列出的对象,而不包含它们的子孙对象。

7.3.4 图形窗口的控制命令

命令 1 subplot

功能 生成与控制多个坐标轴。把当前图形窗口分隔成几个矩形部分,不同的部分是按行方向以数字进行标号的。每一部分有一坐标轴,后面的图形输出于当前的部分中。

用法 `subplot(m,n,p)` 将一图形窗口分成 `m*n` 个小窗口,在第 `p` 个小窗口中创建一坐标轴。则新的坐标轴成为当前坐标轴。若 `p` 为一向量,则创建一坐标轴,包含所有罗列在 `p` 中的小窗口。

`subplot(h)` 使句柄 `h` 对应的坐标轴称为当前的,用于后面图形的输出显示。

`subplot('Position',[left bottom width height])` 在由 4 个元素指定的位置上创建一坐标轴。位置元素的单位为归一化单位。

`h = subplot(...)` 返回一新坐标的句柄于 `h`。

命令 2 hold

功能 保持当前图形窗口中的图形。该命令是决定是否在当前坐标轴中只能增加新的图形对象还是覆盖原有图形对象。测试保持状态命令为 `ishold`。该命令可以设置当前坐标轴与当前图形的属性 `NextPlot`。若一图形窗口中有多个坐标轴,则每个坐标轴有自己的保持状态。

用法 `hold on` 保留当前图形与当前坐标轴的属性值,后面的图形命令只能在当前存在的坐标轴中增加图形,即设置当前坐标轴属性 `NextPlot` 为 `add`。当必要的时候



候,坐标轴的一些属性在增加新图时还是要进行相应的改变。例如,当新图形的数据范围超出了当前坐标轴的范围,则命令会自动地改变坐标轴的范围,使能显示新图形。

hold off 在画新图形之前,重新设置坐标轴的属性为缺省值。**off** 是命令 **hold** 命令的缺省值。设置当前坐标轴的属性 **NextPlot** 为 **replace**。

hold 在 **on** 与 **off** 之间转换。即在增加图形与覆盖图形之间切换。当坐标轴不存在时,则生成一坐标轴。同时使当前坐标轴属性 **NextPlot** 在 **add** 与 **replace** 之间切换。

命令 3 gcf

功能 获得当前图形窗口的句柄。

用法 **h = gcf** 返回当前图形窗口的句柄。当前窗口为由命令 **plot**、**title** 与 **surf** 等得到的结果。若不存在图形窗口,则系统自动地生成一个,并返回它的句柄。若用户想当图形窗口不存在时,也不创建新的,则输入: **get(0,'CurrentFigure')**

命令 4 clf

功能 清除当前图形窗口。该命令在命令窗口中执行与在回调程序中执行效果是一样的,即它不能区别由 **callback** 设置的属性 **HandleVisibility**,也就是说,当它从一回调程序中执行时,命令 **clf** 仅仅删除属性 **HandleVisibility** 为 **on** 的图形对象。

用法 **clf** 清除所有当前图形窗口与窗口中的所有那些句柄为不隐藏(例如它们的属性 **HandleVisibility** 为 **on**)的图形对象。

clf reset 无条件地清除当前图形窗口中所有的图形对象,且重新设置所有图形窗口属性为缺省值,除了属性 **Position**, **Units**, **PaperPosition**, **PaperUnits**。

命令 5 close

功能 删除指定的图形窗口。

用法 **close** 删除当前的图形窗口。

close(h) 删除由句柄 **h** 指定的图形窗口。若 **h** 为一向量或矩阵,则 **close** 全部删除其中每一分量指定的图形句柄。

close name 删除指定名字 **name** 的窗口。

close all 删除所有没有隐藏的图形。

close all hidden 删除所有具有隐藏的图形。

status = close(...) 若成功地删除了指定的对象则返回 **status=1**,否则返回 **0**。

命令 6 newplot

功能 做好开始画新图形对象的准备。在高级图形 **m**-文件的开始使用该命令,用于确定在哪一个图形窗口与坐标轴中输出图形。调用命令 **newplot** 能改变当前窗口与坐标轴。基本上,当要在已经存在的窗口与坐标轴中画图,有三个选项可选:

1. 没有改变任何属性与删除任何对象,直接在当前坐标轴中增加新的图形对象;
2. 在画图形的对象之前,删除所有存在于当前坐标轴中的,句柄为非隐藏的对象;
3. 在画图形的对象之前,无条件删除所有的存在于当前坐标轴中的对象(不管句柄是否为隐藏),同时设置大部分的属性为缺省值;
4. 首先,**newplot** 读取当前图形的属性 **NextPlot** 的属性值(关于该属性的含义参见 **figure** 或 **axes** 的属性表),再执行相应的动作;



5. 然后, newplot 确定在哪个窗口中画图, 它读取当前图形的属性 NextPlot 的属性值, 执行相应的操作。

用法 newplot 画好图形窗口与坐标轴, 后面的图形命令就可以在该坐标轴内画图。

h = newplot 效果如上, 且返回当前坐标轴的句柄给 h。

7.4 颜色与光照模式命令

7.4.1 颜色控制命令

命令 1 colormap

功能 设置或获取当前色图。色图为一个 $m \times 3$ 的、元素在 0 到 1 之间的实数的矩阵, 每一行为定义一个颜色的 RGB 向量。色图矩阵的第 k 行定义了第 k 个颜色, 其中 $\text{map}(k,:) = [r(k) \ g(k) \ b(k)]$ 指定了组成该颜色中红色、绿色、蓝色的强度。

用法 colormap(map) 通过矩阵 map 设置色图。若矩阵 map 中的元素不在 [0 1] 区间之内, 则返回一个错误。在目录 color 中的 m-文件能够生成许多色图, 每一个 m-文件能够接受颜色数作为函数参数, 例如命令 colormap(hsv(64)) 生成了有 64 种颜色的 hsv 色图。若用户没有指定颜色数, 例如命令 colormap(hsv), 生成与当前色图中颜色数相同的 hsv 色图。MATLAB 支持的色图见表 7-14。

表 7-14

色图名称	包含的颜色范围
Cool	青蓝和洋红的色度
Bone	带一点蓝色的灰度
Flag	交替为红色、白色、蓝色和黑色
Jet	Hsv 的一种变形 (以蓝色开始和结束)
Copper	线性铜色度
Hsv	色彩饱和度 (以红色开始和结束)
Hot	从黑色到黄色到白色
Gray	线性灰度
Pink	粉红的彩色度
Prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
Lines	线性色图
White	全白色图
Colorcube	增强立方色图
Autumn	红色黄色阴影色图
Spring	洋红黄色阴影色图
Summer	绿色黄色阴影色图
Winter	蓝色绿色阴影色图

例 7-52

colormap('default') 设置当前色图为缺省色图。

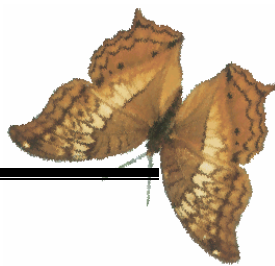
cmap = colormap 获取当前色图矩阵。

命令 2 bone

功能 生成带淡蓝色的灰度刻度化的色图。

用法 bone(m) 返回一个阶数为 $m \times 3$ 的包含 “bone” 的色图。

bone 返回一个与当前色图行数相同的色图。

**命令 3 cool**

功能 生成带阴影的青色和品红的色图。

用法 cool(m) 返回一个阶数为 $m \times 3$ 的包含“cool”的色图。

cool 返回一个与当前色图行数相同的色图。

命令 4 copper

功能 生成线性铜色色图。

用法 copper(m) 返回一个阶数为 $m \times 3$ 的包含“copper”的色图。

copper 返回一个与当前色图行数相同的色图。

命令 5 flag

功能 生成一个颜色顺序为红、白、兰、黑的色图。

用法 flag(m) 返回一个阶数为 $m \times 3$ 的包含“flag”的色图。增加 m 的值，会增加色图的颗粒程度。

flag 返回一个与当前色图函数相同的色图。

命令 6 gray

功能 生成一个线性灰度化的色图。

用法 gray(m) 返回一个阶数为 $m \times 3$ 的包含灰度化的色图。

gray 返回一个与当前色图函数相同的色图。

命令 7 hot

功能 生成一个颜色顺序为黑、红、黄、白的色图。

用法 hot(m) 返回一个阶数为 $m \times 3$ 的包含“hot”的色图。

hot 返回一个与当前色图函数相同的色图。

命令 8 hsv

功能 生成一个包含色度-饱和度值的色图。一个 hsv 色图包含各种饱和和色度颜色的色度的成分。其颜色从红色到黄色、绿色、青色、蓝色、品红，最后返回红色。该色图对于显示周期函数很有用处。

用法 hsv(m) 返回一个阶数为 $m \times 3$ 的包含 hsv 的色图。

hsv 返回一个与当前色图函数相同的色图。

命令 9 jet

功能 不同于 hsv 色图的另外一种色图。

用法 jet(m) 返回一个阶数为 $m \times 3$ 的，与 hsv(m) 不同的色图，用于显示 NCSA 流体激光图片。

jet 返回一个与当前色图函数相同的色图。

命令 10 pink

功能 生成一个带柔和阴影粉红色图。

用法 pink(m) 返回一个阶数为 $m \times 3$ 的包含“pink”的色图。

pink 返回一个与当前色图函数相同的色图。

命令 11 prism

功能 生成一个三棱镜色图。如同 hsv 色图一样，prism 色图中的颜色使用顺序是一样的，不同的是，命令 prism 重复使用它的六中颜色，而命令 hsv 是连续地变换它的颜色。



用法 prism(m) 返回一个阶数为 $m \times 3$ 的包含六种循环使用的颜色：红色、橙色、黄色、绿色、蓝色、紫色。

prism 这种没有任何输入输出参量的形式，改变当前坐标轴中的线对象的颜色为三棱镜中的颜色。

7.4.2 色图控制命令

命令 1 brighten

功能 增亮或变暗色图。

用法 brighten(beta) 增亮或变暗当前的色图。若 $0 < \text{beta} < 1$ ，则增亮色图；若 $-1 < \text{beta} < 0$ ，则变暗色图。改变的色图将代替原来的色图，但本质上是相同的颜色。

brighten(h,beta) 对指定的句柄对象 h 中的子对象进行操作。

newmap = brighten(beta) 该命令没有改变当前图形的亮度，而是返回变化后的色图给 newmap。

newmap = brighten(cmap,beta) 该命令没有改变指定色图 cmap 的亮度，而是返回变化后的色图给 newmap。

命令 2 colorbar

功能 显示能指定颜色刻度的颜色条。且调整当前坐标轴，以适应当前的颜色条。

用法 colorbar 更新最近生成的颜色条。或若当前坐标轴没有一颜色条，则在右边显示一垂直的颜色条。

colorbar('vert') 增加一垂直的颜色条到当前的坐标轴。

colorbar('horiz') 增加一水平的颜色条到当前的坐标轴。

colorbar(h) 用坐标轴 h 来生成一颜色条。若坐标轴的宽度大于高度，则颜色条是水平放置的。

h = colorbar(...) 返回一颜色条句柄 h，该句柄是一坐标轴对象。

colorbar(...,'peer',axes_handle) 生成一与坐标轴 axes-handle 有关的颜色条，代替当前的坐标轴。

命令 3 contrast

功能 提高灰度色图的对比度。该命令可以增强图像的对比度。

用法 cmap = contrast(X) 返回一灰度色图，该色图与当前色图有相同的维数。参量 cmap 为生成的灰度色图。

cmap = contrast(X,m) 返回维数为 $m \times 3$ 的灰度色图 cmap。

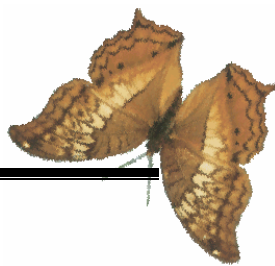
例 7-53

```
>>load clown;  
>>cmap = contrast(X);  
>>image(X);  
>>colormap(cmap);
```

命令 4 rgbplot

功能 画出色图。

用法 rgbplot(cmap) 画出维数为 $m \times 3$ 的色图矩阵 cmap 的每一列，矩阵的第一列为红色强度，第二列为绿色强度，第三列为蓝色强度。

**命令 5 diffuse**

功能 漫反射率。

用法 $R = \text{diffuse}(N_x, N_y, N_z, S)$ 返回曲面的漫反射率向量 $[N_x, N_y, N_z]$, S 为一三维向量, 用于定义光源的方向; S 也可以为球面坐标系中的二维向量 $[\text{Theta}, \text{Phi}]$ 。

Lambert 定律: $R = \cos(\text{PST})$, 其中 PST 为曲面法线与光源方向之间夹角。

命令 6 specular

功能 镜面反射率。

用法 $R = \text{specular}(N_x, N_y, N_z, S, V, \text{spread})$ 返回一曲面的镜面反射率向量 $[N_x, N_y, N_z]$, 向量参量 S 与 V 分别用于指定光源位置与观察点的位置。它们可以为三维直角坐标系向量 $[x, y, z]$ 或者为二维球面向量 $[\text{Theta}, \text{Phi}]$ 。当标准向量的方向为 $(S+V)/2$, 则镜面的高光效果最强。第六个参量 spread 为镜面反射扩散系数。

命令 7 surf

功能 三维带光照模式的阴影图。图形的色泽取决于曲面的漫反射、镜面反射与环境光照模式。

用法 $\text{surf}(\dots)$ 效果与命令 $\text{surf}(\dots)$ 基本上一样, 除了它受光源影响的曲面之外。

$\text{surf}(Z)$ 、 $\text{surf}(X, Y, Z)$ 、 $\text{surf}(Z, S)$ 、 $\text{surf}(X, Y, Z, S)$ 、 $\text{surf}(X, Y, Z, S, K)$ 这些都是有效的使用形式。若参数中有 S , 则为一三维向量 $[S_x, S_y, S_z]$, 用于指定光源的方向。 S 也可视为点坐标系下的二维向量 $[\text{AZ}, \text{EL}]$ 。 S 的缺省值为从当前观察方向逆时针旋转 45 度。使用命令组 `cla; hold on; view(AZ, EL); surf(...); hold off` 等可画出视角方向为 (AZ, EL) 的带光照模式的曲面图。第五参数 $K=[k_a, k_d, k_s,$

$\text{spread}]$ 指定环境光、漫反射光、镜面反射光、扩散系数等的强弱。

$\text{surf}(\dots, 'light')$ 用 LIGHT 对象生成一带颜色的、带光照模式的曲面。该命令可以生成与用缺省光照模式不同效果的曲面。

$\text{surf}(\dots, 'cdata')$ 指定的曲面的反射光的颜色为 $cdata$ 。

$H = \text{surf}(\dots)$ 返回曲面与光源的句柄。