第5章 MATLAB符号运算

本章要点：

- ◇ MATLAB符号运算的特点；
- ◇ MATLAB符号对象的创建和使用；
- ◇ MATLAB符号多项式函数运算；
- ◇ MATLAB符号微积分运算；
- ◇ MATLAB符号方程求解。

5.1 MATLAB符号运算的特点

MATLAB语言的符号运算是指基于数学公理和数学定理，采用推理和逻辑演绎的方式对符号表达式进行运算从而获得符号形式的解析结果。在MATLAB中，符号常量、符号函数、符号方程和符号表达式等的计算成为符号运算的内容，一方面这些符号运算都严格遵循数学中的各种计算法则、基本的计算公式来进行，另一方面符号运算时可以根据计算精度的实际要求来调整运算符号（数值）的有效长度，从而符号运算所得的结果具有完全准确的特点，这种完全准确的结果可以解决纯数值计算中误差不断累积的问题。

在实际的科研计算和工程计算过程中，取得符号形式的解析结论是有重大意义的。它能完整清晰地给出对结论有影响的变量是哪一些，详尽描述各变量相互的定性及定量的关系。相比较数值计算所得的结论，符号解析式的结论易于分析和总结，其逻辑清晰，结构完整，作为阶段性的结论又能轻易地移植到比它高一级的知识结构中，同时，符号形式的解析结论也是知识成果的传统保存形式之一。但在许多的计算工作中，取得符号形式的解析结论还是有一定的困难的，而不得不采用数值计算的方式来对自然现象和科学实验进行研究和诠释。

在进行符号计算时，MATLAB会调用内置的符号计算工具箱进行运算，然后返回到指令窗口中。例如，MATLAB 7.8的符号计算工具箱采用MuPAD（Multi Processing Algebra Data Tool）软件，其不但具有符号计算的功能，还具有完善的绘图功能。可以输入多个2D函

数、极坐标函数或3D函数，选择绘图参数，就可以轻松地完成图形的绘制，此外图形的动画制作也非常方便。MATLAB的符号计算工具箱功能远不止于推导公式，结合系统的建模和仿真Simulink功能模块，在算法开发、数据可视化、数据分析以及数值计算等方面具有广泛的应用空间。

5.2 MATLAB符号对象的创建和使用

MATLAB符号运算工具箱处理的符号对象主要是符号常量、符号变量、符号表达式以及符号矩阵。符号常量是不含变量的符号表达式。符号变量即由字母(除了i与j)与数字构成的，且由字母打头的字符串。任何包含符号对象的表达式或方程，即为符号表达式。任何包含符号对象的矩阵，即为符号矩阵。换言之，任何包含符号对象的表达式、方程或矩阵也一定是符号对象。要实现符号运算，首先需要将处理对象定义为符号变量或符号表达式。

1. 符号对象的创建

符号对象的创建可以使用sym和syms函数来实现。在声明一个符号变量时，二者没有区别，可以互用，但syms函数可以同时声明多个变量且还可以直接声明符号函数。

1) sym函数的调用格式

```
P = sym(p ,flag) % 由数值 p 创建一个符号常量 P
```

当p是数值时，sym函数可以将其转变为一个符号常量P；为了说明所创建的符号常量P所需的计算精度，参数flag可以是'f'、'e'、'r'或'd'4种格式；'f'表示符号常量P是浮点数，'e'表示给出估计误差，'r'表示有理数，'d'表示采用基数为十的浮点数，其有效长度由VPA函数指定；默认为'r'格式。

在MATLAB窗口直接输入下列的命令行，示范sym函数的使用：

```
>> a1 = sym(sin(pi/3),'d')
 % 采用基数为十的浮点数表示，有效长度由 VPA 函数指定，这里是 32 位有效长度
a1  = 0.86602540378443859658830206171842
```

继续输入：

```
>> a2 = sym(sin(pi/3),'f')    % 采用有理分式的浮点数表示
 a2  = 3900231685776981/4503599627370496
```

继续输入：

```
>> a3 = sym(sin(pi/3),'e') % 采用有理数表示，并给出误差估计的有理分式
 a3  = 3^(1/2)/2 - (47 * eps)/208
```

继续输入：

```
>> a4 = sym(sin(pi/3),'r')      % 采用有理数表示,不给出误差
 a4 = 3^(1/2)/2
```

继续输入：

```
>> a5 = sym(sin(pi/3))      % 默认采用有理数表示
 a5 = 3^(1/2)/2
```

【例 5-1】 举例说明数值常量与符号常量的区别。

在 MATLAB 窗口直接输入下列的命令行：

```
>> a = sin(pi/3);                % 创建了一个数值常量 a
>> a1 = sym(sin(pi/3),'d');      % 创建了一个符号常量 a1
>> vpa(a - a1)                   % 采用两者之差来说明区别
 ans = 0.00000000000000000501754211090345161834713688395 63
% ans 显示了 a 与 a1 的差值,并以 32 位有效长度的数字加以表示
```

2）声明符号变量(集)、创建符号表达式以及符号矩阵

先使用 sym 和 syms 函数来声明符号变量或符号变量集，再使用已定义符号变量去构造需表达的符号表达式以及符号矩阵。

通过输入以下的命令行进行举例：

```
>> syms a b c x              % 声明一个符号变量集
>> f1 = a*x^2 + b*x + c      % 创建符号表达式
 f1 = a*x^2 + b*x + c
```

继续输入：

```
>> syms a b c x
>> A = [a*x^2 b*c;a*sqrt(x) b+c]      % 创建符号矩阵
A =
[    a*x^2, b*c]
[ a*x^(1/2), b+c]
```

2. 自由符号变量

在符号表达式 f1＝ax^2＋bx＋c 中，式子等号右边共有 4 个符号，其中，x 被习惯认为是自变量，其他被认为是已知的符号常量。在 MATLAB 中，x 称为自由符号变量，其他已知的符号常量被认为是符号参数，解题时是围绕自由符号变量进行的。得到的结果通常是“用符号参数构成的表达式表述自由符号变量”，解题时，自由符号变量可以人为指定，也可以由软件默认地自动认定。

1）自由符号变量的确定

符号表达式中的多个符号变量，系统按以下原则来选择自由符号变量：首选自由符

号变量是 x，倘若表达式中不存在 x，则与 x 的 ASCII 值之差的绝对值小的字母优先；差绝对值相同时，ASCII 码值大的字母优先。例如：

识别自由符号变量时，字母的优先顺序为 x、y、w、z、v 等。

规定大写字母比所有的小写字母都靠后，例如：

在符号表达式 ax^2+bX^2 中，自由符号变量的顺序为 x、b、a、X。

此外，字母 pi、i 和 j 等不能作为自由符号变量。

2）列出符号表达式中的符号变量

```
symvar(S)            % 列出表达式 S 中的所有的符号变量
symvar(S ,n)         % 按优先次序列出表达式中 n 个自由符号变量
```

通过输入以下的命令行进行举例：

```
>> syms a b c x
>> f1 = a * x ^ 2 + b * x + c
>> symvar(f1)        % 列出表达式 f1 中的所有的符号变量
ans =
[ a, b, c, x]
```

继续输入：

```
>> symvar(f1,3)        % 按优先次序列出表达式中的 3 个自由符号变量
ans =
 [ x, c, b]
```

上述 symvar 函数还可列出整个符号矩阵中的自由符号变量，如下：

```
>> A = [a * x ^ 2 b * c;a * sqrt(x) b + c];
>> symvar(A)        % 列出矩阵 A 中的所有的符号变量
ans =
 [ a, b, c, x]
```

继续输入：

```
>> symvar(A,3)      % 按优先次序列出矩阵 A 中的 3 个自由符号变量
 ans =
  [ x, c, b]
```

3. 基本的符号运算

MATLAB 中基本的符号运算范围及所采用的运算符号与数值运算没有大的差异，涉及的运算函数也几乎与数值计算中的情况完全一样，简介如下。

1）算术运算

加、减、乘、左除、右除、乘方：+、-、*、\、/、^。

点乘、点左除、点右除、点乘方：.*、.\、./、.^。

共轭转置、转置：'。

2）关系运算

相等运算符号：==。

不等运算符号：~=。

符号关系运算仅有以上两种。

3）三角函数、双曲函数

三角函数：sin、cos 和 tan 等。

双曲函数：sinh、cosh 和 tanh 等。

4）三角函数、双曲函数的反函数

反三角函数：asin、acos 和 atan 等。

反双曲函数：asinh、acosh 和 atanh 等。

5）复数函数

求复数的共轭：conj。

求复数的实部：real。

求复数的虚部：imag。

求复数的模：abs。

求复数的相角：angle。

6）矩阵函数

求矩阵的对角元素：diag。

求矩阵的上三角矩阵：triu。

求矩阵的下三角矩阵：tril。

求矩阵的逆：inv。

求矩阵的行列式：det。

求矩阵的秩：rank。

求矩阵的特征多项式：poly。

求矩阵的指数函数：expm。

求矩阵的特征值和特征向量：eig。

求矩阵的奇异值分解：svd。

通过输入以下命令行进行举例：

```
  syms a b c
>> B = [a b;c 0]
 >> C = inv(B)        % 求矩阵 B 的逆
C =
[ 0,        1/c]
[ 1/b, -a/(b*c)]
```

【例 5-2】 举例说明数值量与符号对象的混合运算。

通过输入以下命令行继续举例：

```
>> D = [1 2;3 4]        % 定义一个数值矩阵 D
D =
```

```
     1    2
     3    4
```

继续输入：

```
>> C + D       % 数值矩阵 D 与符号矩阵 C 直接相加
ans =
[       1,       1/c + 2]
[ 1/b + 3, 4 - a/(b * c)]
```

继续输入：

```
>> C * D
ans =          % 数值矩阵 C 与符号矩阵 D 直接相乘
[               3/c,             4/c]
[ 1/b - (3 * a)/(b * c), 2/b - (4 * a)/(b * c)]
```

4. 符号对象的识别与精度转换

在 MATLAB 中，函数指令繁杂多样，数据对象种类亦有多种。有的函数指令适用于多种数据对象，但也有一些函数指令仅适用某一种数据对象。在数值计算与符号计算混合使用的情况下，常常遇到由于指令和数据对象不匹配而出错的情况，因而在 MATLAB 中提供了数据对象识别与转换的函数指令。

1）识别数据对象属性

```
class(var)           % 给出变量 var 的数据类型
isa(var, 'Obj')      % 若变量是 Obj 代表的类别，给出 1 表示真
whos                 % 给出所有 MATLAB 内存变量的属性
```

通过输入以下命令行进行举例说明：

```
>> class(D)        % 识别矩阵 D 的数据类型
ans = double
```

继续输入：

```
>> class(C)        % 识别矩阵 C 的数据类型
ans = sym
```

继续输入：

```
>> class(C + D)        % 识别矩阵 C + D 之后的数据类型
ans = sym
```

继续输入：

```
>> isa(C + D,'sym')          % 询问 C + D 之后的数据类型是否为符号型
ans = 1                      % 结果为 1,表明 C + D 之后的数据类型是符号型
```

【例 5-3】 举例说明如何观察内存变量类型及其他属性。

通过输入以下命令行进行举例说明：

```
>> clear          % 清除工作区中的内存变量
>> a = 1;b = 2;c = 3;d = 4;
>> Mn = [a,b;c,d];
>> Mc = '[a,b;c,d]';
>> Ms = sym(Mn);
>> whos Mn Mc Ms  % 显示出变量 Mn Mc Ms 的大小、空间及类型属性
  Name     Size     Bytes  Class     Attributes
  Mc       1x9      18     char
  Mn       2x2      32     double
  Ms       2x2      112    sym
```

2）符号对象数值计算的精度转换

符号对象的数值计算需要考虑运行速度和内存空间的占用。符号对象数值计算可以采用系统默认的数值精度，亦可以根据计算需求设置任意的有效精度。要了解当前系统默认的数值精度和设置目前所需求的有效精度，可采用如下的函数指令：

```
>> digits          % 显示系统数值计算精度,以十进制浮点的有效数字位数表示
Digits = 32
```

或输入：

```
>> digits(16)      % 设定系统数值计算精度,有效数字位数被设定为 16 位
>> digits
Digits = 16
```

对于某个符号对象，可以根据计算需求取得 digits 所指定的系统数值计算精度，也可以个别设定某个具体的符号对象的数值计算精度。可参见如下的函数指令：

```
>> fs = vpa(sin(2) + sqrt(2))   % 将式 sin(2) + sqrt(2)转换为符号常量 fs,精度为 16 位
fs = 2.323510989198777
```

或输入：

```
>> gs = vpa(sin(2) + sqrt(2),8)     % 将式 sin(2) + sqrt(2)转换为符号常量 gs,精度设为 8 位
gs = 2.323511
```

前面介绍了由数值表达式转换为符号常量的函数指令，但有些时候需要将符号对象转换为双精度数值对象，MATLAB 采用如下的函数指令来转换：

```
num = double(s)      % 将符号对象 s 转换为双精度数值对象 num
>> n = double(gs)    % 将上例中的符号对象 gs 转换为双精度数值对象 n
n = 2.3000
```

由这个例子可以看出，双精度数值对象 n 运算速度最快，占用内存最少，但转换后得到的结果并不精确。双精度数值对象往往不能满足科学研究和工程计算的需要，此时可以使用 sym 函数指令将其转换为有理数类型。

【例 5-4】 举例说明一个数值表达式在双精度数值、有理数和任意精度数值等不同数值类型下的具体数字。

通过输入以下命令行进行举例说明：

```
>> clear
>> reset(symengine)                          % 重置 MATLAB 内部的 MuPAD 符号运算引擎
>> sa = sym(sin(3) + sqrt(3),'d')            % 将数值式子 sin(3) + sqrt(3)转换为符号常量 sa
sa = 1.873170815628744323433335229234071448445320129407 % 符号常量 sa 精度为十进制 32 位
```

继续输入：

```
>> a = sin(3) + sqrt(3) % 将数值式子 sin(3) + sqrt(3)赋值给双精度变量 a
a = 1.8732
>> format long
>> a = sin(3) + sqrt(3) % 将 sin(3) + sqrt(3)赋值给长格式双精度变量 a
a = 1.873170815628744
```

继续输入：

```
>> digits(48)
>> a = sin(3) + sqrt(3)
a = 1.873170815628744 % 采用 48 位系统精度后，长格式变量 a 值不变
```

继续输入：

```
>> sa48 = vpa(sin(3) + sqrt(3)) % 将式 sin(3) + sqrt(3)转换为 sa48，精度设为 48 位
 sa48 =
 1.873170815628744323433352292340714484453201293 95
```

5.3 符号多项式函数运算

5.3.1 多项式函数的符号表达形式及相互转换

多项式依运算的要求不同，有时需要整理并给出合并同类项后的表达形式，而有时又需要分解因式或将多项式展开，不一而足，均为常见的多项式运算操作。针对多项式运算操作，MATLAB 提供了多种形式的多项式表达方法。

1. 多项式展开和整理

1）多项式的展开

采用以下的函数指令可以将多项式展开成乘积项和的形式。

```
g = expand(f)      %将多项式展开成乘积项和的形式
```

通过输入以下命令行进行举例说明：

```
syms x y a b c
>> f1 = (x - a) * (x - b) * (x - c);
>> f2 = sin(x + y);
>> f3 = a * sin(x + b) + c * sin(x + a);
>> g1 = expand(f1)      %将多项式 f1 展开
g1 =
x^3 - b*x^2 - c*x^2 - a*x^2 - a*b*c + a*b*x + a*c*x + b*c*x
```

继续输入：

```
>> g2 = expand(f2)     %将多项式 f2 展开
g2 =
cos(x) * sin(y) + cos(y) * sin(x)
```

继续输入：

```
>> g3 = expand(f3)     %将多项式 f3 展开
g3 =
a * cos(b) * sin(x) + a * sin(b) * cos(x) + c * cos(a) * sin(x) + c * sin(a) * cos(x)
```

2）多项式的整理

多项式的书写习惯是按照升幂或降幂的规则来完成的，否则需要加以整理。上例中g1式子并不符合人们的书写习惯，可以使用下列的函数指令加以整理：

```
h = collect(g)          %按照默认的变量整理表达式 g,g 可以是符号矩阵
h = collect(g,v)        %按照指定的变量或表达式 v 整理表达式 g
```

通过输入以下命令行进行举例说明：

```
>> h1 = collect(g1) %按照变量 x 整理表达式 g1
h1 =
x^3 + (- a - b - c)*x^2 + (a*b + a*c + b*c)*x - a*b*c
```

继续输入：

```
>> h2 = collect(g2,cos(x))      %按照指定的表达式 cos(x)整理表达式 g2
h2 =
sin(y) * cos(x) + cos(y) * sin(x)
```

继续输入：

```
>> h3 = collect(g2,cos(y))      %按照指定的表达式 cos(y)整理表达式 g2
h3 =
sin(x) * cos(y) + cos(x) * sin(y)
```

继续输入：

```
>> h4 = collect(g3,cos(x)) % 按照指定的表达式 cos(x)整理表达式 g3
h4 =
 (a * sin(b) + c * sin(a)) * cos(x) + a * cos(b) * sin(x) + c * cos(a) * sin(x)
```

表达式 h1、h2、h3 和 h4 还可以进一步整理为排版形式的表达方式，如此加以美化之后，更加符合人们的书写习惯，所以有助于人们对表达式的解读，但美化之后的式子已非 MATLAB 所认可的符号表达式。

继续输入以下命令行进行举例说明：

```
>> pretty(h1) % 对符号表达式加以美化,注意式中指数的位置
 3          2
x + (- a - b - c) x + (a b + a c + b c) x - a b c
```

继续输入：

```
>> pretty(h2)
sin(x) cos(y) + cos(x) sin(y)
```

继续输入：

```
>> pretty(h4)
(a sin(b) + c sin(a)) cos(x) + a cos(b) sin(x) + c cos(a) sin(x)
```

2. 多项式因式分解与转换成嵌套形式

1）多项式的因式分解

把一个多项式在一个范围内化为几个整式的积的形式，这种变形称为因式分解，也称为分解因式。MATLAB 所提供的因式分解函数指令格式为

```
p = factor(f) % 将符号对象 f 进行因式分解
```

输入以下命令行进行举例说明：

```
>> syms x y a b c              % 声明一个符号变量集
>> f1 = x^2 - 3 * x + 2;       % 定义符号表达式 f1
>> h1 = factor(f1)             % 将 f1 进行因式分解
h1 =
[ x - 1, x - 2]
```

继续输入：

```
>> f2 = x^2 - 7 * x + 7;
>> h2 = factor(f2)
h2 =
x^2 - 7 * x + 7 % f2 无法进行进一步因式分解
```

继续输入：

```
>> f3 = (x + y)^2 - 10 * (x + y) + 25;
>> h3 = factor(f3)
h3 =
 [ x + y - 5, x + y - 5] % f3 对(x + y)所做的因式分解
```

继续输入：

```
>> f4 = a * x^2 + b * x + c;
>> h4 = factor(f4)
h4 =
a * x^2 + b * x + c        % 不支持全符号的表达式 f4 进行因式分解
>> factor(120)             % 对 120 进行质因数分解
ans =
     2     2     2     3     5
```

对于类似于全符号表达式 f4 的因式分解，可以采用另一种思路加以求解。构造一个 f4＝0 方程式，再用 MATLAB 所提供的函数指令 solve 求出其根，再写出其乘积的形式。此外，函数指令 factor(f)还可以对一个数，例如 120，进行质因数分解。

2）多项式转换成嵌套形式

在编制多项式计算程序时，如若知道多项式的嵌套形式，那么便可以采用一种迭代的算法来完成多项式的计算。MATLAB 所提供的多项式转换成嵌套形式的函数指令格式为

```
g= horner(f) % 将多项式 f 转换成嵌套形式 g
```

输入以下命令行进行举例说明：

```
>> syms x y z a b c
>> f1 = 2 * x^6 - 5 * x^5 + 3 * x^4 + x^3 - 7 * x^2 + 7 * x - 20;
>> g1 = horner(f1) % 将一维多项式 f1 转换成嵌套形式 g1
g1 =
x * (x * (x * (x * (x * (2 * x - 5) + 3) + 1) - 7) + 7) - 20
```

继续输入：

```
>> f2 = 3 * x^5 + 4 * x^4 * y + 2 * x^3 * y^2 - x * y^4 - y^5 + 9;
>> g2 = horner(f2) % 将二维多项式 f2 转换成嵌套形式 g2
g2 =
x * (x^2 * (2 * y^2 + x * (3 * x + 4 * y)) - y^4) - y^5 + 9
```

继续输入：

```
>> f3 = (2 + 2j) * z^3 + (1 + j) * z^2 + (2 + j) * z + (2 + j);
>> g3 = horner(f3) % 将复数多项式 f3 转换成嵌套形式 g3
g3 =
z * (z * (z * (2 + 2i) + 1 + 1i) + 2 + 1i) + 2 + 1i
```

3. 多项式的因式代入替换与多项式的数值代入替换

在符号运算中，有符号因式（或称子表达式）会多次出现在不同的地方，为了使总表达式简洁易读，MATLAB提供了如下指令用于多项式的符号因式代入替换。此外，符号形式的多项式已经得到，需要代入具体的数值进行计算，MATLAB亦提供了相应的指令。

1）多项式的符号因式代入替换

MATLAB所提供的多项式因式代入替换的函数指令格式为

```
DW = subexpr(D,'W') % 从 D 中自动提取公因子 w,并重写 D 为 DW
```

输入以下命令行进行举例说明：

```
[V,D] = eig(A) % 为了说明因子代入替换,求 A 的特征值及向量
V =
[ (a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)/2)/c - d/c,
(a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)/2)/c - d/c]
[                                          1,
                                                          1]
D =
[ a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)/2,
                                                          0]
[                                  0,
a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)/2]
```

继续输入：

```
>> DW = subexpr(D,'W') % 从 D 中自动提取公因子 W,并重写 D 为 DW
W =
(a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)
DW =
[ a/2 - W/2 + d/2,          0]
[          0, W/2 + a/2 + d/2]
```

继续输入：

```
>> VW = subexpr(V,'W') % 从 V 中自动提取公因子 W,并重写 V 为 VW
W =
 (a^2 - 2*a*d + d^2 + 4*b*c)^(1/2)
VW =
 [ (a/2 - W/2 + d/2)/c - d/c, (W/2 + a/2 + d/2)/c - d/c]
[               1,                    1]
```

2）多项式的数值代入替换

MATLAB所提供的数值代入替换的函数指令格式如下，需要说明的是，subs函数指令不但可以将数值代入多项式中，也可以将符号代入多项式中。在工程计算中subs

函数指令还可用来化简结果。

```
SR = subs(S,new)          % 用 new 代入替换 S 中的自由变量后得到 SR
SR = subs(S,old,new)      % 用 new 代入替换 S 中的 old 后得到 SR
```

输入以下命令行进行举例说明：

```
>> syms a b x y
>> f1 = a + b * cos(x);
f1 =
a + b * cos(x)
>> fr1 = subs(f1,cos(x),log(y)) % 用 log(y)代入替换 f1 中的 cos(x)后得到 fr1
fr1 =
a + b * log(y)
```

继续输入：

```
fr2 = subs(f1,{a,b,x},{2,3,pi/5}) % 用{2,3,pi/5}代入替换 f1 中的{a,b,x}后得到 fr2
fr2 =
 (3 * 5 ^ (1/2))/4 + 11/4
```

继续输入：

```
>> fr3 = subs((3 * 5 ^ (1/2))/4 + 11/4) % 用 subs 将 fr2 计算化简为一个数值表达式 fr3
fr3 =
4984416289487807/1125899906842624
```

继续输入：

```
>> fr3 = 4984416289487807/1125899906842624
fr3 =
    4.4271 % 依工程计算要求,fr3 最终化简为一个双精度数
```

【例 5-5】 试计算正弦函数 $f=a\sin(\omega t+\varphi)$ 在 12 秒处的值，其中 a、ω 和 φ 为已知量。

输入以下命令行进行计算：

```
>> syms t a w fai
>> f = a * sin(w * t + fai);
f =
a * sin(fai + t * w)
>> f12 = subs(subs(f,{a,w,fai},{10,100 * pi,pi/4}),t,12)
f12 =
5 * 2 ^ (1/2)              % 用 subs 分两步将 f12 计算出来
>> 5 * 2 ^ (1/2)
ans = 7.0711               % 依工程计算要求,f12 最终需简化为一个数
```

5.3.2 符号多项式的向量表示形式及其计算

1. 以向量形式输入多项式

对于诸如 $f=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_0$ 形式的一元多项式(已整理为降幂的标准形式),MATLAB提供了系数行向量$[a_n\ a_{n-1}\cdots a_0]$的表达方式,其等同于输入了多项式 f,相比于符号的表达形式,多项式向量的输入更为简洁。

输入以下命令行进行举例说明:

```
>> syms x a b c
>> m = a * x^2 + b * x + c;          % 以符号方式输入一个一元多项式 m
>> n = [a b c]
n =
 [ a, b, c]                          % 输入系数向量用来代表一个一元多项式 n
```

继续输入:

```
>> roots(m)            % 求一元方程式 m = 0 的符号解(根)
ans =
Empty sym: 0-by-1      % 提示符号为空,不支持以符号方式输入 m
```

继续输入:

```
>> roots(n)                                  % 求一元方程式 n = 0 的符号解(根)
ans =
-(b + (b^2 - 4 * a * c)^(1/2))/(2 * a)
-(b - (b^2 - 4 * a * c)^(1/2))/(2 * a)      % 求解完成
```

继续输入:

```
>> p = [1 2 3 4] % 求一元方程式 p = 0 的数值解(根)
>> roots(p)
ans =
  -1.6506 + 0.0000i
  -0.1747 + 1.5469i
  -0.1747 - 1.5469i
```

2. 将系数向量形式写成字符串形式的多项式

接着上面继续输入:

```
>> f = poly2str(p,'x') % 依照系数向量 p 写出 x 为变元的多项式
f =
   x^3 + 2 x^2 + 3 x + 4
```

但要注意的是,式 f 此时不是 MATLAB 所认可的符号表达式,而是一个字符串。

5.3.3 反函数和复合函数求解

1. 反函数的求解

在工程中的许多应用场景下，人们需要知道一个已知函数的反函数。对于单值函数的反函数，其函数值及函数图形都易于理解和掌握，多值函数的反函数则需要先定义一个主值范围，然后再对主值外的函数值加以讨论。

1）单自变量函数求反函数的指令

格式如下：

```
g = finverse(f) % 对原函数 f 的默认变量求反函数 g
```

输入以下命令行进行举例说明：

```
>> syms x
>> f1 = 2 * x ^ 2 + 3 * x+1
>> g1 = finverse(f)
Warning: finverse(2 * x ^ 2 + 3 * x + 1) is not unique.
> In sym.finverse at 43                    % 运行过程中警示其反函数值不止一个
g1 =
 - 3/4 + 1/4 * (1 + 8 * x)^(1/2)           % 对反函数 g 的定义域需定义一个主值范围
```

输入以下命令行进行举例说明：

```
syms a b x
>> f = a/x ^ 2 + b * cos(x)
>> g = finverse(f)
g =
RootOf(cos(_Z) * _Z ^ 2 * b + a - x * _Z ^ 2) % 提示函数 g 的解析形式是挂号内函数的根
```

上例中，函数 g 并没有给出一个确定形式的符号解，仅作了一个提示，对于这种情况，要么寻求其他的办法求符号解，要么就考虑数值方法求解。

2）多自变量函数求反函数的指令

格式如下：

```
g = finverse(f,v) % 对原函数 f 的指定变量 v 求反函数 g
```

输入以下命令行进行举例说明：

```
>> syms t x a b
>> f1 = b * exp( - t + a * x);
>> g1 = finverse(f1,t) % 对原函数 f1 的指定变量 t 求反函数 g1
g1 =
a * x - log(t/b)
```

继续输入：

```
>> g2 = finverse(f1) % 对原函数 f1 的默认变量 x 求反函数 g2
g2 =
 (t + log(x/b))/a
```

2. 求复合函数

复合函数的概念在各种应用环境下都被广泛使用。MATLAB 软件中也提供了求复合函数的指令，因函数复合的法则及其变量代入位置的不同，存在各种格式，如下所列：

1) f(g(y))形式的复合函数

```
k1 = compose(f,g)          % 复合法则是 g(y)代入 f(x)中 x 所在的位置
k2 = compose(f,g,t)        % g(y)代入 f(x)中 x 所在的位置，变量 t 再代替 y
```

输入以下命令行进行举例说明：

```
>> syms x y z t u
>> f = x * exp( - t);
>> g = sin(y);
>> k1 = compose(f,g)
k1 =
sin(y) * exp( - t)
```

继续输入：

```
>> k2 = compose(f,g,t)
k2 =
sin(t) * exp( - t)
```

2) h(g(z))形式的复合函数

```
k3 = compose(h,g,x,z) % 生成 h(g(z))形式的复合函数，g(z)代入 x 位置
k4 = compose(h,g,t,z) % 生成 h(g(z))形式的复合函数，g(z)代入 t 位置
```

输入以下命令行进行举例说明：

```
>> h = x^ - t
>> p = exp( - y/u)
>> k3 = compose(h,g,x,z)
k3 =
sin(z)^( - t)
```

继续输入：

```
>> k4 = compose(h,g,t,z)
k4 =
x^( - sin(z))
```

3）h(p(z))形式的复合函数

```
k5 = compose(h,p,x,y,z) % p(z)代入 x 位置,z 代入 y 所在位置
k6 = compose(h,p,t,u,z) % p(z)代入 t 位置,z 代入 u 所在位置
```

输入以下命令行进行举例说明：

```
>> k5 = compose(h,p,x,y,z)
k5 =
exp( - z/u)^( - t)
```

继续输入：

```
>> k6 = compose(h,p,t,u,z)
k6 =
x^( - exp( - y/z))
```

5.4 符号微积分运算

5.4.1 函数的极限和级数运算

MATLAB 具备强大的符号函数微积分运算能力，提供了求函数极限的命令，使用起来十分方便。计算函数在某个点处的极限数值是探讨函数连续性的一种主要的方法，而函数连续是许多算法的基础。此外，还可以通过导数的极限定义式来求导数，当然，在MATLAB 中可直接用求导数的指令来求取函数的导数。

1. 求函数极限

1）求函数极限的指令格式

```
limit(f,x,a)            % 相当于数学符号 lim_{x→a} f(x)
limit(f,a)              % 求函数 f 极限,只是变元为系统默认
limit(f)                % 求函数 f 极限,变元为系统默认,a 取 0
limit(f,x,a,'right')    % 求函数 f 右极限(x 右趋于 a)
limit(f,x,a,'left')     % 求函数 f 左极限(x 左趋于 a)
```

输入以下命令行进行举例说明：

```
>> syms x a
>> limit(sin(x)/x) % 已知 f(x) = sinx/x,求 lim_{x→0} f(x)
ans =
1
```

继续输入：

```
>> limit((1 + x)^(1/x)) % 已知 f(x) = (1 + x)^(1/x), 求 limx→0 f(x)
ans =
exp(1)
```

继续输入：

```
>> limit(1/x,x,0,'left') % 已知 f(x) = 1/x, 求 limx→0+ f(x), 左趋于 0+
ans =
 - Inf
```

继续输入：

```
>> limit(1/x,x,0,'right') % 已知 f(x) = 1/x, 求 limx→0- f(x), 右趋于 0-
ans =
Inf
```

2）求复变函数的极限

复变函数 $f(z)$的自变量 z 点在复平面上可以采用任意方式趋近于 z_0 点，必须是所有的趋近方式下得到的极限计算值均相同，复变函数 $f(z)$的极限才存在。求复变函数 $f(z)$的极限通常有两种方法：一种是参量方法；另一种是分别求实部二元函数 $u(x,y)$和虚部二元函数 $v(x,y)$的极限。下面以参量方法列举一例。

【例 5-6】 求复变函数 $f(z)=z^2$，z 趋于点 $2+4i$ 时的极限值，已知 z 局限于复平面上一条直线 $y=x+2$ 上运动。

欲求 $\lim_{z\to z_0} f(z)$，依题意不妨设 $x=t$，则 $y=t+2$，代入复变函数中可得 $f(t)$，z 趋于点 $2+4i$ 时为 t 趋近于 2，输入以下命令行对极限值进行求解：

```
>> syms x y z t
>> x = t;
>> y = t + 2;
>> z = x + i * y
z =
t + sqrt( - 1) * (t + 2)
>> f = z ^ 2
f =
 (t + sqrt( - 1) * (t + 2))^2
```

继续输入：

```
>> limit(f,t,2)
ans =
 - 12 + 16 * sqrt( - 1)
>> subs( - 12 + 16 * sqrt( - 1))
ans =
 - 12.0000 + 16.0000i
```

因而得 $\lim_{z\to z_0} f(z)=-12+16\mathrm{i}$。

2. 基本的级数运算

1）级数求和

```
symsum(s,x,a,b) % 计算表达式 s 当 x 从 a 到 b 的级数和
symsum(s,x,[a b]) 或 symsum(s,x,[a;b]) % 功能同上
symsum(s,a,b) % 计算 s 以默认变量从 a 到 b 的级数和
symsum(s) % 计算 s 以默认变量 n 从 0 到 n-1 的级数和
```

输入以下命令行进行举例说明：

```
>> syms n k x
>> symsum(n)
 ans =
n^2/2 - n/2
```

继续输入：

```
>> symsum(n,0,k-1)
ans =
    (k*(k - 1))/2
```

继续输入：

```
>> an = 5^(-n/2)
>> symsum(an,0,k)
ans =
5^(1/2)/4 - (1/5)^(k + 1)*5^(k/2 + 1/2)*(5^(1/2)/4 + 5/4) + 5/4
```

2）一维函数的泰勒级数展开

```
taylor(f,x,a) % 将函数 f 在 x=a 处展开成 5 阶(默认)泰勒级数
taylor(f,x) % 将函数 f 在 x=0 处展开成 5 阶泰勒级数
taylor(f) % 将函数 f 在默认变量为 0 处展开成 5 阶泰勒级数
```

此外，以上指令格式中还可以添加参数，指定'ExpansionPoint'（扩展点）、'Order'（阶数）和'OrderMode'（阶的模式）等计算要求，其格式如下：

```
taylor(f,x,a, 'PARAM1',val1,'PARAM2',val2,…)
```

输入以下命令行进行举例说明：

```
syms x y z
>> f = exp(-x)
>> h1 = taylor(f)
h1 =
 - x^5/120 + x^4/24 - x^3/6 + x^2/2 - x + 1
```

继续输入：

```
>> h2 = taylor(f,'order',7)
h2 =
x^6/720 - x^5/120 + x^4/24 - x^3/6 + x^2/2 - x + 1
```

继续输入：

```
>> h3 = taylor(f,'ExpansionPoint',1,'order',3)
h3 =
exp(-1) - exp(-1)*(x - 1) + (exp(-1)*(x - 1)^2)/2
```

MATLAB还可以求二维函数的泰勒级数展开。

5.4.2 符号微分运算

1. 求函数导数的命令

1）单变量函数求导

```
diff(f,x,n) % 计算 f 对变量 x 的 n 阶导数
diff(f,x) % 计算 f 对变量 x 的一阶导数
diff(f,n) % 计算 f 对默认变量的 n 阶导数
diff(f) % 计算 f 对默认变量的一阶导数
```

输入以下命令行进行举例说明：

```
>> syms x a
>> f = a*x^5
>> g1 = diff(f)
g1 =
5*a*x^4
```

继续输入：

```
>> g2 = diff(f,2)
g2 =
20*a*x^3
```

2）多元函数求偏导

```
diff(f,x,y) % 计算 f 对变量 x 偏导数,再求对变量 y 偏导
diff(f,x,y,z) % 求 x 偏导数,再求 y 偏导,然后再求 z 偏导
```

输入以下命令行进行举例说明：

```
>> syms x y z a b c
>> f = sin(a*x^2 + b*y^2 + c*z^2)
```

```
>> h1 = diff(f,x)
h1 =
2 * a * x * cos(a * x^2 + b * y^2 + c * z^2)
```

继续输入：

```
>> h2 = diff(f,x,y)
h2 =
-4 * a * b * x * y * sin(a * x^2 + b * y^2 + c * z^2)
```

继续输入：

```
>> h3 = diff(f,x,y,z)
h3 =
-8 * a * b * c * x * y * z * cos(a * x^2 + b * y^2 + c * z^2)
```

2. 隐函数求导数

1）求隐函数的一阶导数

由多元复合函数的求导法则可以推导出隐函数 $F(x,y)$的一阶导数求解公式为

$$\frac{\mathrm{d}y}{\mathrm{d}x}=-\frac{F_x}{F_y} \tag{5-1}$$

由式(5-1)可知，由隐函数 $F(x,y)$所确定的 y 与 x 之间的函数法则，无须作显性化处理就可以求导，但需要隐函数 $F(x,y)$对 x、y 的偏导数成立。

输入以下命令行进行举例说明：

```
>> syms x y a b
>> F1 = x^2 + y^2 - 1
>> DF1_dx = - diff(F1,x)/diff(F1,y)
DF1_dx =
-x/y
```

继续输入：

```
>> F2 = (x/a)^2 + (y/b)^2 - 1
>> DF2_dx = - diff(F2,x)/diff(F2,y)
DF2_dx =
-(b^2 * x)/(a^2 * y)
```

2）Jacobian(雅可比)矩阵计算

雅可比矩阵是多元函数(通常为隐函数形式)的一阶偏导数以一定方式排列而成的矩阵。这里提出雅可比矩阵的概念是因其元素均为一阶偏导数，则恰当元素之比便可应用于隐函数求导数。其格式如下：

```
jacobian(F,v) % 格式中 F、v 均为行向量，所得元素(i,j) = ∂F_i/∂v_j
```

输入以下命令行进行举例说明：

```
>> jacobian(F1,[x,y])
ans =
    [ 2 * x, 2 * y] % 一步求出 F1x = 2x、F1y = 2y
```

继续输入：

```
>> jacobian([F1,F2],[x,y])
ans =
 [     2 * x,     2 * y]
[ (2 * x)/a^2, (2 * y)/b^2] % 一步求出 F1x、F1y、F2x、F2y
```

3. 离散数据差分计算

diff 函数式用于求连续函数的导数，也可以用于求离散函数的差分。无论是求微分(导数)还是求差分，计算原理类似。差分计算格式如下：

```
diff(X,n,d)          % 当 d = 1 时，对 X 算 n 阶行差分，d = 2 则算列差分
diff(X,n)            % 对 X 算 n 阶行差分
diff(X)              % 对 X 算一阶差分
```

输入以下命令行进行举例说明：

```
>> V = [1 2 3 5 8 13 21 34 55]
>> diff(V) % 前后相邻元素之差
ans =
     1     1     2     3     5     8    13    21
```

继续输入：

```
>> A = [1 2 3 5;8 13 21 34;55 89 144 233]
A =
     1     2     3     5
     8    13    21    34
    55    89   144   233
```

继续输入：

```
>> G1 = diff(A) % 前后两行元素之差
G1 =
     7    11    18    29
    47    76   123   199
```

继续输入：

```
>> G2 = diff(A,2) % 行元素的二阶差分
G2 =
    40    65   105   170
```

继续输入：

```
>> G3 = diff(A,1,2) % 前后两列元素之差
G3 =
     1     1     2
     5     8    13
    34    55    89
```

继续输入：

```
>> G4 = diff(A,2,2) % 列元素的二阶差分
G4 =
     0     1
     3     5
    21    34
```

5.4.3 符号积分运算

1. 求函数积分的命令

```
int(S,v,a,b)          % 求函数 S 对指定变量 v 在[a,b]区间上的定积分
int(S,a,b)            % 求函数 S 对默认变量在[a,b]区间上的定积分
int(S,v)              % 求函数 S 对指定变量 v 的不定积分
int(S)                % 求函数 S 对默认变量的不定积分
```

求符号函数积分的指令格式如上所示，非常简洁，但在积分的应用计算中还有许多的问题需要加以考虑，诸如双重积分、复变函数积分以及积分上限函数计算等。积分形式的函数是数学应用中最常见的函数形式之一，学生或工程师为了取得结果通常要进行大量的积分运算。使用 MATLAB 所提供的符号积分运算功能，将极大地降低积分运算的难度，使得数学这一工具能够得到更便利的应用。

1）不定积分和定积分运算举例

【例 5-7】 求函数 $f=1/\sqrt{x^2+1}$ 的原函数。

输入以下命令行进行解题：

```
>> syms x
>> f = (x^2 + 1)^( - 1/2)
>> g = int(f) % 求 f 的不定积分便可以求出 f 的原函数
g =
asinh(x)
```

因而，函数 f 的原函数是函数 g，$g = a\sinh(x)+C$。

【例 5-8】 求定积分 $\int_0^{\infty}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}dx$ 的值。

输入以下命令行进行解题：

```
>> syms x
>> f = (1/sqrt(2 * pi)) * exp( - x ^ 2/2)
>> int(f,0,inf)
ans =
   (7186705221432913 * 2 ^ (1/2) * pi ^ (1/2))/36028797018963968
```

以上得到的是一个计算式而非一个数，需要再做一次计算从而得到一个确切的数，继续输入：

```
>> (7186705221432913 * 2 ^ (1/2) * pi ^ (1/2))/36028797018963968
ans =
    0.5000
```

2）双重积分和三重积分运算举例

【例 5-9】 求由方程 $x^2+y^2=1$ 确定的圆的面积。

输入以下命令行进行解题：

```
>> syms x y
>> S = int(int(1,y, - sqrt(1 - x ^ 2),sqrt(1 - x ^ 2)),x, - 1,1)
S =
pi
```

所求面积公式为 $S=\int_{-1}^{1}\left[\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}}\mathrm{d}y\right]\mathrm{d}x$，需要说明的是，算法是由人设计的，即本题中的计算公式先由人推导出，其后交由 MATLAB 进行计算。当然，本题中求面积的算法不是唯一的，例 5-10 中的三重积分的算法就更多了。

【例 5-10】 试计算椭球体 $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leqslant 1$ 的体积。

采用先求平行于 xoy 平面的椭球剖面的面积，然后再在 z 轴上将剖面的面积叠加求出体积。所求计算公式经推导如下：

$$V=\int_{-c}^{c}\left[\iint_{S(z)}\mathrm{d}x\mathrm{d}y\right]\mathrm{d}z=\pi ab\int_{-c}^{c}\left(1-\frac{z^2}{c^2}\right)\mathrm{d}z \tag{5-2}$$

方括号内即为椭球剖面的面积(平行于 xoy 平面)，其计算公式为

$$Sz=\int_{-\sqrt{a^2\left(1-\frac{z^2}{c^2}\right)}}^{\sqrt{a^2\left(1-\frac{z^2}{c^2}\right)}}\int_{-\sqrt{b^2\left(1-\frac{x^2}{a^2}-\frac{z^2}{c^2}\right)}}^{\sqrt{b^2\left(1-\frac{x^2}{a^2}-\frac{z^2}{c^2}\right)}}\mathrm{d}y\mathrm{d}x \tag{5-3}$$

输入以下命令行进行解题：

方法 1：

```
>> syms x y z a b c
>> g = sqrt(b ^ 2 * (1 - x ^ 2/a ^ 2 - z ^ 2/c ^ 2))
g =
(b ^ 2 * (1 - x ^ 2/a ^ 2 - z ^ 2/c ^ 2))^(1/2)
```

继续输入：

```
>> h = sqrt(a^2*(1-z^2/c^2))
h =
 (a^2*(1-z^2/c^2))^(1/2)
```

继续输入：

```
>> S = int(int(1,y,-g,g),x,-h,h)
S =
 -i*b^2*(-log(-i*(a^2*(c^2-z^2)/c^2)^(1/2)/(b^2*(a^2*(c^2-z^2)/c^2)^
(1/2)/a^2)^(1/2)*(b^2/a^2)^(1/2))+log(i*(a^2*(c^2-z^2)/c^2)^(1/2)/(b^2*(a
^2*(c^2-z^2)/c^2)^(1/2)/a^2)^(1/2)*(b^2/a^2)^(1/2)))*(c^2-z^2)/c^2/(b^2/a
^2)^(1/2)
```

继续输入：

```
>> V = int(S,z,-c,c)
V =
4/3*b^2*pi/(b^2/a^2)^(1/2)*c
```

方法 2：

```
syms a b c x y z
>> V = int(pi*a*b*(1-z^2/c^2),z,-c,c)
V =
 (4*pi*a*b*c)/3
```

虽然方法 1 的结果还需进一步整理变形，但方法 1 和方法 2 都能得到正确的结果。本题说明了解题计算方法的重要性，算法正确的情况下计算就十分简便。假如算法不优，则计算过程就复杂一些，时间也不可避免地拉长了。

2. 符号积分变换

对于电子类专业的学生和工程技术人员而言，傅里叶变换、拉普拉斯变换和 Z 变换是必须学习和掌握的专业基础知识。

1）傅里叶变换及逆变换

```
F = fourier(f)          % 对默认的变量进行傅里叶变换,F 的自变量默认为 w
F = fourier(f,v)        % 对默认的变量进行傅里叶变换,F 的自变量指定为 v
F = fourier(f,u,v)      % 对指定的变量 u 进行傅里叶变换,F 的自变量为 v
```

以上为傅里叶变换的命令格式，下面为傅里叶逆变换的命令格式：

```
f = ifourier(F)           % 对默认变量 w 进行傅里叶逆变换,f 的自变量为 x
f = ifourier(F,v)         % 对指定的变量 v 进行傅里叶逆变换,f 的自变量为 x
f = ifourier(F,v,u)       % 对指定的变量 v 进行傅里叶逆变换,f 的自变量为 u
```

输入以下命令行进行举例说明：

```
>> syms x t u v w a b
>> f1 = sin(a * x);
```

继续输入：

```
>> Fw1 = fourier(f1)
Fw1 =
pi * (dirac(a + w) - dirac(a - w)) * 1i
```

继续输入：

```
>> f3 = ifourier(Fw1)
f3 =
 (exp( - a * x * 1i) * 1i)/2 - (exp(a * x * 1i) * 1i)/2
>> f3 = simplify(f3)
ans =
sin(a * x) % f3 与 f1 相比,形式上是一样的
```

继续输入：

```
>> f2 = sin(b * t)
>> Fw2 = fourier(f2)
Fw2 =
pi * (dirac(b + w) - dirac(b - w)) * 1i
```

继续输入：

```
>> f4 = ifourier(Fw2)
f4 =
   (exp( - b * x * 1i) * 1i)/2 - (exp(b * x * 1i) * 1i)/2
>> f4 = simplify(f4)
f4 =
sin(b * x) % f4 与 f2 相比,形式上是一样的,但自变量不一样
```

比较函数 f1 和函数 f2，其自变量是不同的，分别为 x 和 t。经傅里叶变换命令 fourier(f)均可以计算得出频谱函数 Fw1 和 Fw2，并以 w 为自变量。频谱函数 Fw1 和 Fw2 经傅里叶逆变换命令 ifourier(F)后又得函数 f3 和 f4，理论上函数 f3 和 f4 应等于函数 f1 和 f2。但函数 f3 和 f4 的自变量均为 x，从而函数 f2 与函数 f4 的自变量就不一样了。

要解决函数 f2 与函数 f4 的自变量不一样的问题，只需采用 f = ifourier(F,w,t)格式的命令。见下列程序行：

```
>> f5 = ifourier(Fw2,w,t)
f5 =
 (exp( - b * t * 1i) * 1i)/2 - (exp(b * t * 1i) * 1i)/2
>> f5 = simplify(f5)
f5 =
sin(b * t)
```

比较函数 f2 和函数 f5,其自变量是相同的。这个问题的解决有益于理清二维函数傅里叶变换的计算思路。

2）拉普拉斯变换及其逆变换

```
L = laplace(F)              % 对默认变量进行拉普拉斯变换,L 的自变量默认为 s
L = laplace(F,z)            % 对默认变量进行拉普拉斯变换,L 的自变量指定为 z
L = laplace(F,w,u)          % 对指定变量 w 进行拉普拉斯变换,L 的自变量指定为 u
```

输入以下命令行进行举例说明：

```
>> syms a s t w x F(t)
>> f = a * sin(w * x)
>> L1 = laplace(f)              % 对变量 x 进行函数 f 的拉普拉斯变换
L1 =
 (a * w)/(s ^ 2 + w ^ 2)        % L1 的自变量取默认的 s
```

继续输入：

```
>> L2 = laplace(f,t)             % 对变量 x 进行函数 f 的拉普拉斯变换
L2 =
 (a * w)/(t ^ 2 + w ^ 2)         % L2 的自变量指定为 t
```

继续输入：

```
>> L3 = laplace(f,w,t)             % 对指定变量 w 进行函数 f 的拉普拉斯变换
L3 =
 (a * x)/(t ^ 2 + x ^ 2)           % L3 的自变量指定为 t
```

继续输入：

```
>> L4 = laplace(diff(F(t))) % 对函数 f(默认变量)的导数进行拉普拉斯变换
L4 =
s * laplace(F(t), t, s) - F(0)
```

以上为拉普拉斯变换的命令格式及应用,下面为拉普拉斯逆变换的命令格式及应用：

```
F = ilaplace(L)             % 对 L(默认变量 s)进行逆变换,F 自变量默认为 t
F = ilaplace(L,y)           % 对 L(默认变量 s)进行逆变换,F 自变量指定为 y
F = ilaplace(L,y,x)         % 对 L(指定变量 x)进行逆变换,F 自变量默认为 y
```

输入以下命令行进行举例说明：

```
>> f1 = ilaplace(L1)          % 对 L1(默认变量 s)进行逆变换,f1 自变量默认为 t
f1 =
a * sin(t * w)
```

继续输入：

```
>> f2 = ilaplace(L2)          % 对 L2(默认变量 w)进行逆变换,f2 自变量默认为 t
f2 =
a * cos(t ^ 2)
```

继续输入：

```
>> f3 = ilaplace(L2,x,t)              % 对 L2(指定变量 x)进行逆变换,f3 自变量默认为 t
f3 =
 (a * w * dirac(t))/(t ^ 2 + w ^ 2)     % 因 L2 中无变量 x,则取 x = 1 来进行计算
```

继续输入：

```
>> f4 = ilaplace(L2,t,x)          % 对 L2(指定变量 t)进行逆变换,f3 自变量默认为 x
f4 =
a * sin(w * x)
```

比较函数 f 与函数 f2、f3、f4，只有 f4＝f。说明只有正确地应用拉普拉斯逆变换的命令格式，才能得到想要的逆变换结果(这里是函数 f4)。

3) Z 变换与 Z 逆变换

```
F = ztrans(f)          % 对 f(默认变量 n)进行 Z 变换,F 的自变量默认为 z
F = ztrans(f,w)        % 对 f(默认变量 n)进行 Z 变换,F 的自变量指定为 w
F = ztrans(f,k,w)      % 对 f(指定变量 k)进行 Z 变换,F 的自变量指定为 w
```

输入以下命令行进行举例说明：

```
>> syms n k w z a b
>> f = a * sin(k * n) + b * cos(k * n)
f =
b * cos(k * n) + a * sin(k * n)
```

继续输入：

```
>> F1 = ztrans(f)           % 对变量 n(默认的)进行离散函数 f 的 Z 变换
F1 =                        % F1 的自变量取默认的 z
(a * z * sin(k))/(z ^ 2 - 2 * cos(k) * z + 1) + (b * z * (z - cos(k)))/(z ^ 2 - 2 * cos(k) * z +
1)
```

继续输入：

```
>> F2 = ztrans(f,w)        % 对变量 n(默认的)进行离散函数 f 的 Z 变换
F2 =                       % F2 的自变量取指定的 w
(a * w * sin(k))/(w ^ 2 - 2 * cos(k) * w + 1) + (b * w * (w - cos(k)))/(w ^ 2 - 2 * cos(k) * w +
1)
```

继续输入：

```
>> F3 = ztrans(f,k,w)          % 对变量 k(指定的)进行离散函数 f 的 Z 变换
F3 =                           % F3 的自变量取指定的 w
(a * w * sin(n))/(w^2 - 2 * cos(n) * w + 1) + (b * w * (w - cos(n)))/(w^2 - 2 * cos(n) * w + 1)
```

由 Z 变换的性质易知,离散函数 f(n)移位之后的 Z 变换形式将发生变化,输入以下命令行进行举例说明:

```
>> syms f(n)
>> ztrans(f(n))
ans =
ztrans(f(n), n, z)
```

继续输入:

```
>> ztrans(f(n + 1))
ans =
z * ztrans(f(n), n, z) - z * f(0)
```

继续输入:

```
>> ztrans(f(n - 1))
ans =
f( - 1) + ztrans(f(n), n, z)/z
```

以上为 Z 变换的命令格式及应用,下面为 Z 逆变换的命令格式及应用:

```
f = iztrans(F)          % 对 F(默认变量 z)进行逆变换,f 自变量默认为 n
f = iztrans(F,k)        % 对 F(默认变量 z)进行逆变换,f 自变量默认为 k
f = iztrans(F,w,k)      % 对 F(指定变量 w)进行逆变换,f 自变量默认为 k
```

输入以下命令行进行举例说明:

```
>> f2 = iztrans(F2)
f2 =
a * sin(k * n) + (cos(k * n) * (b * cos(k) + a * sin(k)))/cos(k) - (a * cos(k * n) * sin(k))/cos
(k)
```

继续输入:

```
>> f2 = simplify(f2) % 对 F2 进行 Z 逆变换后再对结果进行简化
f2 =
b * cos(k * n) + a * sin(k * n)
 % f2 = f,说明 f 进行 Z 变换再进行逆变换后又回到原来的函数形式 f
>> f3 = iztrans(F3)
f3 =
a * sin(n^2) + (cos(n^2) * (b * cos(n) + a * sin(n)))/cos(n) - (a * cos(n^2) * sin(n))/cos(n)
```

继续输入：

```
>> simplify(f3)
ans =
a * sin(n^2) + b * cos(n^2)
```

继续输入：

```
>> f4 = iztrans(F3,w,k)
f4 =
a * sin(k * n) + (cos(k * n) * (b * cos(n) + a * sin(n)))/cos(n) - (a * cos(k * n) * sin(n))/cos
(n)
```

继续输入：

```
>> simplify(f4)
ans =
b * cos(k * n) + a * sin(k * n)
```

f3 与 f4 的结果不同，说明在进行 Z 逆变换时需要仔细选择正确的命令格式。

5.5 符号方程求解

5.5.1 符号代数方程求解

数学上方程大致可分为线性方程和非线性方程，也可以分为代数方程、常系数微分方程和偏微分方程等。首先要指出的是，利用 MATLAB 对符号方程求解，有些符号解答（解析的答案）可能求不出来，则 MATLAB 可以转而去寻求方程的数值解。有的时候只给出了方程的部分解，需要求解的人去做进一步的分析和检查。MATLAB 解方程的函数指令的使用较为复杂烦琐，为了能让读者易于上手及掌握，本节采用逐条介绍函数指令的方式。

1. solve 函数介绍和应用

MATLAB 所提供的 solve 函数指令主要用来求解代数方程（多项式方程）的符号解析解。也能解一些简单方程的数值解，不过对于解其他方程的能力比较弱，所求出的解往往是不精确或不完整的。注意可能得到的只是部分的结果，并不是全部解。

1）单变量符号方程求解

可采用的函数指令格式如下：

```
S = solve(eqn1)      % 求解方程 eqn1 关于默认变量的符号解 S,所谓默认变量可由 symvar(eqn1)
                     % 找寻
S = solve(eqn1,var1) % 求解方程 eqn1 关于指定变量 var1 的符号解 S
```

输入以下命令行进行举例说明：

```
>> syms a b x y
>> eqn1 = a * sin(x) == b
eqn1 =
a * sin(x) == b
```

继续输入：

```
>> S = solve(eqn1)    % 求解方程 eqn1 关于指定变量 x 的符号解 S
S =                   % 注意只给出了两个解
      asin(b/a)
pi - asin(b/a)
```

很明显，答案中只给出了两个解，这是需要进一步分析的。此时可以在函数指令中加入参数'ReturnConditions'，参数默认值为 false，若取 true，则需额外提供两个参数。

使用格式及应用举例说明如下：

```
>> [S,params, conditions] = solve(eqn1,'ReturnConditions',true)
S =
      asin(b/a) + 2 * pi * k
   pi - asin(b/a) + 2 * pi * k
params =
k
conditions =
a ~= 0 & in(k, 'integer')
         a ~= 0 & in(k, 'integer')
```

很明显，答案中给出了全部的解，其中含一个参数 k(params=k)。又进一步给出了两个解分别成立的条件：a ~= 0 & in(k, 'integer')及 a ~= 0 & in(k, 'integer')，解读为 a 不为 0 且 k 取整数。

如果方程无解，那么 solve 函数指令的运行又会出来怎样的结果呢？请看下面举例：

```
>> solve(2 * x + 1,3 * x + 1,x)
ans =
Empty sym: 0 - by - 1 % 直接显示无解
```

2）多变量符号方程组求解

可采用的函数指令格式如下：

```
[Svar1,Svar2,…,SvarN] = solve(eqn1,eqn2,…,eqnM,var1,var2,…,varN)
```

为了避免求解方程时对符号解产生混乱，需要指明方程组中需要求解的变量 var1，var2，…，varN，其所列的次序就是 slove 返回解的顺序，M 不一定等于 N。

输入以下命令行进行举例说明：

```
>> syms a b x y
>> eqn2 = x - y == a
>> eqn3 = 2 * x + y == b
[Sx,Sy] = solve(eqn2,eqn3,x,y)
Sx =
a/3 + b/3
Sy =
b/3 - (2 * a)/3
```

上面的例子中 M＝N，下面假如 M＜N，试看一下其运行的结果：

```
>> [Sx,Sy] = solve(eqn2,x,y)
Sx =
a
Sy =
0
```

根据 eqn2 方程，solve 解方程后给出一组解。此时可以在函数指令中加入参数'ReturnConditions'，以取得通解和解的条件。

solve 函数指令中加入其他的参数：'IgnoreProperties'默认取值为 false，当为 true 时求解会忽略变量定义时的一些假设，如假设变量为正(syms x positive)。

输入以下命令行进行举例说明：

```
>> syms t x positive % 声明 x 为正数变量
>> [St,Sy] = solve(t^2 - 1,x^3 - 1,t,x)
% 指令中无'IgnoreProperties',仅能得到一组正数解
St =
1
Sy =
1
```

继续输入：

```
>> [St,Sy] = solve(t^2 - 1,x^3 - 1,t,x,'ignoreproperties',true)
St =  % 加上'IgnoreProperties'参数,列出了全部解
 -1
   1
  -1
   1
  -1
   1
Sy =
                  1
                  1
   - (3^(1/2) * 1i)/2 - 1/2
   - (3^(1/2) * 1i)/2 - 1/2
     (3^(1/2) * 1i)/2 - 1/2
     (3^(1/2) * 1i)/2 - 1/2
```

solve 函数指令除'ReturnConditions'和'IgnoreProperties'参数之外，还有'IgnoreAnalyticConstraints'参数、'MaxDegree'参数、'PrincipalValue'参数和'Real'参数等可以选择，其中，'Real'参数为 true 时只给出实数解，调整'MaxDegree'参数可以给出大于3 解的显性解，'IgnoreAnalyticConstraints'参数为 true 时可以忽略掉一些分析的限制，'PrincipalValue'参数为 true 时只给出主值。

2. fsolve 函数介绍和应用

函数指令 fsolve 可以用于求解非线性方程组(采用最小二乘法)。它的一般调用方式为

```
X = fsolve(fun,X0,option)
```

返回的解为 X，fun，是定义非线性方程组的函数文件名，X0 是求根过程的初值，option 为最优化工具箱的选项设定。

函数指令 fsolve 最优化工具箱提供了 20 多个选项，用户可以在 MATLAB 中使用 optimset 命令将它们显示出来。可以调用 optimset()函数来改变其中某个选项。例如，Display 选项决定函数调用时中间结果的显示方式，其中，'off'为不显示，'iter'表示每步都显示，'final'表示只显示最终结果。optimset('Display','off')将设定 Display 选项为'off'。

【例 5-11】 求解下列非线性方程组的解：

$$\begin{cases} 2x - 0.8\sin x - 0.4\cos y = 0 \\ 3y - 0.8\cos x + 0.3\sin y = 0 \end{cases}$$

先于工作目录下编辑一个函数 m 文件，命名为 nonl.m。

```
function [n] = nonl(m)
x = m(1);
y = m(2);
n(1) = 2 * x - 0.8 * sin(x) - 0.5 * cos(y);
n(2) = 3 * y - 0.8 * cos(x) + 0.3 * sin(y);
end
```

然后在命令行窗口中运行下列指令：

```
>> x = fsolve('nonl',[0.8,0.7],optimset('Display','off'))
x =
    0.3993    0.2235
```

这里将解 x 代入到原方程中，对解的精度进行检验：

```
>> e = nonl(x)
e =
    1.0e-07 *
    0.6505    0.7505
```

解具有较高精度，达到了 10^{-7}的误差级别。

5.5.2 符号常微分方程求解

微分方程描述了自变量、未知函数和未知函数的微分之间的相互关系，与线性方程及非线性方程相比，其应用非常广泛，对微分方程的研究不断地推动着科学知识和工程技术的发展，而计算机的出现和发展又为提升微分方程的理论研究能力及工程应用水平提供了强有力的工具。常微分方程是指在微分方程中，自变量的个数仅有一个。

1. 单个符号常微分方程求解

MATLAB 提供的 dsolve 函数指令使用格式如下：

```
S = dsolve(eqn,'cond','v')
```

上列函数指令表示对微分方程 eqn 在条件 cond 下对指定的自变量 v 进行求解。其中，自变量 v 省略不写，自变量默认为 t，或在符号声明中指出自变量；cond 是初始条件，也可省略，而所得解中将出现任意常数符 C，构成微分方程的通解；eqn 为微分方程的符号表达式，方程中 D 被定义为微分，D2、D3 被定义为二阶、三阶微分，y 的一阶导数 dy/dx 或 dy/dt 则可定义为 Dy。

下面举例加以说明。

【例 5-12】 求解下列常微分方程，已知初始条件：$y(0)=1, y'(\pi/a)=0$。

$$\frac{d^2y}{dt^2} = -a^2 y(t)$$

程序代码如下：

```
>> syms y(t) a                    % 定义函数 y 及自变量 t
>> Dy = diff(y)                   % 定义 Dy 为 t 的一阶导数
Dy(t) =
diff(y(t), t)
>> D2y = diff(y,2)                % 定义 D2y 为 t 的二阶导数
D2y(t) =
diff(y(t), t, t)
```

继续输入：

```
>> yt = dsolve(D2y == -a^2*y, y(0) == 1, Dy(pi/a) == 0)
yt =
exp(-a*t*1i)/2 + exp(a*t*1i)/2
```

注意，微分方程及初始条件的格式，均为符号表达式而非字符串形式。符号表达式中的等号应采用关系运算符“==”。

【例 5-13】 求解常微分方程，已知初始条件：$w(0)=0$。

$$\frac{d^3w}{dx^3} = -w(x)$$

程序代码如下：

```
>> syms w(x) a                    % 定义函数 w 及自变量 x
>> Dw = diff(w)                   % 定义 Dw 为 x 的一阶导数
Dw(x) =
diff(w(x), x)
```

继续输入：

```
>> D2w = diff(w,2)                % 定义 D2w 为 x 的二阶导数
D2w(x) =
diff(w(x), x, x)
```

继续输入：

```
>> wx = dsolve(diff(D2w) == -a*w,w(0) == 0)
wx =
C2*exp(-x*((-a)^(1/3)/2 + (3^(1/2)*(-a)^(1/3)*1i)/2)) + C1*exp(-x*((-a)^
(1/3)/2 - (3^(1/2)*(-a)^(1/3)*1i)/2)) - exp((-a)^(1/3)*x)*(C1 + C2)
```

解 wx 中出现了两个常数 C1、C2，这是因为对于三阶的常微分方程只提供了一个初始条件，要给出特解还欠缺两个初始条件。

2. 符号常微分方程组的求解

符号常微分方程组的求解仍然使用 dsolve 函数指令，其使用格式如下：

```
[Sv1,Sv2,…] = dsolve(eqn1, eqn2,…,'cond1', 'cond2',…,'v1', 'v2',…)
```

上列使用格式中，[Sv1,Sv2,…]为返回的解，eqn1，eqn2，…为常微分方程组，最大可包含 12 个常微分方程，均以符号表达式形式填入。'v1'，'v2'，…为指定的自变量，也可以在符号声明中指出自变量及其函数。'cond1'，'cond2'，…是初始条件，既可以是符号表达式形式也可以是字符串形式。

下面举例加以说明。

【例 5-14】 求解下列常微分方程组，已知初始条件：$f(0)=1, g(0)=2$。

$$\begin{cases} f'(t) = f(t) + g(t) \\ g'(t) = g(t) - f(t) \end{cases}$$

程序代码如下：

```
>> syms f(t) g(t)
>> Df = diff(f)
Df(t) =
diff(f(t), t)
```

继续输入：

```
>> Dg = diff(g)
Dg(t) =
diff(g(t), t)
```

继续输入：

```
>> [sf,sg] = dsolve(Df == f + g,Dg == g - f,f(0) == 1,g(0) == 2)
sf =
exp(t) * cos(t) + 2 * exp(t) * sin(t)
sg =
2 * exp(t) * cos(t) - exp(t) * sin(t) % 注意两个返回解的先后次序
```

上面举例是采用标量形式来求解，下面再用向量和矩阵形式来求解：

```
>> syms f(t) g(t)
>> v = [f;g];
>> A = [1 1; -1 1];
 >> [Sf,Sg] = dsolve(diff(v) == A * v, v(0) == [1;2])
Sf =
exp(t) * cos(t) + 2 * exp(t) * sin(t)
Sg =
2 * exp(t) * cos(t) - exp(t) * sin(t)
```

5.5.3 一维偏微分方程求解

使用 MATLAB 求解偏微分方程或者方程组，常见有三种方法。第一种方法是使用 MATLAB 中的 PDE Toolbox。PDE Toolbox 既可以使用图形界面，也可以使用命令行进行求解。PDE Toolbox 主要针对求解二维问题(时间 t 不被计算维度)，欲求解三维问题则要设法降维求解，欲求解一维问题则要设法升维求解。第二种方法就是使用 MATLAB 中的 m 语言进行编程计算，相比 Fortran 和 C 等语言，MATLAB 中编程计算有许多库函数可以使用，对于大型矩阵的运算也要方便得多，当然使用 m 语言编程计算也有其劣势。第三种就是使用 pdepe 函数，MATLAB 中 pdepe 函数主要用于求解一维抛物型和椭圆形偏微分方程(组)。

1. 一维偏微分方程的求解

pdepe 函数指令的使用格式如下：

```
S = pdepe(m,@pdefun,@icfun,@bcfun,xmesh,tspan)
```

使用格式中 pdefun 是指一维偏微分方程具有如下的标准形式，如若不同应加以改写：

$$c\left(x,t,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left[x^{m}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right]+s\left(x,t,\frac{\partial u}{\partial x}\right) \tag{5-4}$$

式中，x 一般表示位置，t 一般表示时间，格式中的给定值 m 由方程的类型确定；格式中 bcfun 是其边界条件的标准形式，若不同于标准形式应加以改写：

$$p(x,t,u)+q(x,t,u)*f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0 \tag{5-5}$$

考虑左右边界条件，应该写出下列的两条：

$$\begin{cases} p(x_L,t,u)+q(x_L,t,u)*f\left(x_L,t,u,\dfrac{\partial u}{\partial x}\right)=0 \\ p(x_R,t,u)+q(x_R,t,u)*f\left(x_R,t,u,\dfrac{\partial u}{\partial x}\right)=0 \end{cases} \tag{5-6}$$

假定给定左边界条件 $u(x_L,t)=0$，则代入上式中第一条公式，得 $p(x_L,t,u)=u(x_L,t)$，$q(x_L,t,u)=0$；给定右边界条件$\dfrac{\partial u}{\partial x}(x_R,t)=N$，则代入上式中第二条公式，得 $p(x_R,t,u)=-N$，$q(x_L,t,u)=1$。

格式中 icfun 是其初始条件的标准形式，若不同于标准形式应加以改写：

$$u(x,t_0)=u_0$$

输出 S 为一个三维数组，$S(x(i),t(j),k)$表示 u_k 解。依照函数指令的使用格式，解题时先应在函数编辑器中编辑好 pdefun、pdebc 及 icfun 三个函数以便调用。

下面举例加以说明。

【例 5-15】 求解下列偏微分方程：

$$\begin{cases} \pi^2\dfrac{\partial u}{\partial t}=\dfrac{\partial}{\partial x}\left(\dfrac{\partial u}{\partial x}\right) \\ u(0,t)=0 \\ \dfrac{\partial u}{\partial t}(1,t)=-\pi e^{-t} \\ u(x,0)=\sin(\pi x) \end{cases}$$

对比已给出的偏微分方程与一维偏微分方程的标准形式，得出

$$c\left(x,t,\frac{\partial u}{\partial t}\right)=\pi^2,\quad m=0,\quad f\left(x,t,u,\frac{\partial u}{\partial x}\right)=\frac{\partial u}{\partial x},\quad s\left(x,t,\frac{\partial u}{\partial x}\right)=0 \tag{5-7}$$

调用 pdepe 运算之前，先编写以下 3 个函数以便于在 pdepe 函数指令的使用中加以调用。按照上述已得到的偏微分方程先编写 pdefun 函数(命名为 pdex1pde.m)：

```
function [c,f,s] = pdex1pde(x,t,u,DuDx)
c = pi^2;
f = DuDx;
s = 0;
end
```

接着对比所给的边界条件与边界条件的标准形式，编写边界 bcfun 函数(命名为 pdex1bc.m)，结果如下：

```
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
pl = ul;
ql = 0;
pr = pi*exp(-t);
qr = 1;
end
```

最后还要对比所给的初始条件与初始条件的标准形式，编写初始 icfun 函数（命名为 pdex1ic.m），结果如下：

```
function u0 = pdex1ic(x)
u0 = sin(pi * x);
end
```

现在，可以开始编写程序 exam_5_15.m 调用 pdepe 函数运行，同时将所得数据可视化。

```
m = 0;
x = linspace(0,1,20);
t = linspace(0,2,10);
sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u = sol(:,:,1); % 将解数组赋值给变量 u
surf(x,t,u)
title('Numerical solution')
xlabel('Distance x')
ylabel('Time t')
figure
plot(x,u(end,:))
title('Solution at t = 2')
xlabel('Distance x')
ylabel('u(x,2)')
```

运行程序 exam_5_15 之后，数值解可以绘制成图 5-1。

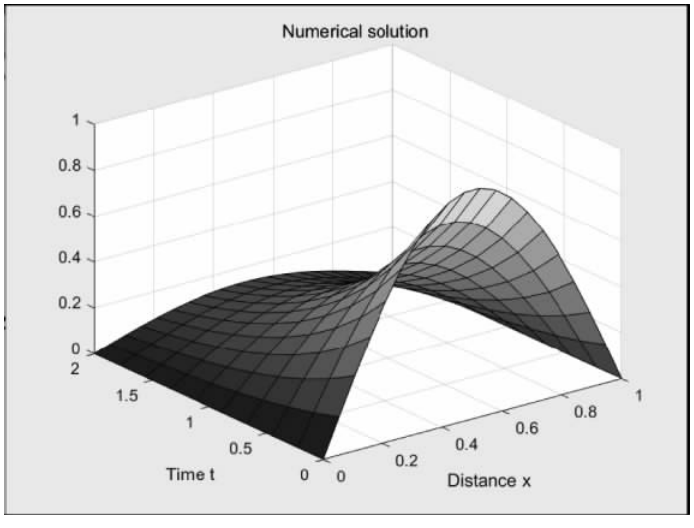

图 5-1　计算结果可视化

由图 5-1 可以直接观察到，t＝0 时的初始条件在 x 轴上是满足的，x＝0 时的边界条件也是满足的，被求的函数 u 值随时间的变化情况一目了然。

2. 一维偏微分方程组的求解

为了进一步了解函数 dsolve 的使用格式，编写偏微分方程组、边界条件和初值条件等函数文件，熟练地解算一维偏微分方程组应用题，下面再举一例加以说明。

【例 5-16】 求解下列偏微分方程组：

$$\begin{cases}\dfrac{\partial u_1}{\partial t}=0.025\dfrac{\partial^2 u_1}{\partial x^2}-F(u_1-u_2)\\[2ex]\dfrac{\partial u_2}{\partial t}=0.18\dfrac{\partial^2 u_2}{\partial x^2}+F(u_1-u_2)\end{cases}$$

其中，$F=e^{5.75x}-e^{-11.56x}$，满足初值条件 $u_1(x,0)=1$，$u_2(x,0)=0$，且满足如下的边界条件：

左边界条件

$$\begin{cases}\dfrac{\partial u_1}{\partial x}(0,t)=0\\[2ex]u_2(0,t)=0\end{cases}$$

右边界条件

$$\begin{cases}u_1(1,t)=1\\[2ex]\dfrac{\partial u_2}{\partial x}(1,t)=0\end{cases}$$

将已给出的偏微分方程组整理得出

$$\begin{bmatrix}1\\1\end{bmatrix}.*\frac{\partial}{\partial t}\begin{bmatrix}u_1\\u_2\end{bmatrix}=\frac{\partial}{\partial x}\begin{bmatrix}0.025\dfrac{\partial u_1}{\partial x}\\[2ex]0.18\dfrac{\partial u_2}{\partial x}\end{bmatrix}+\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}\tag{5-8}$$

对比已给出的偏微分方程与一维偏微分方程的标准形式，容易得出

$$c\left(x,t,\frac{\partial u}{\partial x}\right)=\begin{bmatrix}1\\1\end{bmatrix},\quad m=0$$

$$f\left(x,t,u,\frac{\partial u}{\partial x}\right)=\begin{bmatrix}0.025\dfrac{\partial u_1}{\partial x}\\[2ex]0.18\dfrac{\partial u_2}{\partial x}\end{bmatrix}$$

$$s\left(x,t,\frac{\partial u}{\partial x}\right)=\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

调用 pdepe 运算之前，先编写以下 3 个函数以便于在 pdepe 函数指令的使用中加以调用。按照上述已得到的偏微分方程先编写 pdefun 函数(命名为 pdex2fun.m)：

```
function [c,f,s] = pdex2fun(x,t,u,du)
c = [1;1];
f = [0.025 * du(1);0.18 * du(2)];
temp = u(1) - u(2);
s = [ - 1;1]. * (exp(5.75 * temp) - exp( - 11.56 * temp));
end
```

将已给出的边界条件整理得出

左边界条件$\begin{bmatrix}0\\u_2\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}.*f=\begin{bmatrix}0\\0\end{bmatrix}$，右边界条件$\begin{bmatrix}u_1-1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}.*f=\begin{bmatrix}0\\0\end{bmatrix}$。

接着对比所给的边界条件与边界条件的标准形式，编写边界 bcfun 函数(命名为 pdex2bc.m)，结果如下：

```
function [pl,ql,pr,qr] = pdex2bc(xl,ul,xr,ur,t)
pl = [0;ul(2)];
ql = [1;0];
pr = [ur(1) - 1;0];
qr = [0;1];
end
```

接着对比所给的初始条件与初始条件的标准形式，编写初始条件 icfun 函数(命名为 pdex2ic.m)，结果如下：

```
function u0 = pdex2ic(x)
u0 = [1;0];
end
```

现在，可以开始编写程序 exam_5_16.m 调用 pdepe 函数运行，同时将所得数据可视化。

```
clc
x = 0:0.05:1;
t = 0:0.05:2;
m = 0;
sol = pdepe(m,@pdex2fun,@pdex2ic,@pdex2bc,x,t);
figure('numbertitle','off','name','PDE Demo by Matlabsky')
subplot(211)
surf(x,t,sol(:,:,1))
title('The Solution of u1')
xlabel('x')
ylabel('t')
zlabel('u1')
subplot(212)
surf(x,t,sol(:,:,2))
title('The Solution of u2')
xlabel('x')
ylabel('t')
zlabel('u2')
```

运行程序 exam_5_16 之后，数值解可以绘制成图 5-2。

由图 5-2 可以直接观察到，t=0 时的初始条件在 x 轴上是满足的，x=0 时的边界条件也是满足的，被求的函数 u1 值、函数 u2 值随时间的变化情况一目了然。

偏微分方程的解不仅受方程形式约束，也是由边界条件和初始条件约束的。即偏微分方程、边界条件和初始条件共同考虑才能决定一个确定的解，其符号解析形式的解往往形式复杂，多数以隐函数形式提供，不利于对其解进行定量分析。因而提供数值形式的解并加以可视化也不失其应用的意义。

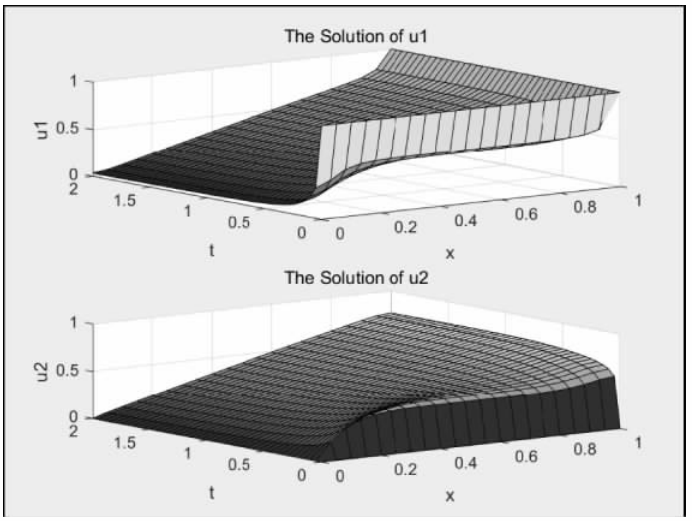

图 5-2　计算结果可视化

5.6　符号运算应用实例

【例 5-17】　求符号矩阵 $\boldsymbol{A}=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 的行列式、逆和特征根。

在 MATLAB 命令窗口中直接输入下列命令行求行列式：

```
>> syms a b c d
>> A = [a b;c d];
>> D = det(A)
D =
a * d - b * c
```

接着直接输入下列命令行求逆：

```
>> B = inv(A)
B =
   [ d/(a * d - b * c), -b/(a * d - b * c)]
[ -c/(a * d - b * c), a/(a * d - b * c)]
```

再接着直接输入下列命令行求特征根：

```
>> S = eig(A)
S =
a/2 + d/2 - (a^2 - 2 * a * d + d^2 + 4 * b * c)^(1/2)/2
   a/2 + d/2 + (a^2 - 2 * a * d + d^2 + 4 * b * c)^(1/2)/2
```

【例 5-18】　定义以下符号矩阵 $\boldsymbol{A}$，试求其逆矩阵 $\boldsymbol{B}$，并验证其逆矩阵 $\boldsymbol{B}$ 的运算结果是否正确？

$$\boldsymbol{A}=\begin{bmatrix} a & h \\ d & k \end{bmatrix}$$

首先定义符号矩阵如下：

```
>> syms a d h k
>> A = [a,h;d,k]
A =
[ a, h]
[ d, k]
```

求符号矩阵的逆：

```
>> B = inv(A)
B =
[ k/(a * k - d * h), -h/(a * k - d * h)]
[ -d/(a * k - d * h), a/(a * k - d * h)]
```

验证逆矩阵的正确性：

```
>> A * B
ans =
[ (a * k)/(a * k - d * h) - (d * h)/(a * k - d * h),                                    0]
[                                    0, (a * k)/(a * k - d * h) - (d * h)/(a * k - d * h)]
>> simplify(A * B)
ans =
[ 1, 0]
[ 0, 1]
```

A * B 是单位矩阵。

```
>> simplify(B * A)
ans =
[ 1, 0]
[ 0, 1]
```

B * A 也是单位矩阵。

MATLAB已提供了多条函数指令用于符号矩阵的运算，常用的矩阵运算函数指令如表5-1所示，这些函数指令的应用极大地减轻了人们在做矩阵运算时的繁重工作量。

表5-1 常用矩阵运算函数指令

函数指令	运算功能	函数指令	运算功能
det(A)	求方阵A的行列式	poly(A)	求矩阵A的特征多项式
inv(A)	求方阵A的逆	rref(A)	求矩阵A的行阶梯形
[V,D]=eig(A)	求A的特征向量V和特征值D	colspace(A)	求矩阵A列空间的基
rank(A)	求A的秩	triu(A)	求矩阵A上三角形

【例 5-19】 创建以下符号表达式 $f(t)$，并求其导数 $f'(t)$。当 $t=1$s 时，$f'(t)$及 $f(t)$的值各是多少？

$$f(t)=\sqrt{2}\cdot 220\cdot\cos\left(100\pi\cdot t+\frac{\pi}{6}\right)$$

```
>> syms t
>> y = sqrt(2) * 220 * cos(100 * pi * t + pi/6)
y =
220 * 2 ^ (1/2) * cos(pi/6 + 100 * pi * t)
```

求表达式 y 的导数：

```
>> dydt = diff(y)
dydt =
 - 22000 * 2 ^ (1/2) * pi * sin(pi/6 + 100 * pi * t)
```

求 y 在 $t=1$ 时的值 y_1：

```
>> y1 = subs(y,t,1)
y1 =
110 * 2 ^ (1/2) * 3 ^ (1/2)
```

将 y_1 值简化为一个有理数：

```
>> eval(y1)
ans =
  269.4439
```

求 y 的导数 dydt 在 $t=1$ 时的值 dydt1：

```
>> dydt1 = eval(subs(dydt,t,1))
dydt1 =
   - 4.8872e + 04
```

【例 5-20】 求以下两个多项式 p_1、p_2 的乘积多项式 $p_{1.2}$对时间 t 的导数，当 $t=5$s 时，$p_{1.2}$多项式的值是多少？再求出多项式 p_1除以多项式 p_2 之后的多项式 $p_{1/2}$对时间 t 的导数。

$$p_1=t^3+5t^2+3t+1,\ p_2=4t^2+2t+6$$

首先将输入多项式 p_1、p_2：

```
>> p1 = [1 5 3 1]
p1 =
     1     5     3     1
>> p2 = [4 2 6]
p2 =
     4     2     6
```

求两个多项式乘积再求导数多项式 p：

```
>> p = polyder(p1,p2)
p =
20    88    84    80    20
```

求导数多项式 p 在 $t=5$ 时的值：

```
>> p5 = polyval(p,5)
p5 =
       26020
```

求多项式 p_1 除以 p_2，再求导数多项式：

```
[Q,D] = polyder(p1,p2)
Q =
     4     4    16    52    16
D =
16    16    52    24    36
```

答案为 Q/D。

【例 5-21】 将函数 $f(t)=\sin(\pi \cdot t)$ 在 $t=1.2$s 处的泰勒级数展开式写出来，并验证其是否正确？

```
>> syms t x
>> f = sin(pi * t)
f =
sin(pi * t)
```

将函数 f 在点 1.2 处展开成 5 阶的泰勒级数 h：

```
>> h = taylor(f,'ExpansionPoint',1.2,'order',5)
h =
(pi^3 * (5^(1/2)/4 + 1/4) * (t - 6/5)^3)/6 - (2^(1/2) * (5 - 5^(1/2))^(1/2))/4 - pi
 * (5^(1/2)/4 + 1/4) * (t - 6/5) + (2^(1/2) * pi^2 * (5 - 5^(1/2))^(1/2) * (t - 6/5)^
2)/8 - (2^(1/2) * pi^4 * (5 - 5^(1/2))^(1/2) * (t - 6/5)^4)/96
```

验证泰勒级数 h 的正确性，先求函数 f 在点 1.2 处的值 f12：

```
>>  f12 = sin(pi * 1.2)
f12 =
   - 0.5878
```

再求泰勒级数 h 在点 1.2 处的值：

```
>> subs(h,t,1.2)
ans =
-(2^(1/2) * (5 - 5^(1/2))^(1/2))/4
```

简化：

```
>> eval(ans)
ans =
    -0.5878
```

eval(ans)= f12=-0.5878，证明泰勒级数 h 是正确的。

【例 5-22】 已知隐函数关系式 $y=\ln(t+y)$，求 $y'(t)$，并给出 $t=3$s 时的 $y'(t)$值。

先输入隐函数 F：

```
>> syms t y
>> F = log(t + y) + y
F =
y + log(t + y)
```

求 F 的导数：

```
>> dt = -diff(F,t)/diff(F,y)
dt =
-1/((1/(t + y) + 1)*(t + y))
```

当 $t=3$ 时，需要知道 y 对应的值，而后代入上式即可求出 F 的导数值；

当 $t=3$ 时，y 对应的值：

```
>>  y = fsolve('nonfun',1,optimset('Display','off'))
y =
1.5052
```

所构造的 nonfun.m 函数代码如下：

```
function [n ] = nonfun( m )
y = m;
n = exp(y) - y - 3;
end
```

将 $t=3$，$y=1.5052$ 代入 F 的导数，得

```
>> -1/((1/( 1.5052 + 3) + 1)*( 1.5052 + 3))
ans =
    -0.1816
```

【例 5-23】 已知积分上限函数 $f(x)$，求导数 $f'(x)$。

$$f(x)=\int_0^{\frac{x}{2}}(5t^2+3)\mathrm{d}t$$

将积分的核函数输入：

```
syms x t
>> f = 5*t^2 + 3
```

```
f =
5 * t^2 + 3
```

再求积分上限函数的导数：

```
>> diff(int(f,0,x/2),x)
ans =
(5 * x^2)/8 + 3/2
```

结果与核函数是不同的。

【例 5-24】 定积分 $s=\int_{-\infty}^{5}\frac{2}{\sqrt{\pi}}e^{-\frac{t^2}{2}}dt$ 的值是多少？

将积分的核函数输入：

```
>> syms t
>> f = 2/sqrt(pi) * exp( - t^2/2)
f =
(5081767996463981 * exp( - t^2/2))/4503599627370496
```

求定积分：

```
>> s = int(f, - inf,5)
s =
(5081767996463981 * 2^(1/2) * pi^(1/2) * (erf((5 * 2^(1/2))/2) + 1))/9007199254740992
```

对结果简化：

```
>> eval(s)
ans =
    2.8284
```

【例 5-25】 求分段函数 $f(t)=\sin(\pi t)u(t)+\sin(\pi(t-1))u(t-1)$ 的 laplace 变换 $F(s)$，$F(s)$ 的 laplace 逆变换函数又是怎样的？

将分段函数 f 输入：

```
>> syms t
>> f = sin(pi * t) * heaviside(t) + sin(pi * (t - 1)) * heaviside(t - 1)
f =
heaviside(t - 1) * sin(pi * (t - 1)) + sin(pi * t) * heaviside(t)
```

求 f 的 laplace 变换：

```
>> Fs = laplace(f)
Fs =
pi/(s^2 + pi^2) + (pi * exp( - s))/(s^2 + pi^2)
```

求 $F(s)$ 的 laplace 逆变换：

```
>> ft = ilaplace(Fs)
ft =
sin(pi * t) + heaviside(t - 1) * sin(pi * (t - 1))
```

对函数 f 和函数 ft 进行比较，两者是一样的。

【例 5-26】 以下线性方程组的符号解是怎样的？

$$\begin{cases} ax + by = 3 \\ cx + dy = 4 \end{cases}$$

建立方程组：

```
>> syms x y a b c d
>> eqn1 = a * x + b * y == 3
eqn1 =
a * x + b * y == 3
>> eqn2 = c * x + d * y == 4
eqn2 =
c * x + d * y == 4
```

求方程组的解：

```
>> [Sx,Sy] = solve(eqn1,eqn2,x,y)
Sx =
 -(4 * b - 3 * d)/(a * d - b * c)
Sy =
(4 * a - 3 * c)/(a * d - b * c)
```

【例 5-27】 常微分方程 $ay'(t)+bt \cdot y(t)=0, y(0)=1$ 的符号解是怎样的？

定义 Dy：

```
>> syms t y(t) a b
>> Dy = diff(y)
```

求常微分方程符号解：

```
>> yt = dsolve(Dy == - b * t * y/a, y(0) == 1)
yt =
exp( - (b * t ^ 2)/(2 * a))
```

5.7 符号运算综合实例

5.7.1 符号函数可视化应用

通常函数表达式以显式的方式列出时较易被人们理解和分析，画图也很方便，只需定义好自变量取值，调用相应的图形绘制函数指令运行便可以了。但函数表达式以隐函数的方式列出时，其图形绘制之前是否需要整理变形，将函数表达式以显式的方式列出

之后再绘制其图形呢？答案是无须如此。MATLAB提供了一组以ez打头的绘图指令，可以方便用户以隐函数的方式直接进行图形绘制，这被称为符号函数可视化。

【例5-28】 绘制以下参数方程表示的三维图形，t的范围为$[0,20\pi]$。

$$\begin{cases} x = t\sin(t) \\ y = t\cos(t) \\ z = t \end{cases}$$

在MATLAB命令窗口中直接输入下列命令行绘制3D图：

```
>> syms t
>> ezplot3(t * sin(t),t * cos(t),t,[0,20 * pi])
```

所绘制的图形如图5-3所示。

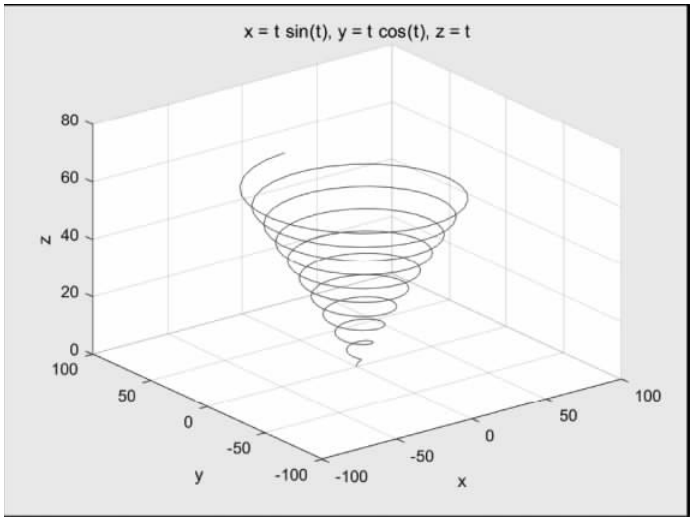

图5-3 符号函数绘图

MATLAB已提供了多条函数指令用于符号函数的绘图，均以ez开头，常用的符号函数的绘图指令如表5-2所示，这些符号函数绘图指令极大地方便了函数绘图工作，特别是隐函数的绘图。

表5-2 常用符号函数绘图指令

函数指令	运算功能	函数指令	运算功能
ezplot	绘制二维曲线图	ezsurfc	绘制带等位线的曲面图
ezplot3	绘制三维曲线图	ezmesh	绘制网线图
ezpolar	绘制极坐标曲线图	ezmeshc	绘制带等位线的网线图
ezsurf	绘制曲面图	ezcontour	绘制等位线图

5.7.2 符号积分应用

MATLAB提供的符号积分功能强大且应用广泛，而符号函数绘图能直接绘图。

【例 5-29】 绘制函数 $y(t)=0.6\mathrm{e}^{-\frac{t}{3}}\cos\frac{\sqrt{3}}{4}t$ 及其积分上限函数 $s(t)=\int_0^t y(t)\mathrm{d}t$ 的图形。

编制名为 exint.m 的程序如下：

```
syms t tao
y = 0.6 * exp( - t/3) * cos(sqrt(3)/4 * t);
s = subs(int(y,t,0,tao),tao,t);
subplot(2,1,1)
ezplot(y,[0,4 * pi]),ylim([ - 0.2,0.7])
grid on
subplot(2,1,2)
ezplot(s,[0,4 * pi]),ylim([ - 0.2,1.1])
grid on
```

之后在 MATLAB 命令窗口中运行 exam_5_29.m，得到函数图形如图 5-4 所示。

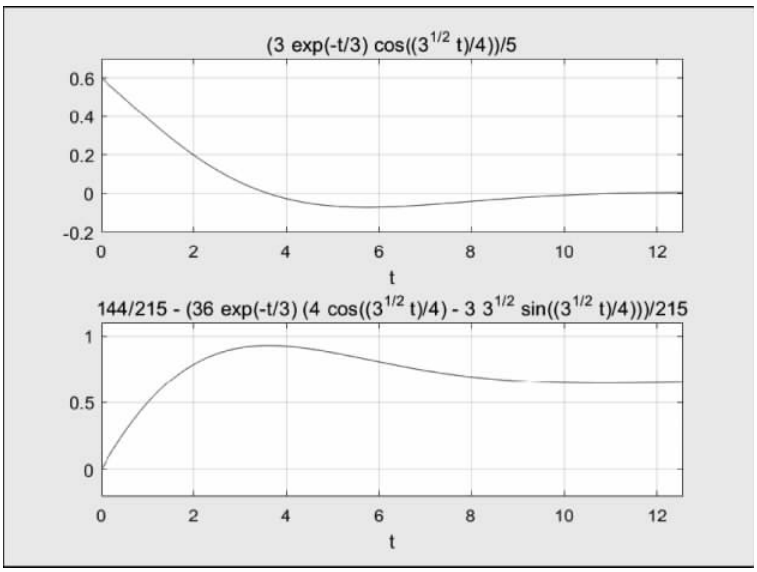

图 5-4 符号积分上限函数绘图

观察图 5-4 的两张子图，第一张子图为 $y(t)$ 的曲线图，图的顶部显示了由程序所定义的函数解析式；第二张子图为 $s(t)$ 的曲线图，图的顶部显示了由 int 函数指令符号计算出的函数解析式。比较子图 1 和子图 2，可以体会到函数积分后图形的直观意义，$s(t)$ 曲线图的顶点正好对应于 $y(t)$ 曲线图的第一次过零点。

5.7.3 符号卷积应用

依据线性时不变系统理论，卷积运算是计算线性系统输出的主要计算方法，卷积定理则揭示了时域卷积（或乘积）运算与变换域乘积（或卷积）运算之间的关系。在工程领域，无论是对线性时不变系统进行分析还是设计滤波器，都要运用到卷积的知识。

【例 5-30】 已知线性时不变系统的系统(传输)函数 $H(s)$如下,在系统输入为单位阶跃信号 $u(t)$时,试求系统的输出信号 $y(t)$。

$$H(s)=\frac{2}{(s+1)(s+3)}$$

依据线性时不变系统理论,系统输出为

$$y(t)=u(t)\circledast h(t)=\int_0^t u(\tau)*h(t-\tau)\mathrm{d}\tau \tag{5-9}$$

上式中的⊛符号代表的是卷积符号。$h(t)$是系统的单位冲激响应,可由下列命令行求出(对 $H(s)$做拉普拉斯逆变换):

```
>> syms s t
>> hs = 2/(s + 1)/(s + 3)
hs =
2/((s + 1) * (s + 3))
```

接着在命令窗口中输入:

```
>> ht = ilaplace(hs,s,t)
ht =
exp( - t) - exp( - 3 * t)
```

下面便进行符号卷积运算,直接在窗口中输入下列命令行:

```
>> syms tao
>> yt = int(heaviside(tao) * subs(ht,t,t - tao),tao,0,t)
yt =
   (sign(t)/2 + 1/2) * (exp( - 3 * t)/3 - exp( - t) + 2/3)
```

上面的解答完整明了,其中 sign(t)为符号函数,改写成数学符号表达如下:

$$y(t)=\frac{\mathrm{e}^{-3t}}{3}-\mathrm{e}^{-t}+\frac{2}{3},\quad t\geqslant 0$$

5.7.4 符号积分变换应用

函数积分变换的重要意义之一便是提供了微分方程和偏微分方程的变换域求解方法,通常是变换到频域或复频域内进行求解,变换域求解方法有效地降低了微分方程和偏微分方程的求解难度,在工程各领域具有广泛的应用。

【例 5-31】 利用拉普拉斯变换方法求解以下波动方程的定解问题。

$$\begin{cases}\dfrac{\partial^2 u}{\partial t^2}=a^2\dfrac{\partial^2 u}{\partial x^2}, & x>0,t>0\\ u(x,0)=0, & \dfrac{\partial u(x,0)}{\partial t}=0\\ u(0,t)=\varphi(t), & \lim\limits_{x\to\infty}u(x,t)=0\end{cases}$$

设函数 $u(x,t)$ 关于 t 取拉普拉斯变换后为 $U(x,s)$，则对上面的波动方程两边关于 t 均取拉普拉斯变换，方程左边关于 t 的变换结果如下：

$$\mathrm{L}\left[\frac{\partial^2 u}{\partial t^2}\right]=s^2\mathrm{L}[u(x,t)]-su(x,0)-\frac{\partial u(x,0)}{\partial t}=s^2U(x,s) \tag{5-10}$$

方程右边关于 t 变换结果如下：

$$\mathrm{L}\left[a^2\frac{\partial^2 u}{\partial x^2}\right]=a^2\frac{\partial^2}{\partial x^2}\mathrm{L}[u(x,t)]=a^2\frac{\partial^2}{\partial x^2}U(x,s) \tag{5-11}$$

边界条件也关于 t 取拉普拉斯变换后结果如下：

$$\mathrm{L}[u(0,t)]=\phi(s),\quad \mathrm{L}[\lim_{x\to\infty}u(x,t)]=\lim_{x\to\infty}\mathrm{L}[u(x,t)]=\lim_{x\to\infty}U(x,s)=0$$

于是，利用拉普拉斯变换方法将波动方程定解问题成功转变为常微分边值问题如下：

$$\begin{cases}a^2\dfrac{\partial^2}{\partial x^2}U(x,s)-s^2U(x,s)=0\\ U(0,s)=\phi(s),\lim\limits_{x\to\infty}U(x,s)=0\end{cases}$$

以上方程的求解可以参照前面常微分方程求解的例 5-12，在 MATLAB 命令窗口中直接输入以下程序行对微分符号 Du 及 D2u 进行定义：

```
>> syms a s u(x)
>> Du = diff(u)
Du(x) =
diff(u(x), x)
>> D2u = diff(Du)
D2u(x) =
diff(u(x), x, x)
```

再直接输入以下程序行求解：

```
>> u = dsolve(D2u == (s/a)^2 * u)
u =
C7 * exp( - (s * x)/a) + C8 * exp((s * x)/a)
```

实质求出的是 $U(x,s)$ 的通解含有两个常数，还需要进一步求解，通过将边值条件代入之后，可以推导出：

$$C7=\Phi(s),\quad C8=0$$

于是 $U(x,s)$ 的解为

$$U(x,s)=\Phi(s)*\exp(-(s*x)/a)$$

而 $u(x,t)$ 的求解只需要对 $U(x,s)$ 关于 s 做拉普拉斯逆变换便可以了，但 MATLAB 做这个逆变换会遇到一个问题：$\Phi(s)$ 并非特定的函数，那么对 $\Phi(s)$ 关于 s 做拉普拉斯逆变换其结果会如何呢？

试在 MATLAB 中输入如下程序行：

```
>> syms fai(s)
>> ilaplace(fai(s))
ans =
ilaplace(fai(s), s, t)
```

从上面的结果可以看出，MATLAB只能给出ilaplace(fai(s), s, t)的表示，而无法按照前面的假设，给出符合人们思维的直接答案：$\varphi(t)=L^{-1}[\phi(s)]$。

在理解了MATLAB的局限性之后，下面对$U(x,s)$关于s做拉普拉斯逆变换：

```
>> syms fai(s) t x a
>> ilaplace(fai(s) * exp( - (x/a) * s),s,t)
ans =
ilaplace(exp( - (s * x)/a) * fai(s), s, t)
```

只能得到上面这样不完全的解答，之所以这样，是因为$U(x,s)$中的符号太多了，我们可以对不参与拉普拉斯逆变换的x、a符号数值化，例如令$x=1000$，$a=314$，将其代入再求拉普拉斯逆变换：

```
syms fai(s) t
>> ilaplace(exp( - (s * 1000)/314) * fai(s), s, t)
ans =
heaviside(t - 1000/314) * ilaplace(fai(s), s, t - 1000/314)
```

上面的这个解答就完全容易理解了，改写成数学符号表达便是

$$u(x,t)=\begin{cases}0, & t\leqslant\dfrac{x}{a}\\ \varphi\left(t-\dfrac{x}{a}\right), & t>\dfrac{x}{a}\end{cases}$$

以上便是波动方程的完整解析解。从以上解偏微分方程的过程可以感受到，虽然MATLAB的符号运算能力强大，但其始终是居于辅助运算的地位，现阶段人的思维仍然是机器无法取代的。

5.8 本章小结

本章简单介绍了MATLAB符号运算的特点，详细介绍了符号对象的创建和使用、符号多项式函数运算、符号微积分运算和符号方程求解。通过大量应用实例的讲解，读者可以更加深刻地认知MATLAB在符号运算中的应用。