第3章 符号运算

符号运算是指利用数学定理和恒等式，通过推理和演绎，分析化简表达式，将复杂表达式变为形式简单的恒等表达式。利用符号运算，可以避免计算过程中产生误差。MATLAB中的符号计算功能是由Maple独立引擎提供的，利用这个内置的Maple符号计算引擎，可以进行各种针对符号对象或解析式的数学运算，如微积分运算，代数、微分方程求解，线性代数和矩阵运算，以及Laplace变换、Fourier变换和Z变换。

3.1 符号运算基础

符号运算与数值计算一样，都是科学研究中的重要内容。运用符号运算，可以轻松解决许多公式和关系式的推导问题。

3.1.1 创建符号对象

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

1. sym函数

sym函数用来创建单个符号量，调用格式为

```
符号量名 = sym('符号字符串')
```

该函数可以建立一个符号量，符号字符串可以是常量、变量、函数或表达式。

【例3-1】 利用sym函数创建符号变量，完成对方程组求解。

```
>> a = sym('a');
>> b = sym('b');
>> x = sym('x');
>> y = sym('y');
>> A = a * x - b * y == 1;
>> B = a * x + b * y == 3;
>> [x,y] = solve(A,B,x,y)
 x =
 2/a
 y =
 1/b
```

【例3-2】 创建符号变量，求复数表达式 $z=x+i*y$ 的共轭复数。

```
>> x = sym('x','real');
>> y = x + i * y;
>> x = sym('x','real');
```

```
>> y = sym('y','real');
>> z = x + i * y;
>> conj(z)
   ans =
   x - y * i
```

2. syms 函数

syms 函数可以在一条语句中定义多个符号变量，调用格式为

```
syms 符号变量名 1 符号变量名 2 …符号变量名 n
```

用这种格式定义符号变量时不要在变量名上加字符串分界符(')，变量间用空格而不要用逗号分隔。在数学表达式中，一般习惯于使用排在字母表中前面的字母作为变量的系数，而用排在后面的字母表示变量。

例如，$f=ax^2+bx+c$，表达式中的 a、b、c 通常被认为是常数，用作变量的系数；而将 x 看作自变量。若在 MATLAB 中表示上述表达式，首先用 syms 函数定义 a、b、c、x 为符号对象。在进行导数运算时，由于没有指定符号变量，则系统采用数学习惯来确定表达式中的自变量，默认 a、b、c 为符号常数，x 为符号变量，即对函数 f 求导为 $\mathrm{d}f/\mathrm{d}x$。

3.1.2 创建表达式

含有符号对象的表达式被称为符号表达式，一个符号表达式应该由符号变量、函数、算术运算符组成，符号表达式的书写格式与数值表达式相同。表 3-1 为符号表达式和 MATLAB 表达式的对照。

表 3-1 符号表达式和 MATLAB 表达式的对照

符号表达式	MATLAB 表达式
$y=\dfrac{1}{\sqrt{2x}}$	y='1/sqrt(2 * x)'
$\cos(x^2)-\sin(2x)$	'cos(x^2)-sin(2 * x)'
$\dfrac{\mathrm{e}^{x^3}}{\sqrt{1-x}}$	'exp(x^3)/sqrt(1-x)'
$\dfrac{1}{2x^n}$	'1/(2 * x^n)'

有 3 种建立符号表达式的办法：

(1) 利用单引号来生成符号表达式。

(2) 用 sym 函数建立符号表达式。

(3) 使用已经定义的符号变量组成符号表达式。

将表达式中的自变量定义为符号变量后，赋值给符号函数名，即可生成符号函数。例如有以下数学表达式：

$$(ax^2+by^2)/c^2$$

其用符号表达式生成符号函数 fxy 的过程为

```
>> syms a b c x y                          % 定义符号运算量
fxy = (a * x^2 + b * y^2)/c^2              % 生成符号函数
```

生成符号函数 fxy 后，即可用于微积分等符号计算。

【例 3-3】 符号函数 fxy=(ax²+by²)/c²，分别求该函数对 x、y 的导数和对 x 的积分。

```
>> syms a b c x y                          % 定义符号变量
>> fxy = (a * x^2 + b * y^2)/c^2;          % 生成符号函数
```

```
>> diff(fxy,x)                % 符号函数 fxy 对 x 求导数
ans = 2 * a * x/c^2
>> diff(fxy, y)               % 符号函数 fxy 对 y 求导数
ans = 2 * b * y/c^2
>> int(fxy, x)                % 符号函数 fxy 对 x 求积分
ans = 1/c^2 * (1/3 * a * x^3 + b * y^2 * x)
```

3.1.3 基本操作

1. 使用 sym 函数创建

sym 函数可以创建符号矩阵,用法如下:

```
A = sym([])
```

【例 3-4】 利用 sym 函数直接创建符号矩阵。

```
A = sym('[a , 2 * b ; 3 * a , 0]')
A =
    [ a, 2 * b]
    [3 * a, 0]
```

注意:符号矩阵的每一行的两端都有方括号,这是与 MATLAB 数值矩阵的一个重要区别。

2. 基于字符串创建

(1) 用字符串直接创建矩阵。

(2) 模仿 MATLAB 数值矩阵的创建方法。

需保证同一列中各元素字符串有相同的长度。

【例 3-5】 模仿数值矩阵的方式创建符号矩阵。

```
A = ['[ a,2 * b]'; '[3 * a, 0]']
A =
    2×8 char 数组
    [ a, 2 * b]
    [3 * a, 0]
```

3. 符号矩阵的修改

修改符号矩阵有两种方式,利用光标键移到指定位置直接修改或是利用指令修改,下面主要介绍利用指令修改。

指令修改运用了 subs 函数,例如:

```
A1 = subs(A, 'new', 'old')
```

【例 3-6】 利用 subs 函数修改符号矩阵。

```
>> A = sym('A', [2,2])
B = sym('B', [2,2])
A =
    [ A1_1, A1_2]
    [ A2_1, A2_2]
B =
    [ B1_1, B1_2]
    [ B2_1, B2_2]
>> A44 = subs(A, A(1,1), B)
A44 =
    [ B1_1, B1_2, A1_2, A1_2]
    [ B2_1, B2_2, A1_2, A1_2]
    [ A2_1, A2_1, A2_2, A2_2]
    [ A2_1, A2_1, A2_2, A2_2]
```

3.1.4 相关运算符

MATLAB 中为符号运算提供了多种多样的运算符，如表 3-2 所示。

表 3-2 符号运算中的运算符

符号	符号用途说明
+	加
-	减
.*	点乘
*	矩阵相乘
^	矩阵求幂
.^	点幂
\	左除
/	右除
.\	点左除
./	点右除
kron	张量积
,	分隔符
;	(a) 写在表达式后面时，运算后不显示计算结果 (b) 在创建矩阵的语句中指示一行元素的结束，例如 m=[x y z;i j k]
:	创建向量的表达式分隔符，如 x=a:b:c a(:,j)表示 j 列的所有行元素；a(i,:)表示 i 行的所有列元素
[ ]	创建数组、向量、矩阵或字符串(字母型)
{ }	创建单元矩阵或结构
%	注释符，特别当编写自定义函数文件时，紧跟 function 后的注释语句，在使用 help 函数名时会显示出来
'	(a) 用于定义字符串 (b) 向量或矩阵的共轭转置符
.'	一般转置符
…	表达式换行标记，表示表达式继续到下一行
=	赋值符号
==	等于关系运算符
<，>	小于，大于关系运算符
&	逻辑与
|	逻辑或
~	逻辑非
xor	逻辑异或

3.1.5 确定自变量

MATLAB 中的符号可以表示符号变量和符号常量，symvar 可以帮助用户查找一个符号表达式中的符号变量。其调用方法如下：

symvar(expr)：确定表达式 expr 中的所有符号为自变量。

symvar(expr,n)：确定表达式 expr 中离字母 x 最近的 n 个自变量。

【例 3-7】 利用 symvar 确定表达式中的自变量。

```
>> syms a x y z t
>> symvar (sin(pi * t))
ans =
```

```
    t
>> symvar(x + i * y - j * z,1)
ans =
    x
>> symvar (x + i * y - j * z,2)
ans =
    [ x, y]
>> symvar (x + i * y - j * z,3)
ans =
    [ x, y, z]
```

注意,MATLAB 按离字母 x 最近原则确定默认变量。

3.2 符号表达式运算

符号表达式可以进行多种运算(如基本的四则运算),也可进行表达式求值、数值转换及变量替换等。

3.2.1 提取分子和分母

如果表达式是一个有理分式(两个多项式之比),或是可以展开为有理分式(包括那些分母为 1 的分式),可以利用 numden 将分子或分母提取出来。

【例 3-8】 利用 numden 提取分子分母。

```
>> [n,d] = numden(sym(4/5))
n =
    4
d =
    5
```

这个表达式是符号数组,numden 返回两个新数组 n 和 d,其中 n 是分子数组,d 是分母数组。如果采用 s=numden(f)形式,numden 仅把分子返回到变量 s 中。

numden 也可以化简分数表达式:

```
>> syms x y;
>> f = x/y + y/x;
>> [n,d] = numden(f)
n =
    x^2 + y^2
d =
    x * y
```

3.2.2 复合函数运算

在 MATLAB 中,符号表达式的复合函数运算主要是通过 compose 函数来实现的,该函数的调用格式如下:

compose(f,g):返回复合函数 f(g(x)),此处(f(x),g)=g(y)。

compose(f,g,x,z):返回自变量为 z 的复合函数 f(g(z)),并且使 x 成为 f 函数的独立变量。

【例 3-9】 compose 复合运算函数示例。

```
>> A = pi
A =
   3.1416
>> formatSpec = " % .8f"
formatSpec =
   " % .8f"
>> str = compose(formatSpec,A)
```

```
str =
    "3.14159265"
>> A = [pi exp(1)]
A =
   3.1416 2.7183
>> formatSpec = "The value of pi %.2e;the value of e is %.5f.";
>> str = compose(formatSpec,A)
str =
    "The value of pi 3.14e+00;the value of e is 2.71828."
```

3.2.3 数值转换

1. 数据类型转换函数

利用数据类型转换函数可以将数值转换为另一种数据类型的数值形式，常用的数据类型转换函数如表 3-3 所示。

表 3-3 数据类型转换函数

函 数 名	作 用	函 数 名	作 用
logical	数值转换为逻辑值	uint32	转换为 32 字节数
char	转换为字符串数组	int64	转换为 64 字节整型数
int8	转换为 8 字节整型数	uint64	转换为 64 字节数
uint8	转换为 8 字节数	single	转换为单精度浮点数
int16	转换为 16 字节整型数	double	转换为双精度浮点数
uint16	转换为 16 字节数	cell	转换为细胞数组
int32	转换为 32 字节整型数	struct	转换为结构体类型

【例 3-10】 利用转换函数转换符号常量。

```
>> a = 3.8495;
>> A = 6 * a + 2^(2 * a);
>> f = sym('A');
>> m = eval(f)
m =
  230.8895
>> int8(m)
ans =
    int8
    127
>> logical(m)
ans =
    logical
       1
```

2. sym2poly

sym2ploy 可以将符号表达式转换为数值多项式的系数向量，且系数从高到低依次排列。

【例 3-11】 使用 sym2poly 函数显示数值多项式的系数向量。

```
>> syms x;
>> c = sym2poly(x^3 - 2 * x - 5)
c =
   1     0    -2    -5
```

3. poly2sym

poly2sym 与 sym2poly 相反，可以将数值多项式的系数向量转换为符号表达式。

```
a = [5 3 7];
poly2sym(a)
```

```
ans =
5 * x^2 + 3 * x + 7
```

4. eval

MATLAB 中的 eval 函数可以计算符号表达式的具体值。

【例 3-12】 计算符号表达式 k * 2+2^m 的值。

```
>> k = sym('5');
>> m = sym('7');
>> f = sym(k * 2 + 2^m);
>> r = eval(f)
r =
    138
```

3.2.4 变量替换

MATLAB 中的 subs 函数用于实现变量的替换,在处理复杂函数方程式的时候会使计算更简便。

subs(S,old,new):用 new 替换 S 中的 old 变量,old 必须是 S 中的符号变量。

subs(S,new):用 new 替换 S 中的自变量。

【例 3-13】 subs 函数用于实现变量的替换。

```
>> syms a m n w;
>> f = 2 + 3^a;
>> subs(f,a,m^2 + 5 * n + w)
   ans =
        3^(m^2 + 5 * n + w) + 2
```

3.2.5 表达式的相互转换

利用函数 sym 可以将数值表达式转换为它的符号表达式。

```
>> sym(1.5)
   ans =
      3/2
>> sym(4.45)
   ans =
      89/20
```

函数 numeric 或 eval 可以将符号表达式转换成数值表达式。

```
>> phi = '(1 + sqrt(5))/2'
   phi =
      (1 + sqrt(5))/2
>> numeric(phi)
   ans =
      1.6180
>> p = '(1 + 2^3)/2'
   p =
      (1 + 2^3)/2
>> eval(p)
   ans =
      4.5000
```

3.2.6 化简与格式化

MATLAB 提供了多种函数来实现对符号运算表达式进行化简,如 factor(因式分解)、collect(合并同类项)、horner(将多项式分解为嵌套形式)、expand(展开表达式为多项式、指数

函数、对数函数、三角函数)、simplify(化简一个表达式)、simple(将表达式化简到最简形式)。

1. 因式分解函数 factor

factor(x),若 x 可分解时,返回分解后的表达式,否则返回原 x。

【例 3-14】 利用 factor 分解表达式 x^2+4*x+5。

```
>> f = sym('x^2 + 4 * x + 5');
>> factor(f)
   ans =
        x^2 + 4 * x + 5
        >> syms a b x y;
        >> A = a^3 - b^3;
        >> factor(A)
   ans =
        (a - b) * (a^2 + a * b + b^2)
>> syms x; f = x^6 + 1;
>> factor(f)
  % factor 也可用于正整数的分解
>> s = factor(100)
   2     2     5     5
>> factor(sym('12345678901234567890'))
   [ 2, 3, 3, 5, 101, 3541, 3607, 3803, 27961]
```

2. 合并同类项函数 collect

collect(S):将 S 中相同次幂的项合并,S 可以是表达式也可以是符号矩阵。

collect(S,v):将 S 中 v 的相同幂次的项进行合并。

【例 3-15】 使用 collect 函数实现合并同类项。

使用 collect 函数的第一种形式合并同类项:

```
>> f = sym('(x^2 + 2 * x) * (x + 2)');
>> collect(f)
  ans =
       x^3 + 4 * x^2 + 4 * x
```

使用 collect 函数的第二种形式合并同类项:

```
>> syms x y;
>> f = (y^3 + 2 * x) * (x + 5);
>> collect (f,x)
   ans =
        2 * x^2 + (y^3 + 10) * x + 5 * y^3
>> syms x y;
>> f = x^2 * y + y * x - x^2 + 2 * x ;
>> collect(f)
>> collect(f,y)
   ans =
       (y - 1) * x^2 + (y + 2) * x
   ans =
       (x^2 + x) * y + 2 * x - x^2
```

3. 多项式分解函数 horner

horner(S),S 是符号多项式矩阵,horner 函数可以将每个多项式转换成嵌套形式。

【例 3-16】 分解多项式 f=5*x^4+3*x^2-x。

```
>> syms x
>> f = 5 * x^4 + 3 * x^2 - x;
>> horner(f)
  ans =
```

```
        x*(x*(5*x^2 + 3) - 1)
>> syms x;
>> f = x^4 + 2*x^3 + 4*x^2 + x + 1;
>> g = horner(f)
   g =
      x*(x*(x*(x + 2) + 4) + 1) + 1
```

4. 展开表达式函数 expand

expand(S),若 S 是多项式,则展开为相应的形式;若 S 是三角函数、指数函数或对数函数,则根据要求展开成相应形式。

【例 3-17】 用 expand 函数展开多项式。

```
>> syms x,y;
>> f = (5*x + 4*y + 3)^2;
>> expand(f)
    ans =
         25*x^2 + 40*x*y + 30*x + 16*y^2 + 24*y + 9
    s = (-7*x^2 - 8*y^2)*(-x^2 + 3*y^2)
    expand(s)                          %对 s 展开
    ans =
         7*x^4 - 13*x^2*y^2 - 24*y^4
%多项式展开
>> syms x;
>> f = (x + 1)^6;
>> expand(f)
  ans =
       x^6 + 6*x^5 + 15*x^4 + 20*x^3 + 15*x^2 + 6*x + 1
%三角函数展开
>> syms x y; f = sin(x + y);
   expand(f)
   ans =
        cos(x)*sin(y) + cos(y)*sin(x)
```

5. 化简表达式函数 simplify

simplify(S),表达式 S 可以是多项式,也可以是符号表达式矩阵。

【例 3-18】 用 simplify 函数化简多项式。

(1) 化简 sin(x)^2+cos(x)^2+2*sin(x)*cos(x)。

```
>> f = sym('sin(x)^2 + cos(x)^2 + 2*sin(x)*cos(x)');
>> simplify(f)
   ans =
        sin(2*x) + 1
```

(2) 化简 log(2*x/y)和(-a^2+1)/(1-a)。

```
>> syms x y a
>> s = log(2*x/y);
>> simplify(s)
   ans =
        log((2*x)/y)
>> s = (-a^2 + 1)/(1 - a)
>> simplify(s)
   ans =
        a + 1
>> syms x;
>> f = sin(x)^2 + cos(x)^2 ;
>> simplify(f)
   ans =
        1
```

3.2.7 反函数

在 MATLAB 中,可以使用 finverse 计算反函数。

g=finverse(f)返回符号函数 f 的反函数 g。其中,f 是一个符号函数表达式,其变量为 x。求得的反函数 g 是一个满足 g(f(x))=x 的符号函数。

```
>> syms x;
>> f = sym(2/sin(x));
>> finverse(f)
   ans =
        asin(2/x)
```

g=finverse(f,v)返回自变量 v 的符号函数 f 的反函数。求得的反函数 g 是一个满足 g(f(v))=v 的符号函数。当 f 包含不止一个符号变量时,往往调用这个格式。

当 finverse 求得的解不唯一时,MATLAB 会给出警告。

```
>> syms x;
>> f = sym(x^2 + 1);
>> finverse(f)
   ans =
        ( - 1 + x)^(1/2)
```

3.2.8 替换函数

subs(s)用赋值语句中给定值替换表达式中所有同名变量。

subs(s, old, new)用符号或数值变量 new 替换符号表达式 s 中的符号变量 old。

常用格式有以下 3 种:

```
subs(f)      %求符号表达式 f 的值
subs(f,a)    %用 a 替换 f 中的默认变量 x 并求值
subs(f,x,a)  %用 a 替换 f 中的指定变量 x 并求值
```

【例 3-19】 表达式替换函数演示。

(1) 将表达式 x^2+y^2 中的 x 取值为 2:

```
>> syms x y;
   f = x^2 + y^2;
>> subs(f,x,2)
   ans =
        y^2 + 4
```

(2) 同时对两个或多个变量取值求解:

```
>> syms x y ;
   f = x^2 + y^2;
>> subs(f,[x,y],[1,2])
   ans =
        5
```

3.3 运算精度

MATLAB 提供了 3 种计算精度:浮点运算的数值算法、精确运算的符号算法和可控精度的算法。

1. 浮点运算的数值算法

浮点运算的数值算法是运算速度最快的运算方法,由于在计算机中以二进制进行存储,计

算时取近似值,不可避免地会产生误差。

【例 3-20】 浮点运算的数值算法样例。

```
>> sym a;
>> a = 2/3 + 4/7
   a =
      1.2381
```

2. 精确运算的符号算法

精确运算速度较慢,但精度高。

【例 3-21】 精确运算的符号算法样例。

```
>> a = sym(2/3 + 4/7)
   a =
      26/21
```

3. 可控精度的算法

可控精度的算法通过规定有效数字位数控制精度,位数不同,精度也不同。

digits(n)规定参加运算有效数字的位数,MATLAB 默认值为 32。

vpa(s)在 digits(n)控制下计算指定精度的 s,如果 n 未指定则默认为 32。

【例 3-22】 可控精度的算法样例。

```
>> syms a b c
>> a = 1/3 + 5/7;
>> b = pi;
>> c = 3.7878882;
>> d = sym(4/9);
>> f1 = vpa(a + b)
   f1 =
        4.189211701208840565868740668520 3
>> f2 = vpa(a + c)
   f2 =
        4.835507247619047710429640574147 9
>> f3 = vpa(a + d)
   f3 =
        1.492063492063492063492063492063 5
>> digits(20)
>> f4 = vpa(a + b)
   f4 =
        4.1892117012088405659
>> f5 = vpa(a + c)
   f5 =
        4.8355072476190477104
>> f6 = vpa(a + d)
   f6 =
        1.4920634920634920635
```

3.4 符号矩阵运算

在进行符号矩阵的计算时,很多方面在形式上与数值计算是相同的,不必再去重新学习一套关于符号运算的新规则。这里介绍的符号矩阵运算在形式上与数值计算中的运算十分相似,容易掌握。

3.4.1 基本代数运算

在 MATLAB 中,符号对象的代数运算和双精度运算从形式上看是一样的,由于 MATLAB

中采用了符号的重载，用于双精度运算的运算符同样可以用于符号对象。

【例 3-23】 符号矩阵的加减乘除运算。

```
>> syms a b c d;                          % 定义基本的符号变量
>> A = sym('[a b;c d]');                  % 定义符号矩阵
>> B = sym('[a 2 * b;c + b d - 2]');      % 定义符号矩阵
>> A + B;                                 % 计算符号矩阵的加法
>> A - B;                                 % 计算符号矩阵的减法
>> A * B;                                 % 计算符号矩阵的乘法
>> A/B;                                   % 计算符号矩阵的除法
>> syms a b c d;                          % 定义基本的符号变量
```

输出结果如下：

```
ans =
    [    2 * a,   3 * b]
    [ 2 * c + b, 2 * d - 2]
ans =
    [   0, - b]
    [ - b,   2]
ans =
    [a^2 + b * (c + b), 2 * a * b + b * (d - 2)]
    [c * a + d * (c + b), 2 * c * b + d * (d - 2)]
```

【例 3-24】 计算符号矩阵的 2 次方和 3 次方。

```
>> A = sym('[7 4 2;1 5 6;3 0 8]');        % 定义符号矩阵
>> A^2                                    % 计算符号矩阵的 2 次方
>> A^3                                    % 计算符号矩阵的 3 次方
```

输出结果如下：

```
A =
     [ 7, 4, 2]
     [ 1, 5, 6]
     [ 3, 0, 8]
ans =
     [ 59, 48, 54]
     [ 30, 29, 80]
     [ 45, 12, 70]
ans =
     [ 623, 476, 838]
     [ 479, 265, 874]
     [ 537, 240, 722]
```

【例 3-25】 计算符号矩阵的 4 次方。

```
>> A = sym('[1 2 3;4 5 6;7 8 9]');        % 定义符号矩阵
>> A^4
```

输出结果如下：

```
A =
    [ 1, 2, 3]
    [ 4, 5, 6]
    [ 7, 8, 9]
ans =
    [ 7560,  9288, 11016]
    [ 17118, 21033, 24948]
    [ 26676, 32778, 38880]
```

【例 3-26】 计算符号矩阵的指数。

```
>> A = sym('[1 2 3;4 5 6;7 8 9]');
>> B = exp(A)
```

输出结果如下：

```
A =
    [ 1, 2, 3]
    [ 4, 5, 6]
    [ 7, 8, 9]
B =
    [ exp(1), exp(2), exp(3)]
    [ exp(4), exp(5), exp(6)]
    [ exp(7), exp(8), exp(9)]
```

3.4.2 线性代数运算

符号对象的线性代数运算和双精度的线性代数运算一样，有关函数如表 3-4 所示。

表 3-4 符号矩阵线性运算函数

函数名称	功能介绍	函数名称	功能介绍
inv	矩阵求逆	null	零空间的正交基
det	计算行列式的值	colspace	返回矩阵列空间的基
diag	对角矩阵	transpose	返回矩阵的转置
triu	抽取矩阵的上三角部分	eig	特征值分解
tril	抽取矩阵的下三角部分	jordan	约当标准型变换
rank	计算矩阵的秩	svd	奇异值分解
rref	返回矩阵的所见行阶梯矩阵		

下面介绍各函数的具体用法。

1. inv 函数

inv 函数指令用于计算符号矩阵的逆，其具体用法如下：

inv(A)，计算符号矩阵 A 的逆。

【例 3-27】 生成数值希尔伯特矩阵，计算其逆矩阵。

```
>> A = hilb(4);              % 定义符号矩阵
>> A = sym(A);
>> inv(A)
```

输出结果如下：

```
A =
    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
ans =
    [    16,   -120,    240,   -140]
    [  -120,    1200,  -2700,    1680]
    [   240,   -2700,   6480,   -4200]
    [  -140,    1680,  -4200,    2800]
```

2. det 函数

det 函数指令用于计算符号矩阵的行列式，其具体用法如下：

det(A)，计算符号矩阵 A 的行列式。

【例 3-28】 计算例 3-5 中符号矩阵的行列式。

```
>> A = hilb(4);              % 定义符号矩阵
>> A = sym(A);
>> det(A)
```

输出结果如下：

```
A =
    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
ans =
    1.6534e-07
```

3. diag 函数

diag 函数指令用于实现对符号矩阵对角线元素的操作，其具体用法如下：

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 X 上方第 k 条对角线；当 k<0 时，v 为 X 下方第 k 条对角线。

diag(v)，与 k=0 相同，将向量 v 置于主对角线。

diag(A,k)，A 是矩阵，结果是由矩阵 A 的第 k 条对角线上的元素组成的列向量。

diag(A)，A 是矩阵，是 diag(A,k)用法中 k=0 的情况，结果是由矩阵 A 的主对角线元素组成的列向量。

【例 3-29】 a 是由 4 个元素构成的向量，利用 diag 函数求解符号矩阵的对角线。

```
>> syms a b c;               % 定义符号矩阵
>> A = [a b+1 c*3 4];
>> diag(A,1)
```

输出结果如下：

```
ans =
    [ 0, a,     0,    0, 0]
    [ 0, 0, b + 1,    0, 0]
    [ 0, 0,     0, 3*c, 0]
    [ 0, 0,     0,    0, 4]
    [ 0, 0,     0,    0, 0]
```

【例 3-30】 利用 diag 函数将向量 a 置于主对角线上。

```
>> syms a b c;               % 定义符号矩阵
>> A = [a b+1 c*3 4]
>> diag(A)
```

输出结果如下：

```
A =
    [ a, b + 1, 3*c, 4]
ans =
    [ a,     0,    0, 0]
    [ 0, b + 1,    0, 0]
    [ 0,     0, 3*c, 0]
    [ 0,     0,    0, 4]
```

【例 3-31】 A 为 4 阶希尔伯特矩阵，利用 diag 函数求矩阵的对角线。

```
>> A = hilb(4)
>> diag(A)
```

输出结果如下：

```
A =
    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
ans =
    1.0000
    0.3333
    0.2000
    0.1429
```

【例 3-32】 A是随机矩阵,分别找出由矩阵A的第1、2、3、4条对角线上的元素组成的列向量。

```
>> s = sym(4);              % 定义符号矩阵
>> class(s);
>> A = rand(s);
>> diag(A,3);
>> diag(A,2);
>> diag(A,1);
>> diag(A,0)
```

输出结果如下:

```
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
ans =
    0.9572
ans =
    0.9575
    0.4854
ans =
    0.6324
    0.9649
    0.8003
ans =
    0.8147
    0.0975
    0.1576
    0.1419
```

【例 3-33】 A是一个3阶魔方矩阵,利用diag函数找出矩阵A主对角线上元素的列向量。

```
>> s = sym(4);              % 定义符号矩阵
>> class(s);
>> A = magic(s);
>> diag(A)
```

输出结果如下:

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

4. triu 函数

triu 函数用来对符号矩阵的上三角部分进行操作，其具体用法如下：

triu(A)，抽取矩阵 A 主对角线之上的三角部分重新组成一个新矩阵，其他部分用 0 来填充。

triu(A,k)，抽取矩阵 A 的第 k 条对角线之上的部分重新组成一个新矩阵，其他部分用 0 来填充。当 k＞0 时，抽取的元素是在主对角线之上且在第 k 条对角线上的元素，其他部分用 0 来填充；当 k＜0 时，抽取的元素是在主对角线下且在第 k 条对角线上的元素，其他部分用 0 来填充；当 k＝0，即 triu(A,0)时，与 triu(A)相同，抽取主对角线之上的部分。

【例 3-34】 A 是一个 5 阶魔方矩阵，利用 triu 函数生成一个由 A 主对角线上的元素组成的矩阵。

```
>> s = sym(5);            % 定义符号矩阵
>> class(s);
>> A = magic(s);
>> triu(A)
```

输出结果如下：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
ans =
    17    24     1     8    15
     0     5     7    14    16
     0     0    13    20    22
     0     0     0    21     3
     0     0     0     0     9
```

【例 3-35】 A 是一个 5 阶魔方矩阵，利用 triu 函数生成由 A 主对角线之上的元素组成的矩阵，由 A 主对角线之下的元素组成的矩阵以及由 A 主对角线的元素组成的矩阵。

```
>> s = sym(5);            % 定义符号矩阵
>> class(s);
>> A = magic(s);
>> triu(A,2);
>> triu(A,-2);
>> triu(A,0);
```

输出结果如下：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
ans =
     0     0     1     8    15
     0     0     0    14    16
     0     0     0     0    22
     0     0     0     0     0
     0     0     0     0     0
ans =
    17    24     1     8    15
```

```
    23     5     7    14    16
     4     6    13    20    22
     0    12    19    21     3
     0     0    25     2     9
ans =
    17    24     1     8    15
     0     5     7    14    16
     0     0    13    20    22
     0     0     0    21     3
     0     0     0     0     9
```

【例 3-36】 A 是一个 3 阶随机矩阵,利用 triu 函数指令生成由 A 主对角线之上的元素组成的矩阵,由 A 主对角线之下的元素组成的矩阵以及由 A 主对角线的元素组成的矩阵。

```
>> s = sym(3);              % 定义符号矩阵
>> class(s);
>> A = rand(s);
>> triu(A,1);
>> triu(A, - 1);
>> triu(A,0);
```

输出结果如下:

```
A =
    0.4218    0.9595    0.8491
    0.9157    0.6557    0.9340
    0.7922    0.0357    0.6787
ans =
         0    0.9595    0.8491
         0         0    0.9340
         0         0         0
ans =
    0.4218    0.9595    0.8491
    0.9157    0.6557    0.9340
         0    0.0357    0.6787
ans =
    0.4218    0.9595    0.8491
         0    0.6557    0.9340
         0         0    0.6787
```

【例 3-37】 利用 triu 函数生成由符号矩阵 A 主对角线之上的元素组成的矩阵,由 A 主对角线之下的元素组成的矩阵以及由 A 主对角线的元素组成的矩阵,对比它们的不同。

```
>> syms a b c;              % 定义符号矩阵
>> A = [a^2 b + c 6 exp(c);a + b b a 5;4 b c 1;a^b c a 8]
>> triu(A);
>> triu(A,1);
>> triu(A, - 1)
```

输出结果如下:

```
A =
    [    a^2, b + c, 6, exp(c)]
    [  a + b,     b, a,      5]
    [      4,     b, c,      1]
    [    a^b,     c, a,      8]
ans =
    [ a^2, b + c, 6, exp(c)]
    [   0,     b, a,      5]
    [   0,     0, c,      1]
    [   0,     0, 0,      8]
```

```
ans =
    [ 0, b + c, 6, exp(c)]
    [ 0,     0, a,      5]
    [ 0,     0, 0,      1]
    [ 0,     0, 0,      0]
ans =
    [   a^2, b + c, 6, exp(c)]
    [ a + b,     b, a,      5]
    [     0,     b, c,      1]
    [     0,     0, a,      8]
```

5. tril 函数

tril 函数生成一个新矩阵，该新矩阵式抽取自原矩阵的下三角部分，其他部分用 0 来填充，具体用法如下：

tril(A)，抽取矩阵 A 的主对角线之下的三角部分重新组成一个新矩阵，其他部分用 0 来填充。

tril(A,k)，抽取矩阵 A 的第 k 条对角线之下的部分重新组成一个新矩阵，其他部分用 0 来填充。当 k>0 时，抽取的元素是在主对角线之上且第 k 条对角线之下的元素，其他部分用 0 来填充；当 k<0 时，抽取的元素是在主对角线之下且第 k 条对角线之下的元素，其他部分用 0 来填充；当 k=0，即 tril(A,0)时，与 tril(A)相同，抽取主对角线之下的部分。

【例 3-38】 利用 tril 函数生成由矩阵 A 主对角线之下的元素所组成的矩阵。

```
>> A = magic(4)
>> B = sym(A)
>> tril(B)
```

输出结果如下：

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
B =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
ans =
    [ 16,  0,  0, 0]
    [  5, 11,  0, 0]
    [  9,  7,  6, 0]
    [  4, 14, 15, 1]
```

【例 3-39】 利用 tril 函数生成由符号矩阵 A 主对角线之上的元素组成的矩阵，由 A 主对角线之下的元素组成的矩阵以及由 A 主对角线的元素组成的矩阵。

```
>> syms a b c;
>> A = [a^b c a 7;a + b b c 3;a^3 b + a 6 exp(c);2 a c 5]
>> B = sym(A)
>> tril(B,1)
>> tril(B,2)
>> tril(B, - 1)
>> tril(B, - 2)
>> tril(B,0)
```

输出结果如下：

```
A =
    [   a^b,     c, a,      7]
    [ a + b,     b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,     a, c,      5]
B =
    [   a^b,     c, a,      7]
    [ a + b,     b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,     a, c,      5]
ans =
    [   a^b,     c, 0,      0]
    [ a + b,     b, c,      0]
    [   a^3, a + b, 6, exp(c)]
    [     2,     a, c,      5]
ans =
    [   a^b,     c, a,      0]
    [ a + b,     b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,     a, c,      5]
ans =
    [     0,     0, 0, 0]
    [ a + b,     0, 0, 0]
    [   a^3, a + b, 0, 0]
    [     2,     a, c, 0]
ans =
    [   0, 0, 0, 0]
    [   0, 0, 0, 0]
    [ a^3, 0, 0, 0]
    [   2, a, 0, 0]
ans =
    [   a^b,     0, 0, 0]
    [ a + b,     b, 0, 0]
    [   a^3, a + b, 6, 0]
    [     2,     a, c, 5]
```

6. rank 函数

在线性代数中,一个矩阵 A 的列秩是 A 的线性无关的纵列的极大数目。类似地,行秩是 A 的线性无关的横行的极大数目。矩阵的列秩和行秩总是相等的,因此它们可以简单地称作矩阵 A 的秩。通常表示为 r(A)、rk(A)或 rank A。

在 MATLAB 中提供了 rank 函数指令用来计算符号矩阵的秩,其具体用法如下:

rank(A),返回矩阵 A 的秩。

【例 3-40】 利用 rank 指令计算四阶希尔伯特矩阵的秩。

```
>> A = hilb(4)
>> A = sym(A)
>> rank(A)
```

输出结果如下:

```
A =
    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
```

```
A =
    [   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]
ans =
     4
```

【例 3-41】 利用 rank 指令计算符号矩阵的秩。

```
>> syms a b c;
>> A = [b * 2 c a 2;c b a 3;c^2 7 b 1;b * 3 a c 8]
>> rank(A)
```

输出结果如下：

```
A =
    [ 2 * b, c, a, 2]
    [    c, b, a, 3]
    [  c^2, 7, b, 1]
    [ 3 * b, a, c, 8]
ans =
    4
```

7. rref 函数

矩阵的简化行阶梯式是高斯-约当消元法解线性方程组的结果，其形式为

$$\begin{bmatrix} 1 & \cdots & 0 & * \\ \vdots & \ddots & \vdots & * \\ 0 & \cdots & 1 & * \end{bmatrix}$$

MATLAB 提供了 rref 函数返回符号矩阵的简化行阶梯矩阵，其具体用法如下：

- R＝rref(A)，在计算的过程中利用高斯-约当消元法和行主元素法，返回矩阵的简化行阶梯矩阵 R。
- [R,jb]＝rref(A)，返回矩阵的简化行阶梯矩阵 R 和向量 jb。R(1:r,jb)为 r×r 阶不确定性矩阵，矩阵 A 的秩为 r＝length(jb)；x(jb)为线性系统 Ax＝b 的边界向量。
- [R,jb]＝rref(A,tol)，返回矩阵的简化行阶梯 R 和向量 jb 的要求与以上提到的相同，tol 指明了返回矩阵元素的误差。

【例 3-42】 使用 rref 函数返回符号矩阵 A 的简化行阶梯矩阵。

```
>> syms a b c;
>> A = [b * 2 c a 2;c b a 3;c^2 7 b 1;b * 3 a c 8]
>> R = rref(A)
```

输出结果如下：

```
A =
    [ 2 * b, c, a, 2]
    [    c, b, a, 3]
    [  c^2, 7, b, 1]
    [ 3 * b, a, c, 8]
R =
    [ 1, 0, 0, 0]
    [ 0, 1, 0, 0]
    [ 0, 0, 1, 0]
    [ 0, 0, 0, 1]
```

【例 3-43】 使用 rref 函数返回魔方矩阵的简化行阶梯矩阵。

```
>> A = magic(4)
>> A = sym(A)
>> R = rref(A)
```

输出结果如下：

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
R =
    [ 1, 0, 0,   1]
    [ 0, 1, 0,   3]
    [ 0, 0, 1, -3]
    [ 0, 0, 0,   0]
```

【例 3-44】 使用 rref 函数返回三阶希尔伯特矩阵 A 的简化行阶梯矩阵。

```
>> A = hilb(3)
>> A = sym(A)
>> R = rref(A)
```

输出结果如下：

```
A =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
A =
    [   1, 1/2, 1/3]
    [ 1/2, 1/3, 1/4]
    [ 1/3, 1/4, 1/5]
R =
    [ 1, 0, 0]
    [ 0, 1, 0]
    [ 0, 0, 1]
```

【例 3-45】 使用 rref 函数返回三阶随机矩阵 A 的简化行阶梯矩阵。

```
>> A = rand(3)
>> A = sym(A)
>> R = rref(A)
```

输出结果如下：

```
A =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
A =
    [ 7338378580900475/9007199254740992, 8226958330713791/9007199254740992,
    627122237356493/2251799813685248]
    [ 8158648460577917/9007199254740992, 1423946432832521/2251799813685248,
    153933462881711/281474976710656]
    [ 1143795557080799/9007199254740992,  109820732902227/1125899906842624,
    8624454854533211/9007199254740992]
```

```
R =
    [ 1, 0, 0]
    [ 0, 1, 0]
    [ 0, 0, 1]
```

8. null 函数

与线性系统相联系的两个子空间是值域和零空间。如果 A 为 m×n 的矩阵，它的秩为 r，那么 A 的向量空间就是由 A 的列划分的线性空间，这个空间的维数是 r，也就是 A 的秩。如果 r=n，则 A 的列线性无关。A 的零空间是由满足 Ax=0 的所有向量 x 组成的线性子空间。在 MATLAB 中可以用 null 函数来求得零空间的正交基，其具体用法如下：

N=null(A)：计算矩阵 A 的零空间的正交基，运算依赖矩阵 A 的奇异值分解。

N=null(A,'r')：计算矩阵 A 的零空间的正交基，运算依赖矩阵 A 的简化行阶梯矩阵。

【例 3-46】 使用 null 函数返回符号矩阵 A 的零空间正交基。

```
>> syms a b c;
>> A = [b * 2 c a 2;c b a 3;c^2 7 b 1;b * 3 a c 8]
>> N = null(A)
```

输出结果如下：

```
A =
    [ 2 * b, c, a, 2]
    [    c, b, a, 3]
    [  c^2, 7, b, 1]
    [ 3 * b, a, c, 8]
N =
     Empty sym: 4 - by - 0
```

【例 3-47】 使用 null 函数返回魔方矩阵 A 的零空间正交基。

```
>> A = magic(4)
>> A = sym(A)
>> N = null(A)
```

输出结果如下：

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
N =
     - 1
     - 3
       3
       1
```

9. colspace 函数

MATLAB 提供了 colspace 函数计算矩阵空间的基，其具体用法如下：

C=colspace(A)：返回符号矩阵 A 的列空间的基。

【例 3-48】 使用 colspace 函数指令计算符号矩阵 A 的列空间的基。

```
>> A = rand(4)
>> A = sym(A)
>> C = colspace(A)
```

输出结果如下：

```
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
A =
    [ 7338378580900475/9007199254740992, 1423946432832521/2251799813685248,
    8624454854533211/9007199254740992, 8621393422876569/9007199254740992]
    [ 8158648460577917/9007199254740992,  109820732902227/1125899906842624,
    8690943295155051/9007199254740992, 4371875181445801/9007199254740992]
    [ 1143795557080799/9007199254740992,  627122237356493/2251799813685248,
    354913107955861/2251799813685248, 1802071410739743/2251799813685248]
    [ 8226958330713791/9007199254740992,   153933462881711/281474976710656,
    2185580645132801/2251799813685248,  638999261770491/4503599627370496]
C =
    [ 1, 0, 0, 0]
    [ 0, 1, 0, 0]
    [ 0, 0, 1, 0]
    [ 0, 0, 0, 1]
```

【例 3-49】 使用 colspace 函数计算魔方矩阵 A 的列空间的基。

```
>> A = magic(4)
>> A = sym(A)
>> C = colspace(A)
```

输出结果如下：

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
C =
    [ 1, 0,   0]
    [ 0, 1,   0]
    [ 0, 0,   1]
    [ 1, 3, -3]
```

10. transpose 函数

MATLAB 提供了 transpose 函数计算矩阵的转置，其具体用法如下：

T＝transpose(A)：返回符号矩阵 A 的转置。

【例 3-50】 利用 transpose 函数计算符号矩阵 A 的转置矩阵。

```
>> syms a b c;
>> A = [ a^b c a 7;a + b b c 3;a^3 b + a 6 exp(c);2 a c 5]
>> T = transpose(A)
```

输出结果如下：

```
A =
    [   a^b,      c, a,      7]
    [ a + b,      b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,      a, c,      5]
T =
    [ a^b, a + b,    a^3, 2]
    [   c,     b,  a + b, a]
    [   a,     c,      6, c]
    [   7,     3, exp(c), 5]
```

【例 3-51】 利用 transpose 函数计算符号矩阵 A 的共轭的转置矩阵。

这里需要说明的是 A 与 A'是不同的，A'代表矩阵 A 的共轭矩阵，因此在求解转置矩阵的过程中所求的也是不同的矩阵，需要注意。

输入指令如下：

```
>> syms a b c;
>> A = [ a^b c a 7;a + b b c 3;a^3 b + a 6 exp(c);2 a c 5]
>> T = transpose(A')
```

输出结果如下：

```
A =
    [   a^b,      c, a,      7]
    [ a + b,      b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,      a, c,      5]
T =
    [         conj(a^b),           conj(c), conj(a),           7]
    [ conj(a) + conj(b),           conj(b), conj(c),           3]
    [         conj(a)^3, conj(a) + conj(b),       6, exp(conj(c))]
    [                2,           conj(a), conj(c),           5]
```

其中，conj 用于计算共轭。

11. eig 函数

eig 函数用于对符号矩阵进行特征值分解，即计算矩阵的特征值和特征向量，其具体用法如下：

E＝eig(A)，返回由方阵 A 的特征值组成的矩阵。

[V,D]＝eig(A)，返回方阵 A 的特征值矩阵 D 和特征向量矩阵 V，其中特征值矩阵 D 是以 A 的特征值为对角线的对角矩阵，V、D 和 A 之间满足 AV＝VD。

【例 3-52】 利用 eig 函数计算三阶随机矩阵的特征值和特征向量。

```
>> syms a b d;
>> A = [a 1;b d]
>> E = eig(A)
>> [V,D] = eig(A)
```

输出结果如下：

```
A =
    [ a, 1]
    [ b, d]
E =
    a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2
    a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2
```

```
V =
    [ (a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2)/b - d/b,
    (a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2)/b - d/b]
    [                  1,                          1]
D =
    [ a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2,                              0]
    [ 0,                             a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2]
```

12. jordan 函数

jordan 函数用于将矩阵变换为约当标准型,计算约当标准型也就是找一个非奇异矩阵 V,使 J=V/A*V 最接近对角矩阵,其中 V 称为转换矩阵。利用矩阵分块可以简化很多有关矩阵的证明和计算,任何仿真都可以通过相似变换,变为约当标准型。jordan 函数的具体用法如下:

J=jordan(A),返回矩阵 A 的约当标准型。

[V,J]=jordan(A),返回矩阵 A 的约当标准型并且给出变换矩阵 V,满足 J=V/A*V。

【例 3-53】 利用 jordan 函数指令计算三阶魔方矩阵的特征值和特征向量。

```
>>  A = magic(3)
>>  A = sym(A)
>>  [V,J] = jordan(A)
```

输出结果如下:

```
A =
     8     1     6
     3     5     7
     4     9     2
A =
    [ 8, 1, 6]
    [ 3, 5, 7]
    [ 4, 9, 2]
V =
    [ (2*6^(1/2))/5 - 7/5,  - (2*6^(1/2))/5 - 7/5, 1]
    [ 2/5 - (2*6^(1/2))/5,    (2*6^(1/2))/5 + 2/5, 1]
    [                   1,                       1, 1]
J =
    [  -2*6^(1/2),          0,  0]
    [           0, 2*6^(1/2),  0]
    [           0,          0, 15]
```

13. svd 函数

svd 函数用来进行矩阵的奇异值分解,奇异值分解在矩阵分解中占有极其重要的地位。svd 函数的具体用法如下:

[U,S,V] = svd (X),返回一个与 X 同大小的对角矩阵 S,两个矩阵 U 和 V,且满足= U*S*V'。若 A 为 m×n 阵,则 U 为 m×m 阵,V 为 n×n 阵。奇异值在 S 的对角线上,非负且按降序排列。

[U,S,V] = 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×n。

【例 3-54】 利用[U,S,V]=svd(A)函数对符号矩阵 A 进行奇异值的分析。

```
>> A = [9 8 7;6 5 4;3 2 1]
>> digits(30)
>> [U,S,V] = svd(A)
```

输出结果如下：

```
A =
     9     8     7
     6     5     4
     3     2     1
U =
   -0.8263    0.3879    0.4082
   -0.5206   -0.2496   -0.8165
   -0.2148   -0.8872    0.4082
S =
   16.8481         0         0
         0    1.0684         0
         0         0    0.0000
V =
   -0.6651   -0.6253   -0.4082
   -0.5724    0.0757    0.8165
   -0.4797    0.7767   -0.4082
```

3.4.3 科学计算

极限、微分和积分是微积分学中的核心和基础，MATLAB 提供了强大的函数指令来对其进行计算，下面做简单的介绍。

1. 符号极限的计算

极限是高等数学的出发点和基础，高等数学的许多内容是建立在极限理论基础上的。在 MATLAB 中提供了 limit 函数来对极限进行运算，其用法如下：

limit(fx,a)，当变量 x 趋近于常数 a 时，计算符号函数 f(x)的极限值。

limit(f,a)，相当于变量 x 趋近于 a 时计算符号函数 f(x)的极限值。在没有指定符号函数 f(x)的自变量时，使用此函数来计算符号函数的极限，系统按 findsym 函数指示的默认变量来确定符号函数 f(x)的变量。

limit(f)，在没有指定变量的目标值时，系统默认变量趋近于 0，相当于变量 x 趋近于 0 时计算符号函数 f(x)的极限值，系统按 findsym 函数指示的默认变量来确定符号函数 f(x)的变量。

limit(f,x,a,'right')和 limit(f,x,a,'left')，由于求极限可以从两边趋近，对于某些从左面趋近和从右面趋近得到的结果是不同的，针对这种情况，函数 limit 专门提供了本语句来计算函数的左极限和右极限。'right'表示符号函数 f(x)的右极限，即变量 x 从右边趋近于 a；'left'表示符号函数 f(x)的左极限，即变量 x 从左边趋近于 a。

【例 3-55】 计算 $\lim\limits_{x\to\infty}\dfrac{3x^2-2x-1}{7x^3-5x^2+3}$。

```
>> syms x;
>> f = (3 * x * x - 2 * x - 1)/(7 * x * x * x - 5 * x * x + 3);
>> limit(f,x,inf)
```

输出结果如下：

```
ans =
     0
```

【例 3-56】 计算 $\lim\limits_{x\to1}(2x-1)$。

```
>> syms x;
>> f = 2 * x - 1;
>> limit(f,x,1)
```

输出结果如下：

```
ans =
     1
```

【例 3-57】 计算 $\lim\limits_{x\to 0}\left(\dfrac{a^x+b^x+c^x}{3}\right)^{\frac{1}{x}}$。

```
>> syms a b c x;
>> f = ((a^x + b^x + c^x)/3)^(1/x);
>> limit(f,a)
```

输出结果如下：

```
ans =
     (a^a + b^a + c^a)^(1/a)/3^(1/a)
```

【例 3-58】 计算 $\lim\limits_{x\to 2}\dfrac{x^2+5}{x-3}$。

```
>> syms x;
>> f = (x^2 + 5)/(x - 3);
>> limit(f,x,2)
```

输出结果如下：

```
ans =
     -9
```

【例 3-59】 计算 $f=\lim\limits_{x\to 0}\dfrac{\tan x-\sin x}{x^3}$的极限。

```
>> syms x;
>> f = (tan(x) - sin(x))/x^3;
>> limit(f,x,0,'left')
```

输出结果如下：

```
ans =
     1/2
```

【例 3-60】 计算例 3-59 中函数的右极限。

```
>> syms x;
>> f = (tan(x) - sin(x))/x^3;
>> limit(f,x,0,'right')
```

输出结果如下：

```
ans =
     1/2
```

【例 3-61】 计算 $f=\dfrac{1}{x^3}$当 x 趋近于 0 时的极限值，并分别求出函数的左极限和右极限。

```
>> syms x;
>> f = 1/x^3;
>> limit(f,x,0)
>> limit(f,x,0,'left')
>> limit(f,x,0,'right')
```

输出结果如下：

```
ans =
    NaN
ans =
    - Inf
ans =
    Inf
```

2. 符号微分的计算

MATLAB 提供了 diff 函数指令计算符号表达式的微分，具体用法如下：

diff(s)，没有指定变量和导数阶数，则系统按 findsym 函数指定的默认变量对符号表达式 s 求一阶微分。

diff(s,'v')，以 v 为自变量，对符号表达式 s 求一阶微分。

diff(s,n)，按 findsym 函数指定的默认变量对符号表达式 s 求 n 阶微分，且 n 为正整数。

diff(s,'v',n)，以 v 为自变量，对符号表达式 s 求 n 阶微分，n 为正整数。

【例 3-62】 计算 $2x^3+4x^2+\cos(x)+\sin^2(x)$函数关于 x 的微分。

```
>> f = sym('2 * x^3 + 4 * x^2 + cos(x) + sin(x)^2');
>> df = diff(f)
```

输出结果如下：

```
df =
    8 * x - sin(x) + 2 * cos(x) * sin(x) + 6 * x^2
```

【例 3-63】 计算 $f=x-\ln(1+x)$的二阶微分。

```
>> f = sym(x == log(1 + x));
>> df = diff(f,2)
```

输出结果如下：

```
df =
    0 = -1/(x + 1)^2
```

【例 3-64】 计算 $f=3ax^2+5bx+6$ 关于 b 的微分。

```
>> f = sym('3 * a * x^2 + 5 * b * x + 6');
>> df = diff(f,'b')
```

输出结果如下：

```
df =
    5 * x
```

【例 3-65】 计算 $f=10x$ 的微分。

```
>> syms x
>> f = 10 * x * exp(-x/2);
>> df = diff(f)
```

输出结果如下：

```
df =
    10 * exp(-x/2) - 5 * x * exp(-x/2)
```

3. 符号积分的计算

积分运算是微分运算的逆运算，MATLAB 提供了 int 函数指令计算符号表达式的积分。该函数既可以计算定积分，也可以计算不定积分和广义积分，其具体用法如下：

int(s)，没有指定积分变量和积分阶数，系统按 findsym 函数指示的默认变量对被积函数

或符号表达式 s 求不定积分。

int(s,v),以 v 为自变量,计算被积函数或符号表达式 s 的不定积分。

int(s,v,a,b),计算表达式 s 的定积分。该函数是求在[a,b]区间上的定积分,a 和 b 分别是定积分的下限和上限。a 和 b 可以是两个具体的数,也可以是一个符号表达式或是无穷(inf),当 a 和 b 有一个或两个是 inf 时,函数返回一个广义积分;当 a 和 b 中有一个符号表达式时,函数返回一个符号函数。系统按 findsym 函数指示的默认变量来确定表达式的变量,当表达式 s 是符号矩阵时,则对矩阵的各个元素分别进行积分。

int(s,v,a,b),符号表达式采用符号标量 v 作为标量,求 v 从 a 变到 b 时,符号表达式 s 的定积分值,a 和 b 的规定同上。

【例 3-66】 计算 $f=\tan^3 x$ 关于 x 的不定积分。

```
>> f = sym('tan(x)^3');
>> int(f)
```

输出结果如下:

```
ans =
     log(cos(x)) - (cos(x)^2 - 1)/(2 * cos(x)^2)
```

【例 3-67】 计算 $f=x^3+a\cos x+b\sin^2 x$ 关于 b 的不定积分。

```
>> f = sym('x^3 + a * cos(x) + b * sin(x)^2');
>> int(f,'b')
```

输出结果如下:

```
ans =
     b * (a * cos(x) + x^3) - b^2 * (cos(x)^2/2 - 1/2)
```

【例 3-68】 计算定积分 $\int_{\frac{\pi}{4}}^{\frac{\pi}{3}} \frac{x}{\sin^2 x}\mathrm{d}x$。

```
>> syms x;
>> f = x/sin(x)^2;
>> int(f,x,pi/4,pi/3)
```

输出结果如下:

```
ans =
     pi/4 + log(6^(1/2)/2) - (pi * 3^(1/2))/9
```

【例 3-69】 计算定积分 x^2+16y^2。

```
>> syms x y;
>> f = x^2 + 16 * y^2;
>> int(f,x)
>> int(f,y)
>> a = 5;b = 10
>> int(f,x,a,b)
```

输出结果如下:

```
ans =
    x^3/3 + 16 * x * y^2
ans =
    x^2 * y + (16 * y^3)/3
ans =
    80 * y^2 + 875/3
```

4. 级数求和的计算

symsum 函数指令用于计算符号表达式的和，其具体用法如下：

r＝symsum(s)，自变量是由 findsym 函数所确定的符号变量，默认自变量为 k，计算表达式 s 从 0 到 k－1 的和。

r＝symsum(s,v)，计算表达式 s 从 0 到 v－1 的和。

r＝symsum(s,a,b)，计算表达式 s 默认变量从 a 到 b 的和。

r＝symsum(s,v,a,b)，计算表达式 s 变量 v 从 a 到 b 的和。

【例 3-70】 利用 symsum 函数计算符号表达式的和。

```
>> syms x k;
>> symsum(x + 3 * x^4)
>> symsum(1/x^k,k,0,inf)
>> symsum(1/x^k,k,1,inf)
```

输出结果如下：

```
ans =
    (3 * x^5)/5 - (3 * x^4)/2 + x^3 + x^2/2 - (3 * x)/5
ans =
    x/( - 1 + x)
ans =
    x/( - 1 + x)
```

【例 3-71】 对 a * n^2＋b * n 求级数和，其中 n 从 1 变到 20。

```
>> syms a b n;
>> f = a * n^2 + b * n;
>> symsum(f,n,1,20)
```

输出结果如下：

```
ans =
     2870 * a + 210 * b
```

5. 泰勒级数的计算

MATLAB 提供了 taylor 函数用来计算符号表达式的泰勒级数展开式，其具体用法如下：

r＝taylor(f)，f 是符号表达式，自变量是由 findsym 函数所确定的符号变量，该函数将返回 f 在变量等于 0 处进行 5 阶泰勒展开时的展开式。

r＝taylor(f,n,v)，符号表达式 f 以符号标量 v 作为自变量，返回 f 的 n－1 阶麦克劳林级数(即在 v＝0 处进行泰勒展开)展开式。

r＝taylor(f,n,v,a)，返同符号表达式 f 在 v＝a 处进行 n－1 阶泰勒展开的展开式。

【例 3-72】 使用 taylor 函数指令计算符号表达式的泰勒级数展开式。

```
>> syms x;
>> f = f * x^3;
>> T = taylor(f)
```

输出结果如下：

```
T =
    x^3 * (a * n^2 + b * n)
```

【例 3-73】 使用 taylor 函数指令计算符号表达式的泰勒级数展开式。

```
>> syms x a;
>> f = x^a;
>> T = taylor(f,4,a)
```

输出结果如下：

```
T =
    (a^3*log(x)^3)/6 + (a^2*log(x)^2)/2 + a*log(x) + 1
```

3.5 符号表达式积分及其变换

积分变换是工程设计和计算常用的工具，常见的积分变换有傅里叶(Fourier)变换、拉普拉斯(Laplace)变化和 Z 变换。

3.5.1 傅里叶变换及其反变换

傅里叶变换是一种分析信号的方法，它可以分析信号的成分，也可以用这些成分合成信号。许多波形可作为信号的成分，比如正弦波、方波、锯齿波等，傅里叶变换用正弦波作为信号的成分。

$f(x)$是关于 x 的函数，如果 x 满足狄里赫莱条件：具有有限个间断点，具有有限个极值点，绝对可积，则有如下公式成立：

$$F(\omega)=\int_{-\infty}^{+\infty}f(x)\mathrm{e}^{-\mathrm{j}\omega x}\,\mathrm{d}x$$

时域中的 $f(x)$和频域中的 $F(\omega)$的傅里叶反变换存在如下关系：

$$f(x)=\frac{1}{\pi}\int_{-\infty}^{\infty}F(\omega)\mathrm{e}^{-\mathrm{j}\omega x}\,\mathrm{d}\omega$$

1. 傅里叶变换

在 MATLAB 中进行傅里叶变换的函数指令如下：

(1) F=fourier(f)，函数 f 的默认自变量是 x，对默认变量计算傅里叶变换时，并且默认输出结果 F 是变量 ω 的函数，记为 $F(\omega)=\int_{-\infty}^{+\infty}f(x)\mathrm{e}^{-\mathrm{j}\omega x}\,\mathrm{d}x$。

(2) F=fourier(f,v)，返回结果为 v 的函数，记为 $F(v)=\int_{-\infty}^{+\infty}f(x)\mathrm{e}^{-\mathrm{j}vx}\,\mathrm{d}x$。

(3) F=fourier(f,u,v)，函数 f 的指定自变量是 u，指定参数变为 v，对函数 f 进行傅里叶变换，记为 $F(v)=\int_{-\infty}^{+\infty}f(u)\mathrm{e}^{-\mathrm{j}vu}\,\mathrm{d}u$。

【例 3-74】 分别在默认自变量和指定参数 v 的情况下计算 f(x)的傅里叶变换。

```
>> syms x w u v;
>> f = sin(x) - cos(x) + 1;
>> fourier(f)
>> fourier(f,v)
```

输出结果如下：

```
ans =
    2*pi*dirac(w) - pi*(dirac(w - 1) + dirac(w + 1)) - pi*(dirac(w - 1) - dirac(w + 1))*i
ans =
    2*pi*dirac(v) - pi*(dirac(v - 1) + dirac(v + 1)) - pi*(dirac(v - 1) - dirac(v + 1))*i
```

【例 3-75】 使用 fourier(f,u,v)函数在指定自变量和变换参数的情况下计算 f(x)的傅里叶变换。

```
>> syms t u v;
>> f = exp( - 1/3*(t + u)^2);
>> fourier(f,t,v)
```

输出结果如下：

```
ans =
     3^(1/2) * pi^(1/2) * exp( - (3 * ((u * 2i)/3 - v)^2)/4 - u^2/3)
```

2. 傅里叶反变换

ifourier(F)：在系统默认自变量和变换参数的情况下，计算函数的傅里叶反变换，记为 $f(x)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}F(\omega)e^{-j\omega x}d\omega$。

ifourier(F,v)：在系统默认自变量，并指定变换参数是 v 的情况下，计算函数的傅里叶反变换，记为 $f(v)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}F(\omega)e^{-j\omega v}d\omega$。

ifourier(F,w,v)：在系统的自变量为 w，并指定变换参数是 v 的情况下，计算函数的傅里叶反变换，记为 $f(v)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}F(w)e^{-jwv}dw$

【例 3-76】 使用函数 f= ifourier(F)在系统默认自变量和变换参数的情况下计算函数的傅里叶反变换。

```
>> syms x w u v;
>> f = sin(x) - cos(x) + 1;
>> ifourier(f)
```

输出结果如下：

```
ans =
     (2 * pi * dirac(t) - pi * (dirac(t - 1) + dirac(t + 1)) + pi * (dirac(t - 1) - dirac(t + 1))
 * i)/(2 * pi)
```

【例 3-77】 使用 ifourier(F,v)函数在指定自变量和返回结果参数的情况下计算 f(x)的傅里叶反变换。

```
>> syms x y;
>> f = exp( - (x + y)^2/3);
>> ifourier(f,v)
```

输出结果如下：

```
ans =
     (3^(1/2) * exp( - (3 * (v + (y * 2i)/3)^2)/4 - y^2/3))/(2 * pi^(1/2))
```

3.5.2 拉普拉斯变换及其反变换

拉普拉斯变换是工程数学中常用的一种积分变换，又名拉氏变换。拉氏变换是一个线性变换，可将一个有引数实数 $t(t\geqslant 0)$的函数转换为一个引数为复数 s 的函数。

如果函数 $f(t)$ 在区间$[0,+\infty)$ 上有定义，并且积分$\int_0^{+\infty}f(t)e^{-st}dt$ 在 s 的某一区域内收敛，则由这个积分确定函数 $F(s)$，即 $F(s)=\int_0^{+\infty}f(t)e^{-st}dt$。此式称为函数 $f(t)$ 的拉普拉斯变换式，记为 $L[f(t)]=F(s)$。

拉普拉斯反变换定义为 $F(t)=\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}L(s)e^{st}dt$，其中 c 为使函数$L(s)$的所有奇点位于直线 $s=c$ 左边的实数，拉普拉斯反变换记为 $F(t)=L^{-1}[L(s)]$。

1. 拉普拉斯变换

在 MATLAB 中提供了如下函数来进行拉普拉斯变换：

laplace(F),在默认自变量(为 x)和默认参变量(为 s)的情况下,计算符号函数的拉普拉斯变换,记为 $L(s)=\int_0^{+\infty}F(x)\mathrm{e}^{-sx}\mathrm{d}x$。

laplace(F,z),在默认自变量(为 x),并指定参变量为 z 的情况下,计算函数的拉普拉斯变换,记为 $L(z)=\int_0^{+\infty}F(x)\mathrm{e}^{-zx}\mathrm{d}x$。

laplace(F,w,z),在指定自变量为 w,并指定参变量为 z 的情况下,计算函数的拉普拉斯变换,记为 $L(z)=\int_0^{+\infty}F(w)\mathrm{e}^{-zw}\mathrm{d}w$。

【例 3-78】 使用函数 laplace(F)在默认自变量和参变量的情况下计算符号函数的拉普拉斯变换。

```
>> syms x;
>> f = 2 * sin(3 * x);
>> laplace(f)
```

输出结果如下:

```
ans =
     6/(s^2 + 9)
```

【例 3-79】 使用函数 laplace(F,w,z)和 f(F,z)分别在指定自变量为 w 并指定参变量为 z 以及在默认自变量并指定参变量为 z 的情况下计算符号函数的拉普拉斯变换。

```
>> syms w z x;
>> f = 1 + log(x + 2);
>> l = laplace(f,w,z)
>> l = laplace(f,z)
```

输出结果如下:

```
l =
   (log(x + 2) + 1)/z
l =
   log(2)/z + 1/z + (exp(2 * z) * (log(2 * z) - log(2) - log(z) + expint(2 * z)))
```

2. 拉普拉斯函数反变换

ilaplace(L),在默认自变量(为 s)和默认参变量(为 t)的情况下,计算函数 $L(s)$的拉普拉斯反变换。

ilaplace(L,v),在默认自变量(为 s)并指定参变量 v 的情况下,计算函数$L(s)$的拉普拉斯反变换。

ilaplace(L,v,x),在指定自变量(为 x)并指定参变量 v 的情况下,计算函数 $L(s)$的拉普拉斯反变换。

【例 3-80】 使用函数 ilaplace(L)在默认自变量和默认参变量的情况下计算函数 $L(s)$的拉普拉斯反变换。

```
>> syms x;
>> f = cos(x - 2);
>> L = ilaplace(f)
```

输出结果如下:

```
L =
    ilaplace(cos(x - 2), x, t)
```

【例 3-81】 使用函数 ilaplace(L,v)和 ilaplace(L,v,x)计算函数的拉普拉斯反变换。

```
>> syms v x w;
>> L = (v^3 + w^2 + 3);
>> ilaplace(L,v)
```

输出结果如下：

```
ans =
     3 * dirac(v) + v^3 * dirac(v) + dirac(v, 2)

>> fn = laplace(L,v)
>> ilaplace(fn,v,x)      % 指定自变量 x 参变量 v,求函数的逆变换
```

输出结果如下：

```
ans =
x^2 + dirac(x,2) + 3
```

3.5.3 Z 变换及其反变换

1. Z 变换

Z 变换(Z-transformation)是对离散序列进行的一种数学变换，常用于求线性时不变差分方程的解。它在离散系统中的地位如同拉普拉斯变换在连续系统中的地位。

在 MATLAB 中进行 Z 变换的函数是：

(1) ztrans(f)：在默认自变量(n)和参变量(z)的情况下，计算符号函数的 Z 变换。

(2) ztrans(f,v)：在默认自变量(n)，参变量(v)的情况下，计算符号函数的 Z 变换。

(3) ztrans(f,k,v)：在自变量(k)，参变量(v)的情况下，计算符号函数的 Z 变换。

【例 3-82】 使用 ztrans(f)函数在默认自变量和参变量的情况下对函数进行 Z 变换。

```
>> syms x;
>> f = cos(4 * x);
>> ztrans(f)
```

输出结果如下：

```
ans =
     (z * (z - cos(4)))/(z^2 - 2 * cos(4) * z + 1)
```

【例 3-83】 使用 ztrans(f,v)函数与 ztrans(f,k,v)分别对函数进行 Z 变换。

```
>> syms k m v;
>> f = tan(3 * k * m);
>> ztrans(f,v)
```

输出结果如下：

```
ans =
     ztrans(tan(3 * k * m), m, v)
>> f = sin(k)
>> ztrans(f,k,v)
ans =
     (v * sin(1))/(v^2 - 2 * cos(1) * v + 1)
```

2. Z 反变换

Z 反变换记为 $f(n)=Z^{-1}(F(z))$。

MATLAB 提供了 iztrans 函数实现 Z 反变换，其调用方法如下：

iztrans(F)，在默认自变量(为 n)和默认参变量(为 z)的情况下，对函数进行 Z 反变换。

itrans(F,v),在默认自变量(为 n)并指定参变量(为 v)的情况下,对函数进行 Z 反变换。

itrans(F,w,v),在指定自变量(为 w)并指定参变量(为 v)的情况下,对函数进行 Z 反变换。

【例 3-84】 使用 iztrans(F)函数指令在默认自变量和指定参变量 v 的情况下对函数进行 Z 反变换。

```
>> syms w,v;
>> f = w * (w - 3) * (w/2 + f * w - 2);
>> iztrans(f,v)
```

输出结果如下:

```
ans =
    (sin(k) + 1/2) * iztrans(w^3, w, v) - (3 * sin(k) + 7/2) * iztrans(w^2, w, v) + 6 * iztrans(w, w, v)
```

【例 3-85】 使用函数指令 iztrans(F,v)和 iztrans(F,w,v)分别计算函数的 Z 反变换,输入指令如下:

```
>> syms z a v
>> F = exp(a/z);
>> iztrans(F,v)
```

输出结果如下:

```
ans =
    a^v/factorial(v)
>> syms v x
>> F = 2 * x/(x - 2)^2;
>> iztrans(f,x,v)
```

输出结果如下:

```
ans =
    w * (w - 3) * kroneckerDelta(v, 0) * (w/2 + w * sin(k) - 2)
```

3.6 绘制符号函数图形

图形是解决数学问题的必要途径,MATLAB 除了为解决代数方程提供支持外,还对函数图像的绘制提供了强大的支持。本节对符号函数绘制加以简单介绍,详细的图形绘制在第 4 章系统地学习。

3.6.1 绘制曲线

MATLAB 提供了 ezplot 函数和 ezplot3 函数用于绘制符号函数的二维曲线和三维曲线。

1. 二维曲线的绘制

MATLAB 提供了 ezplot 函数来绘制符号函数的二维曲线,此函数可以绘制显函数图形、隐函数图形和参数方程的图形,具体用法如下:

ezplot(f),绘制显函数 f 在区间[−2,2]的二维曲线;绘制参数方程 x=x(t),y=y(t)在区间 0<t<2 的曲线。

- ezplot(f,[min,max])、ezplot(f,[xmin,xmax,ymin,ymax])和 ezplot(x,y,[tmin,tmax]),第一种用法是绘制显函数 f 在指定区间[min,max]的二维曲线;第二种用法是绘制隐函数 f 在指定区间 xmin<x<xmax、ymin<y<ymax 的曲线;第三种用法是绘制参数方程 x=x(t),y=y(t)在区间 tmin<t<tmax 的曲线。

【例 3-86】 使用 ezplot(f)函数绘制显函数的二维曲线。

```
>> syms x;
>> f = sin(x);
>> ezplot(f);
>> grid;
>> title('sin(x)');
```

输出结果如图 3-1 所示。

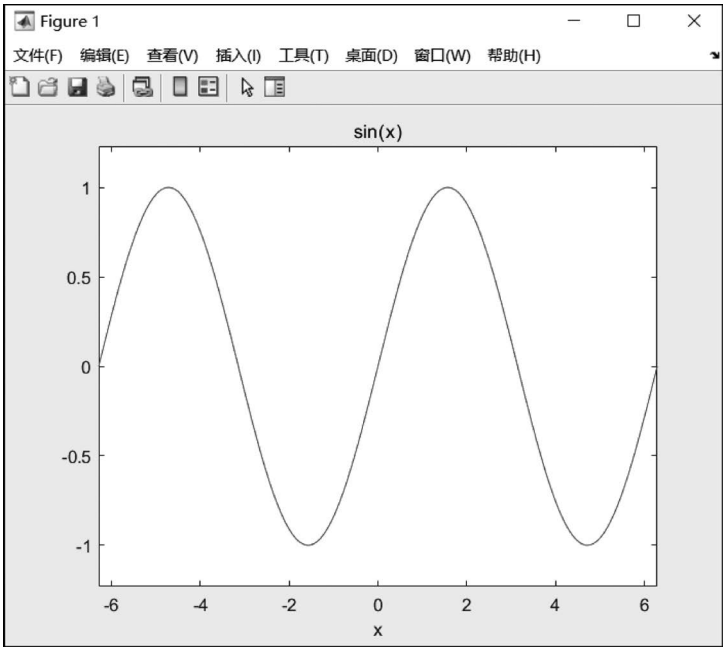

图 3-1 ezplot(f)函数绘制的二维曲线

【例 3-87】 绘制函数 $y=x^3-x^2-x+1$ 的二维曲线。

```
>> syms x;
>> f = x^3 - X^2 - x + 1;
>> ezplot(f);
>> grid;
>> title('exp');
```

输出图形如图 3-2 所示。

2. 三维曲线的绘制

ezplot3 函数用于绘制符号函数的三维曲线,具体用法如下:

ezplot3(x,y,z),绘制参数方程 x=x(t),y=y(t),z=z(t)在默认区间 0<t<2 的三维曲线。

ezplot3(x,y,z,[tmin,tmax]),绘制参数方程 x=x(t), y=y(t), z=z(t)在区间 tmin<t<tmax 的三维曲线。ezplot3(…'animate'),生成空间曲线的动态轨迹。

【例 3-88】 绘制函数的三维曲线。

```
>> syms t;
>> x = cos(t);
>> y = tan(t);
>> z = t;
>> ezplot3(x,y,z)
```

输出图形如图 3-3 所示。

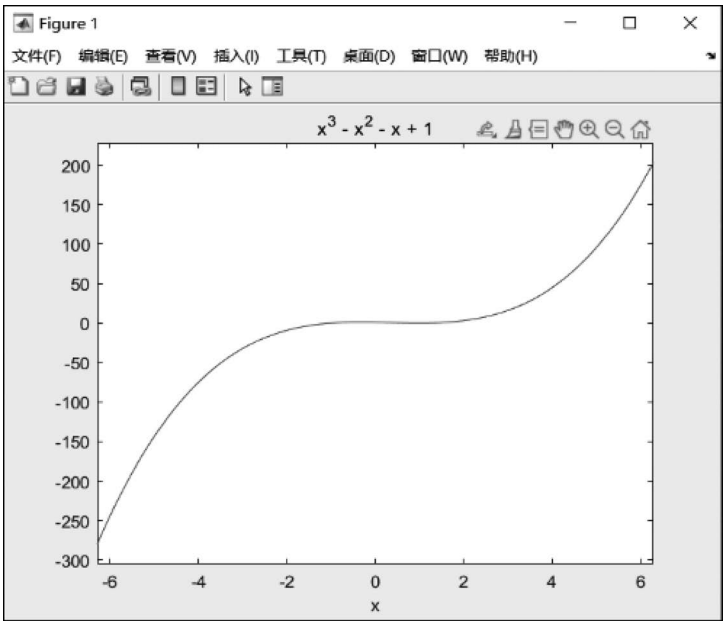

图 3-2　二维曲线

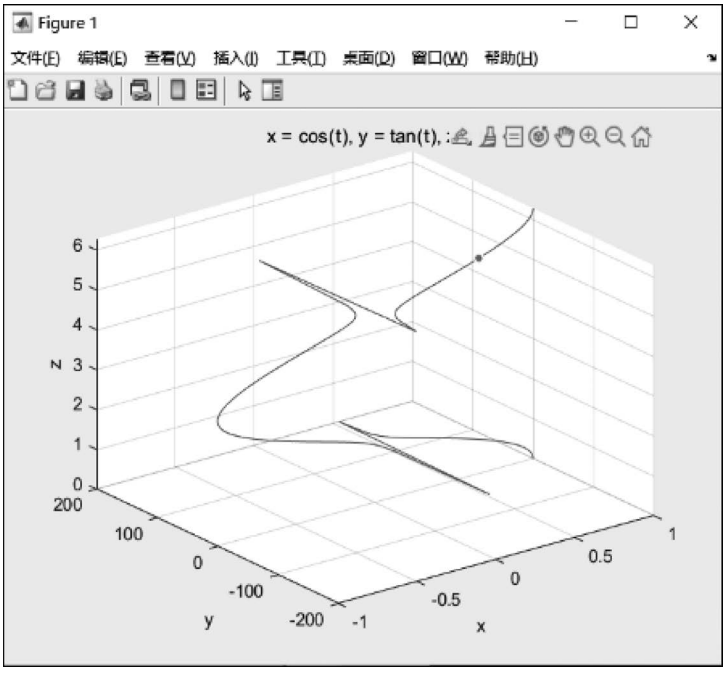

图 3-3　三维曲线

3.6.2　符号函数等值线的绘制

ezcontour 函数和 ezcontourf 函数用于绘制符号函数的等值线,两个函数的使用方法类似,区别在于 ezcontourf 函数绘制带有填充区域的等值线。ezcontour 函数的具体用法如下:

ezcontour(f),绘制二元函数 f(x,y)在默认区域的等值线。

ezcontour(f,domain),绘制二元函数 f(x,y)在指定区域的等值线。

ezcontour(…,n),绘制等值线图,并指定等值线的条数。

【例 3-89】 使用 ezcontour 函数绘制符号函数的等值线。

```
>> syms x;
>> f = x^3 - x^2 - x + 1;
>> ezcontour(f);
```

输出图像如图 3-4 所示。

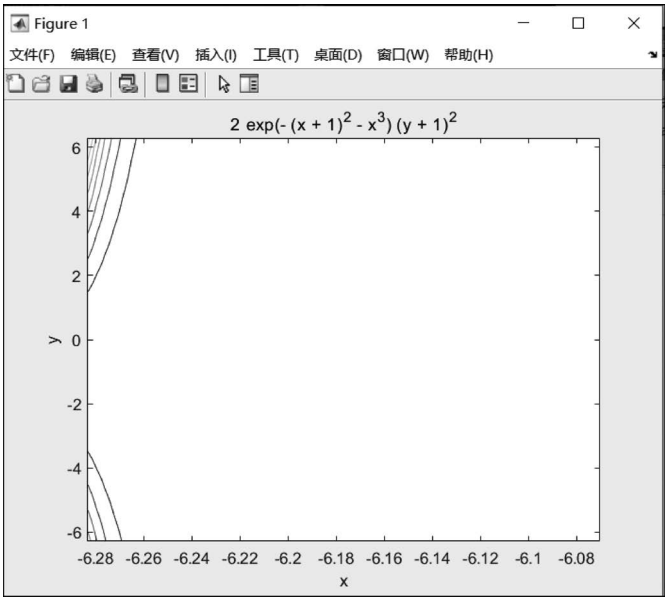

图 3-4 ezcontour 函数绘制的等值线

【例 3-90】 使用 ezcontourf 函数绘制符号函数的等值线。

```
>> syms x y;
>> f = 2 * (1 + y)^2 * exp( - (x^3) - (x + 1)^2);
>> ezcontourf(f);
```

输出图像如图 3-5 所示。

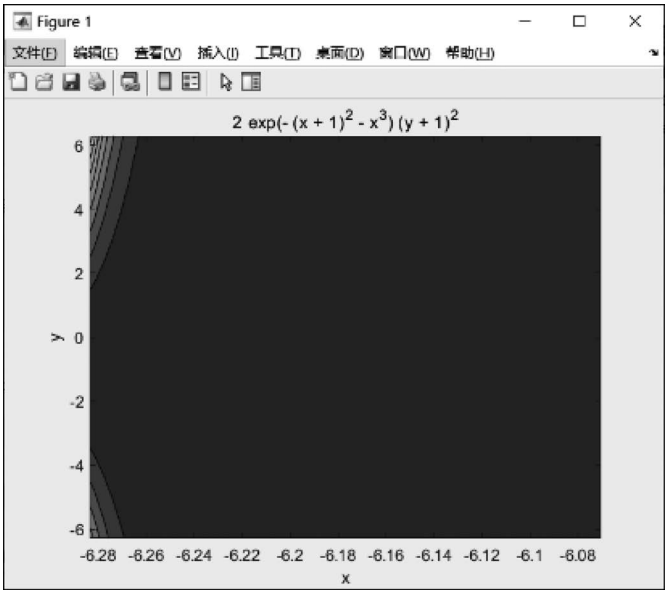

图 3-5 ezcontourf 函数绘制的等值线

3.6.3 符号函数曲面图及表面图的绘制

MATLAB提供了ezmesh函数和ezmeshc函数用于绘制符号函数的三维曲面图,以及ezsurf函数和ezsurfc函数用于绘制符号函数的三维表面图。

ezmesh函数和ezmeshc函数的区别在于:ezmeshc函数在绘制三维曲面图的同时绘制等值线。ezsurf函数和ezsurfc函数的区别在于:ezsurfc函数在绘制三维表面图的同时绘制等值线。

1. ezmesh函数和ezsurf函数

ezmesh函数用于绘制三维曲面图,ezsurf函数绘制三维表面图。两个函数的用法相似。ezmesh的具体用法如下:

ezmesh(f),绘制f(x,y)的图像。

ezmesh(f,domain),在指定区域绘制f(x,y)的图像。

ezmesh(x,y,z),在默认区域绘制三维参数方程的图像。

ezmesh(x,y,z,[smin,smax,tmin,tmax])或ezmesh(x,y,z,[min,max]),在指定区域绘制维参数方程的图像。

【例3-91】 利用ezmesh函数和ezsurf函数绘制三维曲面图和三维表面图。

```
>> syms x y;
>> ezmesh(x * exp(x^3 + y^2),[ - 2.5,2.5]);
>> ezsurf(x * exp(x^3 + y^2),[ - 2.5,2.5]);
```

输出图像如图3-6所示。

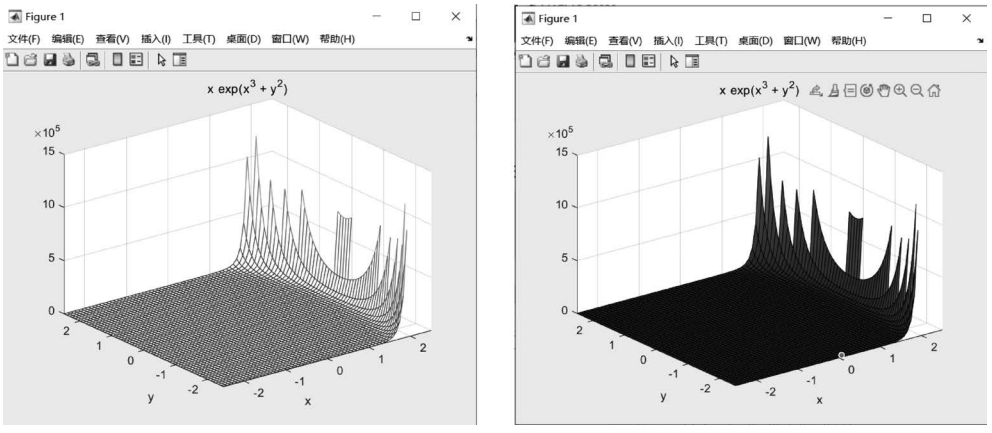

图3-6 三维曲面图和三维表面图

2. ezmeshc函数和ezsurfc函数

ezmeshc函数用于绘制带等值线的三维曲面图,ezsurfc函数用于绘制带等值线的三维表面图,两个函数的用法类似。ezmeshc函数的具体用法如下:

ezmeshc(f):在默认区域[$-2\pi<x<2\pi$,$-2\pi<y<2\pi$]绘制二元函数f(x,y)的图像。

ezmeshc(f,domain):在指定区域绘制二元函数f(x,y)的图像。

ezmeshc(x,y,z):在默认区域[$-2\pi<s<2\pi$,$-2\pi<t<2\pi$]绘制三维参数方程x=x(s,t),y(s,t),z=z(s,t)的图像。

ezmeshc(x,y,z,[smin,smax,tmin,tmax])或ezmeshc(x,y,z,[min,max]),在指定区域绘制三维参数方程的图像。

【例3-92】 使用ezmeshc函数和ezsurfc函数绘制带等值线的三维曲面图和三维表面图。

```
>> syms x y;
>> ezmeshc(x * exp(x^3 + y^2),[ - 2.5,2.5]);
>> syms x y;
>> ezsurfc(x * exp(x^3 + y^2),[ - 2.5,2.5]);
```

输出图形如图 3-7 所示。

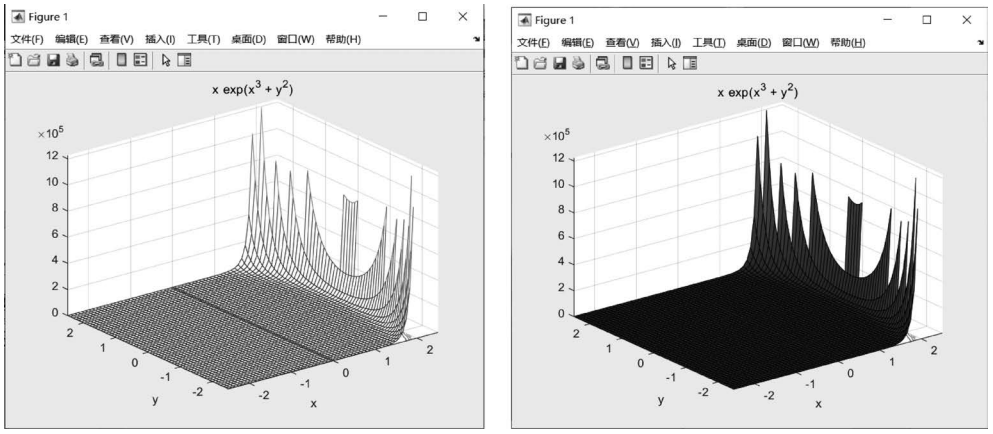

图 3-7 带等值线的三维曲面图和三维表面图

3.7 综合实例 2：求长方体体积

【例 3-93】 已知长方体的表面积为 $12a^2(a>0)$，求体积最大的长方体的长、宽、高，并计算出其体积。

根据题中要求可知，设长方体的长为 x，宽为 y，高为 z；所求的目标函数为 $f(x,y,z)=xyz$；限制条件为 $g(x,y,z)=2(xy+xz+yz)=12a^2$，即 $\mu(x,y,z)=2(xy+xz+yz)-12a^2=0$；引入拉格朗日乘子 λ，构造拉格朗日函数 $L(x,y,z)=f(x,y,z)+\lambda[2(xy+xz+yz)-12a^2]$。

根据以上分析，输入代码。其中 λ 使用 s 表示。

```
>>  syms x y z s a;
>> L = x * y * z + s * (2 * y * z + 2 * z * x + 2 * x * y - 12 * a^2) ;        % 建立拉格朗日方程
>> Lx = diff(L,'x')                                                           % 对 x 求导
Lx =
    y * z + s * (2 * y + 2 * z)
>> Ly = diff(L,'y')                                                           % 对 y 求导
Ly =
    x * z + s * (2 * x + 2 * z)
>> Lz = diff(L,'z')                                                           % 对 z 求导
Lz =
    x * y + s * (2 * x + 2 * y)
Ls = diff(L,'s')
Ls =
    - 12 * a^2 + 2 * x * y + 2 * x * z + 2 * y * z
>> [s x y z] = solve(Lx,Ly,Lz,Ls)
s =
    - (2^(1/2) * a)/4
    (2^(1/2) * a)/4
x =
    2^(1/2) * a
    - 2^(1/2) * a
y =
    2^(1/2) * a
```

```
     -2^(1/2)*a
z =
    2^(1/2)*a
     -2^(1/2)*a
```

根据以上结果,且 x、y、z 应都大于零,因此在表面积固定的情况下,长方体是存在最大体积的,即 x=y=z=$\sqrt{2}$a 时,长方体的体积最大。输入以下代码求出长方体体积。

```
>> V=x.*y.*z
V =
    2*2^(1/2)*a^3
```

3.8 本章小结

本章首先介绍了 MATLAB 的符号运算,它是对未赋值的符号对象(可以是常数、变量、表达式)进行运算和处理。数值型运算会在运算过程中产生舍入误差,而符号运算在运算过程中不会出现数值型运算,不存在舍入误差问题。然后介绍了符号表达式、运算符号运算精度和符号矩阵的计算;最后介绍了符号函数的图形绘制和符号方程的求解。通过本章的学习,读者应初步掌握符号运算的方式和使用方法。符号运算与数值运算一样,都是科学研究中的重要内容。运用符号运算可以轻松解决许多公式和关系式的推导问题。

3.9 习题

(1) 符号运算与数值运算的区别是什么?

(2) 求矩阵 $\boldsymbol{A}=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$ 的行列式值、非共轭转置和特征值。

(3) 符号表达式 $f=2x^2+3x+4$ 与 $g=5x+6$ 的代数运算。

(4) 对表达式 $2\sqrt{5}+\pi$ 进行任意精度控制的比较。

(5) 求微分方程 $x\dfrac{\mathrm{d}^2y}{\mathrm{d}x^2}-3\dfrac{\mathrm{d}y}{\mathrm{d}x}=x^2$,$y(1)=0$,$y(0)=0$ 的解。