第3章 符号与多项式运算

在解决工程技术问题时，通常首先要建立问题的数学模型，然后设计合理的算法，并通过计算机计算得到的问题解。MATLAB 提供了丰富的用于数值计算和处理的函数，并且具有良好的数值稳定性，计算速度快、精度高、效率高。

3.1 符号及其运算

MATLAB 中提供了强大的符号运算功能，可以按照推理解析的方法进行运算。MATLAB 的符号运算的功能是建立在数学计算软件 Maple 基础上的。值得注意的是，在符号运算的整个计算过程中，所有的运算均是以符号进行的，即使以数字形式出现的量也是字符量。

3.1.1 字符型数据变量的创建

在 MATLAB 工作空间中，字符型数据变量同数值型变量一样是以矩阵形式进行保存的。它的语法格式为

```
var = 'expression'
```

【例 3-1】 字符型数据变量的创建。

其 MATLAB 代码编程如下：

```
>> C = 'China'
C =
China
>> A = 'a + b - c - d'
A =
a + b - c - d
>> B = 'MATLAB Mathwork'
B =
MATLAB Mathwork
>> Z = '1 + sin(2)/3'
Z =
1 + sin(2)/3
```

应用 size 函数可以检查前面几个字符变量的大小：

```
>> S = size(C)
S =
     1   5
>> S2 = size(A)
S2 =
     1   7
>> S3 = size(B)
S3 =
     1   15
>> S4 = size(Z)
S4 =
     1   10
```

上例的结果充分说明了字符型变量是以矩阵的形式存储在 MATLAB 的工作空间内的。

3.1.2 符号型数据变量的创建

作为符号对象的符号常量、符号变量、符号函数及符号表达式，可以使用 sym 及 syms 函数规定和创建，利用 class 函数可以测试建立的操作对象为何种操作对象类型及是否为符号对象类型。

1. sym 函数

sym 函数可以生成单个的符号变量，其调用格式为

var = sym('var')：创建一个符号变量 var。

var = sym('var',set)：创建一个符号变量 var，并设置符号对象的格式。

- 'positive'：限定 var 表示正的实型符号变量。
- 'real'：限定 var 为实型符号变量。
- 'unreal'：限定 var 为非实型符号变量。

sym('var','clear')：清除先前设置的符号变量 var。

Num = sym(Num)：将一个数值转换为符号形式。

Num = sym(Num,flag)：将一个数值转换为符号形式。输入参数 flag 为转换的符合对象应该符合的格式类型，flag 可以有如下选择。

- 'r'：最接近有理表示，为系数默认设置。
- 'e'：带估计误差的有理表示。
- 'f'：十六进制浮点表示。
- 'd'：最接近的十进制浮点精确表示。

A = sym('A',dim)：创建一个矢量或矩阵的符号变量。

A = sym(A,set)：创建一个符号矩阵，set 用于设置矩阵的维数。

sym(A,'clear')：清除前面已创建的符号矩阵 A。

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

【例 3-2】 利用 sym 函数创建符号对象。

其 MATLAB 代码编程如下：

```
>> sqrt(3)
ans =
    1.7321
>> a = sqrt(sym(3))
a =
3^(1/2)
>> double(a)
ans =
    1.7321
>> sym(3)/sym(5)
ans =
3/5
>> 3/5 + 6/7
ans =
    1.4571
>> sym(3)/sym(5) + sym(6)/sym(7)
ans =
51/35
```

以上可以看出字符型变量与其他数据类型的区别。

2. syms 函数

syms 函数的功能比 sym 函数要更为强大，它可以一次创建任意多个符号变量。而且 syms 函数的使用格式也很简单，其调用格式为

syms var1…varN：创建符号变量 var1…varN。

syms var1…varN set：创建符号变量 var1…varN，并指定符号对象的格式。

- 'positive'：限定 var 表示正的实型符号变量。
- 'real'：限定 var 为实型符号变量。

syms var1…varN clear：清除前面已指定的符号对象 var1…varN。

syms f(arg1,…,argN)：创建符号函数 f，函数中包含符号变量 arg1,…,argN。

【例 3-3】 利用 syms 函数创建符号表达式。

其 MATLAB 代码编程如下：

```
>> syms s(t) f(x,y)
>> f(x,y) = x + 2*y
f(x, y) =
x + 2*y
>> f(1,2)
ans =
5
```

```
>> whos
  Name      Size      Bytes Class     Attributes
  A         1x7          14 char
  B         1x15         30 char
  C         1x5          10 char
  S         1x2          16 double
  S2        1x2          16 double
  S3        1x2          16 double
  S4        1x2          16 double
  Z         1x10         20 char
  a         1x1          60 sym
  ans       1x1          60 sym
  f         1x1          60 symfun
  s         1x1          60 symfun
  t         1x1          60 sym
  x         1x1          60 sym
  y         1x1          60 sym
```

此外，利用 sym 及 syms 函数也可以创建符号矩阵。

【例 3-4】 创建符号矩阵。

其 MATLAB 代码编程如下：

```
>> clear all;
>> m1 = [1,2 + x,1;3 - x,1,4 + y;1,2 + y,0]
m1 =
[      1, x + 2,      1]
[ 3 - x,      1, y + 4]
[      1, y + 2,      0]
>> m2 = sym('[[1,2 + x,1;3 - x,1,4 + y;1,2 + y,0]]')
m2 =
[ [1, x + 2, 1], [3 - x, 1, y + 4], [1, y + 2, 0]]
>> f(x) = [x x^3; x^2 x^4]
f(x) =
[   x, x^3]
[ x^2, x^4]
>> f(2)
ans =
[ 2,  8]
[ 4, 16]
>> y = f([1 2 3; 4 5 6])
y =
    [2x3 sym]    [2x3 sym]
    [2x3 sym]    [2x3 sym]
>> y{1}
ans =
[ 1, 2, 3]
[ 4, 5, 6]
>> y{2}
ans =
[  1,  4,  9]
```

```
[ 16, 25, 36]
>> y{3}
ans =
[  1,   8,  27]
[ 64, 125, 216]
```

3.1.3 符号计算的运算符与函数

MATLAB采用全新的数据结构、面向对象编程和重载技术，使得符号计算和数值计算在形式和风格上浑然统一，符号计算表达式的运算符和基本函数，无论在形状、名称上，还是在使用方法上，都与数值计算中的运算符和基本函数近乎相同。

1. 算术运算符号

(1) 运算符号“+”、“-”、“*”、“\”、“/”、“^”分别实现符号矩阵的加法、减法、乘法、左除、右除和求幂运算。

【例 3-5】 符号的算术运算。

其MATLAB代码编程如下：

```
>> clear all;
>> A = sym('[x^2 3; 4*x cos(x)]');
>> B = sym('[1/x^2 2*x;3 x^2+x]');
>> C = A + B                          %符号矩阵的加法运算
C =
[ 1/x^2 + x^2,          2*x + 3]
[     4*x + 3, x + cos(x) + x^2]
>> D = A - B                          %符号矩阵的减法运算
D =
[ x^2 - 1/x^2,          3 - 2*x]
[     4*x - 3, cos(x) - x - x^2]
>> E = A * B                          %符号矩阵的乘法运算
E =
[             10,      2*x^3 + 3*x^2 + 3*x]
[ 3*cos(x) + 4/x, 8*x^2 + cos(x)*(x^2 + x)]
>> F = A/B                            %符号矩阵的右除运算
F =
[          (x*(x^4 + x^3 - 9))/(- 6*x^2 + x + 1),   -(2*x^5 - 3)/(x*(- 6*
x^2 + x + 1))]
[ (4*x^3 - 3*x*cos(x) + 4*x^4)/(- 6*x^2 + x + 1), (cos(x) - 8*x^4)/(x*(- 6*
x^2 + x + 1))]
>> J = A\B                            %符号矩阵的左除运算
J =
[ (cos(x) - 9*x^2)/(x^4*cos(x) - 12*x^3), -(3*x - 2*cos(x) + 3)/(x*cos(x) - 12)]
[      (3*x^3 - 4)/(x^3*cos(x) - 12*x^2),  (x*(x^2 + x - 8))/(x*cos(x) - 12)]
>> K = A^2                            %符号矩阵的幂运算
K =
[         x^4 + 12*x, 3*cos(x) + 3*x^2]
[ 4*x*cos(x) + 4*x^3,  cos(x)^2 + 12*x]
```

```
>> M = exp(B)                          % 符号的指数运算
M =
[ exp(1/x^2),      exp(2*x)]
[      exp(3), exp(x^2 + x)]
```

(2) 运算符号".*"、".\"、"./"、".^"分别实现"元素对元素"的数组乘法、左除、右除和求幂运算。

【例 3-6】 符号矩阵的点乘也点除运算。

其 MATLAB 代码编程如下：

```
>> clear all;
>> syms a b c d e f g h;
>> A = sym('[a,b;c,d]')
A =
[ a, b]
[ c, d]
>> B = sym('[e,f;g,h]')
B =
[ e, f]
[ g, h]
>> R = A*B
R =
[ a*e + b*g, a*f + b*h]
[ c*e + d*g, c*f + d*h]
>> R2 = A.*B
R2 =
[ a*e, b*f]
[ c*g, d*h]
>> R3 = A./B
R3 =
[ a/e, b/f]
[ c/g, d/h]
>> R4 = A.\B
R4 =
[ e/a, f/b]
[ g/c, h/d]
```

(3) 运算符号"'"、".'"分别实现符号矩阵的共轭转置和非共轭转置。

【例 3-7】 符号矩阵的转置运算。

其 MATLAB 代码编程如下：

```
>> syms a b c d;
>> A = sym('[a b;c d]');
>> R1 = A'
R1 =
[ conj(a), conj(c)]
[ conj(b), conj(d)]
>> R2 = A.'
R2 =
[ a, c]
[ b, d]
```

2. 关系运算符号

与数值计算中的关系运算符号相区别的是，符号计算中的关系运算符只有以下两种。

(1) 运算符号“==”表示对运算符两边的符号对象进行“相等”的比较，返回值为“1”表示相等，返回值为“0”表示不相等。

(2) 运算符号“~=”表示对运算符号两边的符号对象进行“不相等”的比较，返回值为“1”表示不相等，返回值为“0”表示相等。

3. 复数函数

复数函数包括复数的共轭(conj)、实部(real)、虚部(imag)和模(abs)函数，在数值计算和符号计算中用法都是一样的。

4. 矩阵代数函数

在符号计算中，常用的矩阵代数函数包括：diag 函数、triu 函数、tril 函数、inv 函数、det 函数、rank 函数、rref 函数、null 函数、colspace 函数、poly 函数、expm 函数、eig 函数和 svd 函数。

除了 svd 函数的使用方法有所不同外，其余函数的用法与数值计算一致。

【例 3-8】 符号矩阵的 SVD 分解。

其 MATLAB 代码编程如下：

```
>> clear all;
>> f = sym('[1 2 1;2 3 5;1 7 9]')
f =
[ 1, 2, 1]
[ 2, 3, 5]
[ 1, 7, 9]
>> [U,S,V] = svd(f)
U =
[ 0.16282766200529790566464473003047, 0.36344034057570734984922459430431,
0.91727764135407896387284636832453]
[ 0.46068074153566191523968894576701, 0.79411905473576600135312190977843,
-0.39641919893431769076779069408513]
[ 0.87250238215379167379020797827246, -0.48712015534956807129629192786 34,
0.03812541656390840381793064786 6644]
S =
[ 13.091328222514682899973119408907, 0,                                   0]
[ 0, 1.495927099897267235054653190 9101,                                  0]
[0,                                   0, 1.17444773497312541004873731172 68]
V =
[ 0.14946470625228561177821440226895,    0.97903052548366146592839239860 05,
0.13841796038124413606394471745808]
[ 0.59697641758407588719014824902336,   -0.20094778823707901087658331851 68,
0.77668471289906891033206501338361]
[ 0.78821282559560081306327595918 61, -0.03345469434641044894661071052 3496,
```

```
 - 0.61449273794959304507931576957863]
>> syms x y u v;
>> A = sym('[x,y;u,v]')
A =
[ x, y]
[ u, v]
>> [U,S,V] = svd(A)
错误使用 sym/svd (line 84)
Input arguments must be convertible to floating - point numbers.
```

提示：符号工具箱仅支持元素为符号常量的符号矩阵的 SVD 分解。

3.1.4 寻找符号变量

MATLAB 中的符号对象可以表示符号常量和符号变量。findsym 函数可以帮助用户查找一个符号表达式中的符号变量。函数的调用格式为

findsym(s)：寻找符号表达式 s 中所有的符号变量。

findsym(s,n)：寻找符号表达式中 n 个在字母表中与 x 最接近的变量。

【例 3-9】 利用 findsym 寻找符号表达式中的变量。

其 MATLAB 代码编程如下：

```
>> clear all;
>> syms a b x y;
>> f = a^2 + 6 * b + cos(x - 2) + log(5 + y) + 4 - 5i     % 符号表达式
f =
6 * b + cos(x - 2) + log(y + 5) + a^2 + 4 - 5 * i
>> findsym(f)                                             % 查找符号变量
ans =
a,b,x,y
>> findsym(f,2)
ans =
x,y
>> findsym(f,3)
ans =
x,y,b
>> findsym(f,4)
ans =
x,y,b,a
```

3.1.5 符号精度计算

符号计算的一个非常显著的特点是：由于计算过程中不会出现舍入误差，从而可以得到任意精度的数值解。如果希望计算结果精确，那么就可以牺牲计算时间和存储空间，用符号计算来获得足够高的计算精度。

在符号运算工具箱中有 3 种不同类型的算术运算。

- 数值类型：MATLAB 的浮点算术运算。
- 有理数类型：Maple 的精度符号计算。
- VPA 类型：Maple 的任意精度算术运算。

这 3 种运算各有利弊，在计算时应根据计算精度、时间、存储空间的要求进行合理的选择。浮点运算是最快的运算，需要的计算机内存最小，但是结果不精确。MATLAB 双精度数输出的数字位数由 format 函数控制，但它内部采用的是由计算机硬件提供的 8 位浮点数的表示方法。

符号运算中的有理数算术运算所需要的时间和内存开销都是很大的。只要有足够的内存和足够长的计算时间，总能产生精确的结果。一般符号计算的结果都是字符串，特别是一些符号计算结果从形式上来看是数值，但从变量类型上看，它们仍然是字符串。要从精确解中获得任意精度的解，并改变默认精度，把任意精度符号解变成“真正的”数值解，就如下几个函数。

(1) digits 函数

digits 函数用于设定所用数值的精度。函数的调用格式为

digits(d)：符号对象的近似解的精度变为 d 位有效数字，参数 d 的默认值为 32 位。

d = digits：得到当前采用的数值计算的精度。

【例 3-10】 利用 digits 函数进行精度控制。

其 MATLAB 代码编程如下：

```
>> clear all;
>> digits                                          % 默认精度
Digits = 32
>> a1 = sym(1.6,'d')
a1 =
1.600000000000000008881784197001252323389053
>> digits(42)                                      % 设置精度为 42
>> digits
Digits = 42
>> a2 = sym(1.6,'d')
a2 =
1.600000000000000008881784197001252323389053
```

在程序中，利用 digits 函数进行数值精度的设置。系统默认值为 32 位，程序中将其改为 42 位，并进行输出。

(2) vpa 函数

vpa 函数用于进行可控精度运算。函数的调用格式为

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

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

【例 3-11】 利用函数 vpa 进行精度控制。

其 MATLAB 代码编程如下：

```
>> clear all;
>> a = vpa(hilb(2))                                % 系统默认
```

```
a =
[ 1.0,                                    0.5]
[ 0.5, 0.33333333333333333333333333333333]
>> b = vpa(hilb(3),6)                                  % 设置为 6 位
b =
[     1.0,     0.5, 0.333333]
[     0.5, 0.333333,   0.25]
[ 0.333333,     0.25,    0.2]
>> c = vpa(pi)                                         % 默认
c =
3.1415926535897932384626433832795
>> c = vpa(pi,88)                                      % 设置为 88 位
c =
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803 5
```

3.1.6 显示符号表达式

在符号表达式的显示过程中,默认采用 MATLAB 式的显示。除了默认的显示方式外,MATLAB 符号工具箱中提供了显示函数 pretty,允许用户将符号表达式显示为符合一般数学表达习惯的数学表达式。

pretty(X):将符号表达式用书写方式显示出来,使用默认的宽。

【例 3-12】 利用 pretty 函数显示符号表达式。

其 MATLAB 代码编程如下:

```
>> syms x
s = solve(x^4 + 2*x + 1, x,'MaxDegree',3);
pretty(s)
/          -1          \
|                      |
|           2    1     |
|  #2 ----- + -        |
|         9 #2   3     |
|                      |
|   1          #2   1  |
| ----- #1 --- + -     |
| 9 #2          2   3  |
|                      |
|        1    #2   1   |
| #1 + ------- + -     |
\      9 #2   2   3    /
where
                    /   2         \
         sqrt(3) | ---- + #2 | 1i
                    \9 #2         /
   #1 == ------------------
                       2
            /sqrt(11) sqrt(27)    17\1/3
   #2 == | ------------ |
            \          27           27 /
```

3.1.7 合并符号表达式

在MATLAB中，提供了collect函数用于实现将符号表达式中的同类项进行合并。函数的调用格式为

R = collect(S)：将表达式S中相同的次幂的项合并，系统默认为按照x的相同次幂项进行合并。

R = collect(S,v)：将表达式S按照v的相同次幂项进行合并。输入参数S可以是一个表达式，也可以是一个符号矩阵。

【例3-13】 利用collect函数对符号表达式进行合并同类项。

其MATLAB代码编程如下：

```
>> syms x y
>> collect((exp(x) + x) * (x + 2))
ans =
x^2 + (exp(x) + 2) * x + 2 * exp(x)
>> collect(x^2 * y + y * x - x^2 - 2 * x, x)
ans =
(y - 1) * x^2 + (y - 2) * x
>> collect(x^2 * y + y * x - x^2 - 2 * x, y)
ans =
(x^2 + x) * y - x^2 - 2 * x
>> collect(2 * x * i - 3 * i * y, i)
ans =
(2 * x - 3 * y) * 1i
>> collect(x * pi * (pi - y) + x * (pi + i) + 3 * pi * y, pi)
ans =
x * pi^2 + (x + 3 * y - x * y) * pi + x * 1i
```

3.1.8 展开符号表达式

在MATLAB中，提供了expand函数用于实现将符号表达式展开。函数的调用格式为

expand(S)：表达式S中如果包含函数，MATLAB会利用恒等式变形将其写成相应的形式。这个展开函数主要用于多项式、三角函数、指数函数和对数函数。默认x为第一变量。

expand(S,Name,Value)：设置展开式的参数名Name及其对应的参数值Value。

【例3-14】 利用expand函数对符号表达式进行展开。

其MATLAB代码编程如下：

```
>> syms x y a b c t
>> expand((x - 2) * (x - 4))
ans =
x^2 - 6 * x + 8
>> expand(cos(x + y))
ans =
```

```
cos(x) * cos(y) - sin(x) * sin(y)
>> expand(exp((a + b)^2))
ans =
exp(a^2) * exp(b^2) * exp(2 * a * b)
>> expand([sin(2 * t), cos(2 * t)])
ans =
[ 2 * cos(t) * sin(t), 2 * cos(t)^2 - 1]
>> expand((sin(3 * x) - 1)^2, 'ArithmeticOnly', true)
ans =
sin(3 * x)^2 - 2 * sin(3 * x) + 1
>> expand(log((a * b/c)^2))
ans =
log((a^2 * b^2)/c^2)
>> expand(log((a * b/c)^2), 'IgnoreAnalyticConstraints', true)
ans =
2 * log(a) + 2 * log(b) - 2 * log(c)
```

3.1.9 嵌套符号表达式

在 MATLAB 中,提供了 horner 函数实现对符号表达式进行嵌套。函数的调用格式为

R=horner(P):P 为待嵌套的符号表达式,R 为嵌套后的符号表达式。

【例 3-15】 利用 horner 函数实现符号表达式的嵌套。

其 MATLAB 代码编程如下:

```
>> syms x y
>> horner(x^3 - 6 * x^2 + 11 * x - 6)
ans =
x * (x * (x - 6) + 11) - 6
>> horner([x^2 + x; y^3 - 2 * y])
ans =
   x * (x + 1)
 y * (y^2 - 2)
```

3.1.10 分解符号表达式

在 MATLAB 中,提供了 factor 函数用于实现符号表达式的分解。函数的调用格式为

factor(X):X 为多项式或多项式矩阵,系数是有理数,MATLAB 会将表达式 X 表示成为有理数的低阶多项式相乘的形式;如果多项式 X 不能进行在有理数范围内因式分解,则会返回 X 本身。

【例 3-16】 利用 factor 对符号表达式进行分解。

其 MATLAB 代码编程如下:

```
>> syms x y
factor(x^3 - y^3)
```

```
ans =
(x - y) * (x^2 + x * y + y^2)
>> syms a b
factor([a^2 - b^2, a^3 + b^3])'
ans =
                                (conj(a) - conj(b)) * (conj(a) + conj(b))
(conj(a) + conj(b)) * (conj(a)^2 + conj(b)^2 - conj(a) * conj(b))
>> factor(sym('12345678901234567890'))
ans =
2 * 3^2 * 5 * 101 * 3541 * 3607 * 3803 * 27961
```

在程序中，利用 factor 函数进行符号多项式的因式分解，系数按照从小到大的顺序进行排列。

3.1.11 化简符号表达式

MATLAB 根据一定的规则对符号表达式进行简化，简化的函数为 simplify。函数的调用格式为

simplify(S)：输入参数 S 为符号表达式或符号矩阵。

simplify(S,Name,Value)：指定一个符号表达式的属性名及对应的属性值，并对表达式或矩阵进行化简。

【例 3-17】 利用 simplify 函数对符号表达式或矩阵进行化简。

其 MATLAB 代码编程如下：

```
>> clear all;
% 对符号表达式进行化简
syms x a b c
simplify(sin(x)^2 + cos(x)^2)
ans =
1
>> simplify(exp(c * log(sqrt(a + b))))
ans =
(a + b)^(c/2)
>> % 对符号矩阵进行化简
simplify([(x^2 + 5 * x + 6)/(x + 2), sin(x) * sin(2 * x) + cos(x) * cos(2 * x);(exp( - x *
i) * i)/2 - (exp(x * i) * i)/2, sqrt(16)])
ans =
[ x + 3, cos(x)]
[ sin(x),      4]
>> % 使用 IgnoreAnalyticConstraints 规则对符号表达式进行化简
s = (log(x^2 + 2 * x + 1) - log(x + 1)) * sqrt(x^2);
simplify(s)
ans =
-(log(x + 1) - log((x + 1)^2)) * (x^2)^(1/2)
>> simplify(s, 'IgnoreAnalyticConstraints', true)
ans =
x * log(x + 1)
>> f = ((exp( - x * i) * i)/2 - (exp(x * i) * i)/2)/(exp( - x * i)/2 + exp(x * i)/2);
```

```
simplify(f)
ans =
-(exp(x*2*i)*i - i)/(exp(x*2*i) + 1)
f = (exp(x + exp(-x*i)/2 - exp(x*i)/2)*i)/2 - (exp(- x - exp(-x*i)/2 + exp(x*i)/
2)*i)/2;
simplify(f, 'Criterion','preferReal', 'Steps', 100)     % 化简 100 步
ans =
cos(sin(x))*sinh(x)*i + sin(sin(x))*cosh(x)
```

3.1.12 替换符号表达式

在处理一些结构较为复杂、变量较多的数学模型时，引入一些新的变量进行代换，以简化其结果，从而达到解决问题的目的，这种方法称为变量代换法。

例如，求不定积分$\int \frac{1}{t(t^7+2)}\mathrm{d}t$，设 $x=\frac{1}{t}$，则

$$\int \frac{1}{t(t^7+2)}\mathrm{d}t=-\int \frac{x^6}{1+2x^7}\mathrm{d}x=-\frac{1}{14}\ln|1+2t^7|+c$$

$$=-\frac{1}{14}\ln|2+t^7|+\frac{1}{2}\ln|t|+c$$

变量代换法是一种非常有效的解题方法，尤其是处理一些复杂的不等式问题时，效果明显。合理代换往往能简化题目的信息，凸显隐含条件，沟通量与量之间的关系，对发现解题思路、优化解题过程有着重要的作用。

MATLAB 提供了 subs 函数和 subexpr 函数进行变量代换或者叫作符号表达式的替换。subs 函数利用符号变量或符号表达式替换目标符号表达中的符号变量(包括符号常量)；subexpr 函数利用符号变量替换目标符号表达式的某个子符号表达式。

1. subs 函数

subs 函数可以用指定符号替换符号表达式中的某一特定符号。函数的调用格式为

R = subs(S)：用函数中的值或 MATLAB 工作区间的值替代符号表达式 S 中的所有变量，如果没有指定某符号变量的值，则返回值中的该符号变量不被替换。

R = subs(S, new)：用新的符号变量 new 代替原来的符号表达式 S 中的默认变量。

R = subs(S, old, new)：用新的符号变量 new 替换原来符号表达式 S 中的变量 old，当 new 是数值形式的符号时，实际上用数值替换原来的符号计算表达式的值，结果仍为字符串形式。

【例 3-18】 利用 subs 函数对符号表达式进行替换。

其 MATLAB 代码编程如下：

```
>> syms a b;
subs(a + b, a, 4)
ans =
b + 4
>> syms a b;
subs(cos(a) + sin(b), {a, b}, {sym('alpha'), 2})
```

```
ans =
sin(2) + cos(alpha)
>> syms t;
subs(exp(a * t), 'a', -magic(2))
ans =
[ 1/exp(t), 1/exp(3 * t)]
[ 1/exp(4 * t), 1/exp(2 * t)]
>> syms x y;
subs(x * y, {x, y}, {[0 1; -1 0], [1 -1; -2 1]})
ans =
     0   -1
     2    0
```

2. subexpr 函数

subexpr 函数将表达式中重复出现的字符串用变量代换。函数的调用格式为

[Y,SIGMA] = subexpr(X,SIGMA)：指定用变量 SIGMA 的值，来代替符号表达式中重复出现的字符串。进行符号替换后，函数的返回值为 Y，被替换的符号变量由 SIGMA 返回。

[Y,SIGMA] = subexpr(X,'SIGMA')：函数中输入参数 SIGMA 为字符或字符串，用来替换符号表达式中重复出现的字符串。进行符号替换返回值为 Y，被替换的符号变量由 SIGMA 返回。

【例 3-19】 利用 subexpr 函数对符号表达式进行替换。

```
>> syms a b c d x
solutions = solve(a * x^3 + b * x^2 + c * x + d == 0, x, 'MaxDegree', 3);
>> [r, sigma] = subexpr(solutions)
r = sigma^(1/3) - b/(3 * a) - (-b^2/(9 * a^2) + c/(3 * a))/sigma^(1/3)
(-b^2/(9 * a^2) + c/(3 * a))/(2 * sigma^(1/3)) - sigma^(1/3)/2 - (3^(1/2) * (sigma^
(1/3) + (-b^2/(9 * a^2) + c/(3 * a))/sigma^(1/3)) * 1i)/2 - b/(3 * a)
(-b^2/(9 * a^2) + c/(3 * a))/(2 * sigma^(1/3)) - sigma^(1/3)/2 + (3^(1/2) * (sigma^
(1/3) + (-b^2/(9 * a^2) + c/(3 * a))/sigma^(1/3)) * 1i)/2 - b/(3 * a)
sigma =
((d/(2 * a) + b^3/(27 * a^3) - (b * c)/(6 * a^2))^2 + (-b^2/(9 * a^2) + c/(3 * a))^3)
^(1/2) - b^3/(27 * a^3) - d/(2 * a) + (b * c)/(6 * a^2)
>> solutions = solve(a * x^2 + b * x + c == 0, x)
solutions =
-(b + (b^2 - 4 * a * c)^(1/2))/(2 * a)
-(b - (b^2 - 4 * a * c)^(1/2))/(2 * a)
>> [abbrSolutions,s] = subexpr(solutions,s)
abbrSolutions =
-(b + s)/(2 * a)
-(b - s)/(2 * a)
s =
(b^2 - 4 * a * c)^(1/2)
```

3.1.13 符号函数的操作

MATLAB 具有对符号表达式执行更高级运算的功能。MATLAB 提供了把两个符

号函数复合成一个符号函数的功能函数，同时也提供了对符号函数求函数表达式的逆功能函数等。

1. compose 函数

在 MATLAB 中，提供了 compose 函数用于将符号表达式的复合函数运算。函数的调用格式为

compose(f,g)：返回函数当 f=f(x)和 g=g(y)时的复合函数 f(g(y))。其中，x 为由函数 findsym 确定的 f 的符号变量，y 为由函数 findsym 确定的 g 的符号变量。

compose(f,g,z)：返回 f=f(x)和 g=g(y)时的复合函数 f(g(z))，返回的函数以 z 为自变量。例如，如果 f=sin(x/t)，那么函数 compose(f,g,z)将返回 sin(g(z)/f)。

compose(f,g,x,z)：返回复合函数 f(g(z))，x 为函数 f 的独立变量。例如，如果 f=sin(x/t)，那么函数 compose(f,g,t,z)将返回 sin(g(z)/f)，并且函数 compose(f,g,t,z)将返回 sin(x/g(z))。

compose(f,g,x,y,z)：返回复合函数 f(g(z))，并且 x 为函数 f 的独立变量，y 为函数 g 的独立变量。例如，如果 y=sin(x/t)，并且 g=cos(y/u)，那么函数 compose(f,g,x,y,z)将返回 sin(cos(z/u)/t)，并且函数 compose(f,g,x,u,z)将返回 sin(cos(y/z)/t)。

【例 3-20】 利用 compose 函数对符号表达式进行复合操作。

其 MATLAB 代码编程如下：

```
>> clear all;
>> syms x y z t u
f = 1/(1 + x^2);
g = sin(y);
h = x^t;
p = exp( - y/u);
>> a = compose(f,g)
a =
1/(sin(y)^2 + 1)
>> b = compose(f,g,t)
b =
1/(sin(t)^2 + 1)
>> c = compose(h,g,x,z)
c =
sin(z)^t
>> d = compose(h,g,t,z)
d =
x^sin(z)
>> e = compose(h,p,x,y,z)
e =
exp( - z/u)^t
>> f = compose(h,p,t,u,z)
f =
x^exp( - y/z)
```

2. finverse 函数

在 MATLAB 中，提供了 finverse 函数用于实现符号表达式的反函数操作。函数的

调用格式为

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

g = finverse(f,var)：输入参数 var 为一个符号变量，是函数 f 中的变量，且该函数的返回值 g 是以 var 为自变量。符号表达式 g 必须满足 g(f(var))=var。

【例 3-21】 利用 finverse 函数求符号表达式的反函数。

其 MATLAB 代码编程如下：

```
>> clear all;
>> syms t x
>> f1 = finverse(log(t))                % 反函数,以默认 x 为自变量
f1 =
exp(t)
>> f2 = finverse(cos(2 * t) + 1)
f2 =
acos(t - 1)/2
>> f3 = finverse(exp(t - 2 * x),t)      % 以 t 为自变量
f3 =
2 * x + log(t)
>> f4 = finverse(exp(x - 2 * t), x)
f4 =
2 * t + log(x)
```

提示：当函数 finverse 求得的解不唯一时，MATLAB 会给出警告信息。

3.1.14 符号微积分

微积分运算在数学计算中的重要性是不言而喻的，整个高等数学就是建立在微积分运算的基础上的。

在符号数学工具箱中提供了一些常用的函数来支持具有重要基础意义的微积分运算，涉及微分、求极限、积分、级数求和泰勒级数等。

1. 符号表达式的极限

求微分的基本思想是当自变量趋近某个值时，求函数值的变化。“无穷逼近”是微积分的一个基本思想，求极限是非常普遍的。事实上，导数就是由极限给出的：

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

在 MATLAB 中，用 limit 函数来求表达式的极限。函数的调用格式为

limit(expr, x, a)：求符号函数 expr(x)的极限值。即计算当变量 x 趋近于常数 a 时，expr(x)函数的极限值。

limit(expr, a)：求符号函数 expr(x)的极限值。由于没有指定符号函数 expr(x)的自变量，则使用该格式时，符号函数 expr(x)的变量为函数 findsym(expr)确定的默认自

变量，即变量 x 趋近于 a。

limit(expr)：求符号函数 expr(x)的极限值。符号函数 expr(x)的变量为函数 findsym(expr)确定的默认变量；没有指定变量的目标值时，系统默认变量趋近于 0，即 a=0 的情况。

limit(expr, x, a, 'left')：求符号函数 expr 的极限值。left 表示变量 x 从左边趋近于 a。

limit(expr, x, a, 'right')：求符号函数 expr 的极限值。right 表示变量 x 从右边趋近于 a。

函数 limit 要求第一个输入变量为符号函数，limit 不支持符号函数的句柄，但是对符号函数句柄 f，可以将 f(x)作为输入变量。

【例 3-22】 利用 limit 函数求符号表达式的极限。

其 MATLAB 代码编程如下：

```
>> clear all;
>> syms x;
>> f1 = sym((cos(x) + sin(x) - x)/x)
f1 =
(cos(x) - x + sin(x))/x
>> limit(f1,x,inf)
ans =
 -1
>> limit(f1,x,-inf)
ans =
 -1
>> limit(f1,x,0)
ans =
NaN
>> f2 = sym('(sin(x) - x)/x')
f2 =
 -(x - sin(x))/x
>> limit(f2,x,0,'right')
ans =
0
>> limit(f2,x,0,'left')
ans =
0
```

2. 符号表达式的微分

MATLAB 提供的函数可以完成一元及多元符号表达式函数的各阶微分，功能函数 diff 可以完成一元或多元函数的任意阶数的微分。对于自变量的个数多于一个的符号矩阵，微分为 Jocabian 矩阵，采用 jacobian 函数实现微分。

(1) diff 函数

当创建符号表达式后，就可以利用 diff 对其进行微分运算了。函数的调用格式为

diff(expr)：没有指定变量和导数阶数，则系统按 findsym 函数指示的默认变量对符号表达式 expr 求一阶导数。

diff(expr,v)：指定符号表达式的变量为 v。

diff(expr, sym('v'))：计算符号表达式 expr 的一阶导数，以符号变量 v 为自变量。

diff(expr,n)：对符号表达式 expr 求 n 阶导数。

diff(expr,v,n)：计算符号表达式 expr 的 n 阶导数，以符号变量 v 为自变量。

【例 3-23】 求函数 $f(x)=e^{-2x}\cos(3\sqrt{x})$ 的 4 阶导数。

其 MATLAB 代码编程如下：

```
>> syms x
f = exp( - 2 * x) * cos(3 * x ^ 0.5)
g = diff(f,4)                          % 求复合函数的 3 阶导数
```

运行程序，输出如下：

```
f =
exp( - 2 * x) * cos(3 * x ^ (1/2))
g =
16 * exp( - 2 * x) * cos(3 * x ^ (1/2)) + 48 * exp( - 2 * x) * sin(3 * x ^ (1/2))/x ^ (1/2) - 54 * exp
( - 2 * x) * cos(3 * x ^ (1/2))/x - 9 * exp( - 2 * x) * sin(3 * x ^ (1/2))/x ^ (3/2) - 351/16 * exp
( - 2 * x) * cos(3 * x ^ (1/2))/x ^ 2 - 9/8 * exp( - 2 * x) * sin(3 * x ^ (1/2))/x ^ (5/2) - 135/16 *
exp( - 2 * x) * cos(3 * x ^ (1/2))/x ^ 3 + 45/16 * exp( - 2 * x) * sin(3 * x ^ (1/2))/x ^ (7/2)
```

再执行以下代码：

```
>> pretty(g)
```

运行程序，输出如下：

```
                                                                  1/2
                          1/2          exp( - 2 x) sin(3 x    )
  16 exp( - 2 x) cos(3 x    )    + 48 ----------------------
                                                1/2
                                               x
                                 1/2                                1/2
        exp( - 2 x) cos(3 x    )            exp( - 2 x) sin(3 x    )
  - 54 ----------------------  - 9 ----------------------
                  x                                3/2
                                                  x

                                 1/2                                1/2
  351     exp( - 2 x) cos(3 x    )          exp( - 2 x) sin(3 x    )
  ---- ---------------------- - 9/8 ----------------------
   16              2                                5/2
                  x                                x

                                 1/2                                  1/2
  135     exp( - 2 x) cos(3 x    )       45   exp( - 2 x) sin(3 x    )
  ---- ---------------------- + --- ----------------------
   16              3               16               7/2
                  x                                x
```

(2) jacobian 函数

在 MATLAB 中,提供了 jacobian 函数用于求多元函数的导数。函数的调用格式为

jacobian(f,v):计算数量或向量 f 对于向量 v 的 Jacobian 矩阵。函数的返回值第 i 行第 j 列的数为 df(i)/dv(j)。当 f 为数量时,该函数返回 f 的梯度。此外,参数 v 可以是数量,jacobian(f,v)等价于 diff(f,v)。

【例 3-24】 利用 jacobian 函数求多元函数的导数。

其 MATLAB 代码编程如下:

```
>> syms x y z
f = [x*y*z; y; x + z];
v = [x, y, z];
R = jacobian(f, v)
R =
[ y*z, x*z, x*y]
[   0,   1,   0]
[   1,   0,   1]
>> b = jacobian(x + z, v)
b =
[ 1, 0, 1]
```

3. 符号表达式的积分

数学中,积分和微分是一对互逆的运算。符号数学工具箱中提供了 int 函数来求符号表达式的积分,函数的调用格式为

int(f):没有指定积分变量和积分阶数时,系统按 findsym 函数指示的默认变量对被积函数或符号表达式 f 求不定积分。

int(f,v):以 v 为自变量,对被积函数或符号表达式 f 求不定积分。

int(f, a, b):符号表达式采用默认变量,该函数求默认变量从 a 到 b 时符号表达式 f 的定积分数值。如果 f 为符号矩阵,则积分对各个元素分别进行积分。

int(f, v, a, b):a、b 分别表示定积分的下限和上限。该函数求被积函数 f 在区间[a, b]上的定积分。a 和 b 可以是两个具体的数,也可以是一个符号表达式,还可以是无穷(inf)。当函数 f 关于变量 x 在闭区间[a, b]上可积时,函数返回一个定积分结果。当 a、b 中有一个是 inf 时,函数返回一个广义积分。当 a,b 中有一个是符号表达式时,函数返回一个符号函数。

【例 3-25】 利用 int 函数对符号表达式进行积分运算。

其 MATLAB 代码编程如下:

```
>> syms a x t z
>> int(-2*x/(1 + x^2)^2)
ans =
1/(x^2 + 1)
>> int(x/(1 + z^2), x)
ans =
x^2/(2*(z^2 + 1))
>> int(x/(1 + z^2), z)
```

```
ans =
x * atan(z)
>> int(x * log(1 + x), 0, 1)
ans =
1/4
>> int(2 * x, [sin(t), 1])
ans =
cos(t)^2
>> int([exp(t), exp(a * t); sin(t), cos(t)])
ans =
[ exp(t), exp(a * t)/a]
[ -cos(t),    sin(t)]
```

4. 级数求和

在 MATLAB 中提供了 symsum 函数用于求符号表达式的和。其调用格式为：

F = symsum(f,k,a,b)：求符号表达式 f 中变量 k 从 a 变到 b 时的有限和。

F = symsum(f,k)：计算符号表达式 f 中指定变量为 k 的有限项和。

【例 3-26】 利用 symsum 函数对符号表达式进行级数求和。

其 MATLAB 代码编程如下：

```
>> syms k x
S1 = symsum(k^2, k, 0, 10)
S2 = symsum(1/k^2, k, 1, Inf)
S3 = symsum(x^k/factorial(k), k, 0, Inf)
S1 =
385
S2 =
pi^2/6
S3 =
exp(x)
>> S4 = symsum(k, k)
S5 = symsum(1/k^2, k)
S4 =
k^2/2 - k/2
S5 =
-psi(1, k)
```

5. 泰勒级数

在 MATLAB 中，提供了 taylor 函数用于求符号表达式的泰勒级数展开式。函数的调用格式为

taylor(f,var)：指定符号变量为 var，求 Maclaurin 多项式。

taylor(f,var,a)：返回符号表达式 f 中的指定符号自变量 var(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 var 可以是字符串或符号变量。

taylor(__,Name,Value)：指定一个或多个属性名及其属性值，实现对符号表达式进行求泰勒级数。

【例 3-27】 利用 taylor 函数对符号表达式进行泰勒级展开式。

其 MATLAB 代码编程如下：

```
>> syms x
taylor(exp(x))
taylor(sin(x))
taylor(cos(x))
ans =
x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
ans =
x^5/120 - x^3/6 + x
ans =
x^4/24 - x^2/2 + 1
>> taylor(log(x), x, 'ExpansionPoint', 1)
ans =
x - (x - 1)^2/2 + (x - 1)^3/3 - (x - 1)^4/4 + (x - 1)^5/5 - 1
>> taylor(acot(x), x, 1)
ans =
pi/4 - x/2 + (x - 1)^2/4 - (x - 1)^3/12 + (x - 1)^5/40 + 1/2
>> taylor(1/(exp(x)) - exp(x) + 2*x, x, 'Order', 5)
ans =
-x^3/3
>> taylor(1/(exp(x)) - exp(x) + 2*x, x, 'Order', 5, 'OrderMode', 'relative')
ans =
- x^7/2520 - x^5/60 - x^3/3
```

3.1.15 符号积分变换

在数学中，为了把较复杂的运算转换为比较简单的运算，经常采用一种变换手段。例如数量的乘积或商可以变换成对数的和或差，然后再取反对数即可求得原来数量的乘积或商。这一变换方法的目的就是把比较复杂的乘除运算通过对数变换转换为简单的加减运算。

所谓积分变换，就是通过积分运算，把一类函数 A 变换成另一类函数 B，函数 B 一般是含有参量 α 的积分：$\int_a^b f(t)K(t,\alpha)\mathrm{d}t$。这一变换的目的就是把某函数类 A 中的函数 $f(t)$ 通过积分运算变换成另一类函数 B 中的函数 $F(\alpha)$。这里 $K(t,\alpha)$ 是一个确定的二元函数，叫作积分变换的核。当选取不同的积分区间与变换核时，就成为不同的积分变换。$f(t)$ 叫作原函数，$F(\alpha)$ 叫作象函数。在一定条件下，原函数与象函数两者一一对应，成为一个积分变换对。变换是可逆的，由原函数求象函数叫作正变换，反之则是逆变换。

积分变换的理论与方法，在自然科学与工程技术的各个领域中都有着极广泛的应用，成为不可缺少的运算工具。变换的使用，会极大地简化计算，有的变换则为开创新的学科奠定了基础。

1. 傅里叶变换及反变换

时域中的 $f(t)$ 与它在频域中的 Fourier 变换 $F(\omega)$ 之间存在如下关系：

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

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

由计算机完成这种变换的途径有两条：一是直接调用 fourier 和 ifourier 进行；二是根据上面的定义，利用积分函数 int 实现。fourier 和 ifourier 函数的调用格式为

Fw＝fourier(f，t，w)：求时域上函数 f 的傅里叶变换 Fw，其中 f 是以 t 为自变量的时域函数，Fw 是以圆频率 w 为自变量的频域函数。

f＝ifourier(Fw，w，t)：求频域上函数 Fw 的傅里叶反变换 f，其中 f 是以 t 为自变量的时域函数，Fw 是以圆频率 w 为自变量的频域函数。

【例 3-28】 实现傅里叶变换及其反变换。

其 MATLAB 代码编程如下：

```
%傅里叶变换
>> syms w x y z a b c d t
>> f = exp(-x^2);
fourier(f, x, y)
ans =
pi^(1/2)*exp(-y^2/4)
>> f2 = exp(-x^2)*exp(-t^2);
fourier(f2, y)
ans =
pi^(1/2)*exp(-t^2)*exp(-y^2/4)
>> fourier(f2)
ans =
pi^(1/2)*exp(-t^2)*exp(-w^2/4)
>> fourier(t^3, t, w)
ans =
-pi*dirac(3, w)*2i
fourier(x,[x, w; y, z],[a, b; c, d])
ans =
[ pi*dirac(1, a)*2i, 2*pi*x*dirac(b)]
[   2*pi*x*dirac(c), 2*pi*x*dirac(d)]
%傅里叶反变换
F = sqrt(sym(pi))*exp(-y^2/4);
ifourier(F, y, x)
ans =
exp(-x^2)
F = exp(-w^2/(4*a^2));
ifourier(F, t)
ans =
exp(-a^2*t^2)/(2*pi^(1/2)*(1/(4*a^2))^(1/2))
ifourier([exp(x), 1; sin(y), i*z],[w, x; y, z],[a, b; c, d])
ans =
[                                  exp(x)*dirac(a),     dirac(b)]
[ (dirac(c - 1)*1i)/2 - (dirac(c + 1)*1i)/2, dirac(1, d)]
```

2. 拉普拉斯变换及反变换

Laplace 变换及其反变换的定义为

$$F(s)=\int_0^{\infty} f(t)\mathrm{e}^{-st}\,\mathrm{d}t$$

$$f(t)=\frac{1}{2\pi \mathrm{j}}\int_{c-\infty}^{c+\infty} F(s)\mathrm{e}^{st}\,\mathrm{d}s$$

与 Fourier 变换相似，Laplace 变换与反变换的实现也有两条途径：直接调用 laplace 和 ilaplace 函数；或者根据上面的定义，利用积分函数 int 实现。比较而言，直接使用 laplace 和 ilaplace 函数实现变换较为简单。函数的调用格式为

Fs＝laplace(f,trans_var,eval_point)：求时域上函数 f 的拉普拉斯变换 Fs，其中 f 是以 trans_var 为自变量的时域函数，Fs 是以复频率 eval_point 为自变量的频域函数。

f＝ilaplace(Fs,trans_var,eval_point)：求频域上函数 Fs 的拉普拉斯反变换 f，其中 f 是以 trans_var 为自变量的时域函数，Fs 是以复频率 eval_point)为自变量的频域函数。

【例 3-29】 实现拉普拉斯变换及其反变换。

其 MATLAB 代码编程如下：

```
% Laplace 变换
>> syms a b c d w x y z t s
>> f = 1/sqrt(x);
laplace(f, x, y)
ans =
pi^(1/2)/y^(1/2)
>> f = exp(-a*t);
laplace(f, y)
ans =
1/(a + y)
>> laplace(dirac(t - 3), t, s)
ans =
exp(-3*s)
>> laplace([exp(x), 1; sin(y), i*z],[w, x; y, z],[a, b; c, d])
ans =
[   exp(x)/a, 1/b]
[ 1/(c^2 + 1), 1i/d^2]
% Laplace 反变换
>> F = 1/y^2;
ilaplace(F, y, x)
ans =
x
>> F = 1/(s - a)^2;
ilaplace(F, x)
ans =
x*exp(a*x)
>> ilaplace(1, s, t)
ans =
dirac(t)
>> ilaplace(exp(-2*s)/(s^2 + 1) + s/(s^3 + 1), s, t)
```

```
ans =
heaviside(t - 2) * sin(t - 2) - exp( - t)/3 + (exp(t/2) * (cos((3 ^ (1/2) * t)/2) + 3 ^ (1/
2) * sin((3 ^ (1/2) * t)/2)))/3
>> ilaplace([exp(x), 1; sin(y), i * z],[w, x; y, z],[a, b; c, d])
ans =
[         exp(x) * dirac(a),        dirac(b)]
[ ilaplace(sin(y), y, c), dirac(1, d) * 1i]
```

3. Z 变换及其反变换

一个离散因果序列的 Z 变换及其反变换定义为

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

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

涉及 Z 变换具体计算的方法最常见的有 3 种，分别是幂级数展开法、部分分式展开法和围线积分法。MATLAB 的符号数学工具箱中采用围线积分法设计求 Z 反变换的 iztrans 函数，相应的数学表达式为

$$f(n) = \frac{1}{2\pi \mathrm{j}} \int F(z) z^{n-1} \mathrm{d}z$$

函数的调用格式如下。

Fz=ztrans(f,trans_index,eval_point)：求时域函数 f 的 Z 变换 Fz，f 是以 trans_index 为自变量的时域序列，Fz 是以复频率 eval_point 为自变量的频域函数。

f=iztrans(Fz,trans_index,eval_point)：求频域函数 Fz 的 Z 逆变换 f，f 是以 trans_index 为自变量的时域序列，Fz 是以复频率 eval_point 为自变量的频域函数。

【例 3-30】 Z 变换及其反变换。

其 MATLAB 代码编程如下：

```
% Z 变换
>> syms a b c d w x y z k t n
>> f = sin(k);
ztrans(f, k, x)
ans =
(x * sin(1))/(x ^ 2 - 2 * cos(1) * x + 1)
>> ztrans(heaviside(n - 3), n, z)
ans =
(1/(z - 1) + 1/2)/z ^ 3
>> ztrans(nchoosek(n, 2) * heaviside(5 - n), n, z)
ans =
z/(z - 1)^3 + 5/z ^ 5 + (6 * z - z ^ 6/(z - 1)^3 + 3 * z ^ 2 + z ^ 3)/z ^ 5
>> ztrans([exp(x), 1; sin(y), i * z],[w, x; y, z],[a, b; c, d])
ans =
[                      (a * exp(x))/(a - 1),        b/(b - 1)]
[ (c * sin(1))/(c ^ 2 - 2 * cos(1) * c + 1), (d * 1i)/(d - 1)^2]
% Z 反变换
>> F = 2 * x/(x - 2)^2;
iztrans(F, x, k)
ans =
```

```
2^k + 2^k*(k - 1)
>> iztrans(1/z, z, n)
ans =
kroneckerDelta(n - 1, 0)
>> iztrans((z^3 + 3*z^2 + 6*z + 5)/z^5, z, n)
ans =
kroneckerDelta(n - 2, 0) + 3*kroneckerDelta(n - 3, 0) + 6*kroneckerDelta(n - 4, 0) +
5*kroneckerDelta(n - 5, 0)
>> iztrans([exp(x), 1; sin(y), i*z],[w, x; y, z],[a, b; c, d])
ans =
[ exp(x)*kroneckerDelta(a, 0), kroneckerDelta(b, 0)]
[       iztrans(sin(y), y, c),  iztrans(z, z, d)*1i]
```

3.2 多项式及其运算

在数学上,多项式是一类基本的数学函数,因为它简单且可组成完备函数基,因此在很多研究中用它来作为复杂函数的近似形式。

多项式一般可表示为以下形式:

$$f(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$

对于这种表示形式,很容易用它的系数向量来表示,即

$$\boldsymbol{p} = [a_0, a_1, \cdots, a_{n-1}, a_n]$$

在 MATLAB 中,提供了 poly2sym 函数实现多项式的构造,其调用格式如下。

r = poly2sym(c):c 为多项式的系数向量。

r = poly2sym(c, v):c 为多项式的系数向量,v 为其变量。

【例 3-31】 利用 poly2sym 函数创建多项式。

其 MATLAB 代码编程如下:

```
>> poly2sym([1 3 2])                    %默认创建多项式变量为 x
ans =
x^2 + 3*x + 2
>> poly2sym([.694228, .333, 6.2832])    %默认创建多项式变量为 x
ans =
(6253049924220329*x^2)/9007199254740992 + (333*x)/1000 + 3927/625
>> poly2sym([1 0 1 -1 2],'y')           %指定多项式变量为 y
ans =
y^4 + y^2 - y + 2
```

3.2.1 多项式的求根

1. 多项式的根

找出多项式的根,即使多项式为 0 的值,可能是许多学科共同的问题。MATLAB 能求解这个问题,并提供了特定的函数 roots 求解一个多项式的根。函数的调用格式为

r = roots(c):其中 c 为多项式的系数向量,r 为求解多项式的根。

【例 3-32】 利用 roots 函数求多项式 $x^4-12x^3+25x+116$。

其 MATLAB 代码编程如下：

```
>> clear all;
p = [1 -12 0 25 116]
r = roots(p)
p =
     1   -12   0   25   116
r =
  11.7473 + 0.0000i
   2.7028 + 0.0000i
  -1.2251 + 1.4672i
  -1.2251 - 1.4672i
```

注意：必须包括具有零系数的项。除非特别地辨认，MATLAB 无法知道哪一项为零。

2. 由根创建多项式

因为在 MATLAB 中，无论是一个多项式，还是多项式的根都是向量，MATLAB 按惯例规定，多项式是行向量，根是列向量。给出一个多项式的根，也可以构造相应的多项式。在 MATLAB 中，poly 函数可以实现，函数的调用格式为

p = poly(A)：如果 A 为方阵，则多项式 p 为该方阵的特征多项式；如果 A 为向量，则 A 的元素为该多项式 p 的根。n 阶方阵的特征多项式存放在行向量中，并且特征多项式最高次数的系数一定为 1。

接上例，代码为

```
>> pp = poly(r)
pp =
    1.0000   -12.0000   -0.0000   25.0000   116.0000
```

3.2.2 多项式的四则运算

1. 多项式的加法

对多项式的加法，MATLAB 并未提供一个特别的函数。如果两个多项式向量大小相同，那么多项式相加时就与标准的数组加法相同。

【例 3-33】 计算给定多项式的和与差。

其 MATLAB 代码编程如下：

```
>> clear all;
p1 = [5 40 6 21 9 3];
p2 = [4 0 3 72 1 8];
p3 = p1 + p2                                  % 多项式的和
p3 =
     9 40 9 93 10 11
>> r1 = poly2str(p3,'x')                      % 显示多项式
```

```
r1 =
   9 x^5 + 40 x^4 + 9 x^3 + 93 x^2 + 10 x + 11
>> p4 = p1 - p2                          % 多项式的差
p4 =
     1   40   3   -51   8   -5
>> r2 = poly2str(p4,'x')                 % 显示多项式
r2 =
   x^5 + 40 x^4 + 3 x^3 - 51 x^2 + 8 x - 5
```

注意：当两个多项式阶次不同时，低阶的多项式用首零填补，使其与高阶多项式有同样的阶次。要求首零而不是尾零，是因为相关的系数像 x 幂次一样，必须整齐。

2. 多项式的乘法

在 MATLAB 中提供了 conv 函数用于实现多项式的乘运算，提供了 deconv 函数实现多项式的除运算。它们调用格式如下。

c = conv(a,b)：执行 a,b 两个向量的卷积运算。

c = conv(a,b,'shape')：按形参'shape'返回卷积运算，shape 的取值如下。

- full：为返回完整的卷积，其为默认值。
- same：为返回部分卷积，其大小与向量 a 大小相等。
- valid：只返回无填充零部分的卷积，此时输出向量 c 的最大值为 max(length(a)－max(0,length(b)－1), 0)。

【例 3-34】 利用 conv 函数求多项式 $f(x)=x^4+4x^3-2x^2+7x+11$ 和 $g(x)=9x^4-11x^3+5x^2+8$ 的乘法运算。

其 MATLAB 代码编程如下：

```
>> clear all;
f = [1 4 -2 7 11];
g = [9 -11 5 0 8];
c = conv(f,g)
```

运行程序，输出如下：

```
c =
     9   25   -57   105   20   -54   39   56   88
```

提示：

conv 函数只能进行两个多项式的乘法，两个以上的多项式的乘法需要重复使用 conv。

在一些特殊情况下，一个多项式需要除以另一个多项式。在 MATLAB 中，这是由函数 deconv 完成的。函数的调用格式为

[q,r] = deconv(v,u)：求多项式 v,u 的除法运算，其中 q 为返回多项式 v 除以 u 的商式，r 为返回 v 除以 u 的余式。返回的 q 与 r 仍为多项式系数向量。

【例 3-35】 多项式的除法运算。

其 MATLAB 代码编程如下：

```
>> clear all;
>> c = [1 5 15 35 69 100 118 110 72];
```

```
>> b = [1 2 3 6 8];
>> [a,r] = deconv(c,b)
```

运行程序,输出如下:

```
a =
     1   3   6   8   9
r =
     0   0   0   0   0   -2   -5   -8   0
```

a 是多项式 c 除以多项式 b 的商,余式为 r。本例中 r 为零多项式,因为多项式 b 和多项式 a 的乘积恰好是 c。

3.2.3 多项式的导数

在 MATLAB 中提供了 polyder 函数用于多项式的求导。其调用格式如下。

k = polyder(p):求多项式 p 的导函数多项式。

k = polyder(a,b):求多项式 a 与多项式 b 乘积的导函数多项式。

[q,d] = polyder(b,a):求多项式 b 与多项式 a 相除的导函数,导函数的分子存入 q,分母存入 d 中。

其中,参数 p,a 与 b 是多项式的系数向量,返回结果 q,d,k 也是多项式的系数向量。

【例 3-36】 多项式 $p(x)=(3x^2+6x+9)(x^2+2x)$的求导运算。

其 MATLAB 代码编程如下:

```
>> clear all;
a = [3 6 9];
b = [1 2 0];
k = polyder(a,b)
k =
    12   36   42   18
>> K = poly2str(k,'x')
K =
   12 x^3 + 36 x^2 + 42 x + 18
>> [q,d] = polyder(b,a)
q =
    18   18
d =
     9   36   90   108   81
```

3.2.4 多项式的积分

在 MATLAB 中,提供了 polyint 函数用于对多项式进行积分。函数的调用格式为

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

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

【例 3-37】 已知多项式 x^2-x+2，求 $F(x)=\int p(x)\mathrm{d}x$，且 $F(0)=1/2$。

其 MATLAB 代码编程如下：

```
>> p = [1 -1 2];                          % 原多项式
>> k = 1/2;                               % 常数项 k
>> F = polyint(p,k)                       % 求多项式的积分
F =
    0.3333   -0.5000   2.0000   0.5000
>> df = poly2sym(F)
df =
x^3/3 - x^2/2 + 2*x + 1/2
```

所以 $F(x)=\frac{1}{3}x^3-\frac{1}{2}x^2+2x+\frac{1}{2}$。

3.2.5 多项式的估值

在 MATLAB 中提供了 polyval 函数与 polyvalm 函数用于求多项式 $p(x)$在 $x=a$ 的取值。polyval 的输入可以是标量或矩阵。其调用格式为

y = polyval(p,x)：p 为多项式的系数向量，x 为矩阵，它是按数组运算规则来求多项式的值。

[y,delta] = polyval(p,x,S)：使用可选的结构数组 S 产生由 polyfit 函数输出的估计参数值；delta 是预测未来的观测估算的误差标准偏差。

y = polyval(p,x,[],mu)或[y,delta] = polyval(p,x,S,mu)：使 $\hat{x}=(x-\mu_1)/\mu_2$ 替代 x，在等式中，$\mu_1=\mathrm{mean}(x)$，$\mu_2=\mathrm{std}(x)$，其中心点与坐标值 $mu=[\mu_1,\mu_2]$可由 polyfit 函数计算得出。

polyvalm 函数的输入参数只能是 N 阶方阵，这时可以将多项式看作矩阵函数。其调用格式为

Y = polyvalm(p,X)：p 为多项式的系数向量，X 为方阵，其是按矩阵运算规则来求多项式的值。

【例 3-38】 多项式的求值。

其 MATLAB 代码编程如下：

```
>> clear all;
X = pascal(4)                             % 建立四阶 pascal 矩阵
X =
     1   1   1   1
     1   2   3   4
     1   3   6  10
     1   4  10  20
>> p = poly(X)
p =
    1.0000  -29.0000  72.0000  -29.0000  1.0000
>> P = poly2str(p,'x')
P =
```

```
    x^4 - 29 x^3 + 72 x^2 - 29 x + 1
>> y = polyval(p,X)                          % 利用 polyval 求解多项式的值
y =
  1.0e+004 *
    0.0016    0.0016    0.0016    0.0016
    0.0016    0.0015   -0.0140   -0.0563
    0.0016   -0.0140   -0.2549   -1.2089
    0.0016   -0.0563   -1.2089   -4.3779
>> Y = polyvalm(p,X)                         % 利用 polyvalm 求解多项式的值
Y =
  1.0e-010 *
   -0.0013   -0.0063   -0.0104   -0.0242
   -0.0048   -0.0218   -0.0360   -0.0798
   -0.0115   -0.0512   -0.0822   -0.1812
   -0.0229   -0.0973   -0.1560   -0.3410
```

3.2.6 有理多项式

在许多应用中，例如傅里叶(Fourier)、拉普拉斯(Laplace)和 Z 变换中，出现了两个多项式之比。在 MATLAB 中，有理多项式由它们的分子多项式和分母多项式表示。对有理多项式进行运算的两个函数是 residue 和 polyder。函数 residue 执行部分分式展开的运算。函数的调用格式为

[r,p,k] = residue(b,a)：b，a 分别为分子和分母多项式系数的行向量，r 为留数行向量。其中，

$$\frac{b(s)}{a(s)}=\frac{b_1 s^m+b_2 s^{m-1}+\cdots+b_{m+1}}{a_1 s^n+a_2 s^{n-1}+\cdots+a_{n+1}}$$

[b,a] = residue(r,p,k)：p 为极点行向量，k 为直项行向量。其中，

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

【例 3-39】 求表达式 $f(x)\dfrac{b(s)}{a(s)}=\dfrac{5s^3+3s^2-2s+7}{-4s^3+8s+3}$ 的部分分式展开式子。

其 MATLAB 代码编程如下：

```
>> b = [ 5 3 -2 7];                         % 分子系数向量
a = [-4 0 8 3];                             % 分母系数向量
[r, p, k] = residue(b,a)
r =
   -1.4167
   -0.6653
    1.3320
p =
    1.5737
   -1.1644
   -0.4093
k =
   -1.2500
```

```
>> [b,a] = residue(r,p,k)
b =
   -1.2500   -0.7500    0.5000   -1.7500
a =
    1.0000   -0.0000   -2.0000   -0.7500
```

所以，其部分分式展开表达式为 $f(x)=\dfrac{b(s)}{a(s)}=\dfrac{-1.25s^3-0.75s^2+0.5s-1.75}{s^3-2s-0.75}$。

3.2.7 多项式的微分

多项式微分的概念是显然的，对 n 阶多项式 $p(x)=a_nx^n+a_{n-1}x^{n-1}$，其微分为 $n-1$ 阶多项式，即为 $\mathrm{d}p(x)=na_nx^{n-1}+(n-1)a_{n-1}x^{n-2}+\cdots+a_1$。原多项式及其微分多项式的多项式表示分别为 $P=[a_n,a_{n-1},\cdots,a_0]$，$\mathrm{d}p=[na_n,(n-1)a_{n-1},\cdots,a_1]$。

在 MATLAB 中提供了 polyder 函数用于求多项式的微分，其调用格式为

k = polyder(p)：p,k 分别为原多项式及微分多项式的多项式表示。

k = polyder(a,b)：求多项式 a 与多项式 b 乘积的导函数多项式。

[q,d] = polyder(b,a)：求多项式 b 与多项式 a 相除的导函数，导函数的分子存入 q，分母存入 d。

其中，参数 a 和 b 是多项式的系数向量，返回结果 q 和 d 与是多项式的系数向量。

【例 3-40】 多项式 $p(x)=(3x^2+6x+9)(x^2+2x)$的微分运算。

```
>> clear all;
a = [3 6 9];
b = [1 2 0];
k = polyder(a,b)
k =
    12   36   42   18
>> K = poly2str(k,'x')
K =
   12 x^3 + 36 x^2 + 42 x + 18
>> [q,d] = polyder(b,a)
q =
    18   18
d =
     9   36   90   108   81
```