第 3 章　符号运算功能

MATLAB 自产生之日起就在数值计算功能上独占鳌头，广受各专业计算人员的欢迎。但是由于在数学、物理及力学等各种科研和工程应用中还经常遇到符号运算的问题，因此，一部分 MATLAB 用户还不得不同时掌握另一种符号计算语言，如 Maple，Mathematic，MathCAD 等，这就带来了一些不便。为了解决这个问题，Mathworks 公司于 1993 年从加拿大滑铁卢大学购入了 Maple 的使用权，并在此基础上，利用 Maple 的函数库，开发了 MATLAB 语言的又一重要工具箱——符号数学工具箱（Symbolic Math Toolbox）。从此，MATLAB 便集数值计算、符号计算和图形处理三大基本功能于一体，成为在数学计算各语言中功能最强、操作最简单和最受用户喜爱的语言。

MATLAB 的符号计算由符号数学工具箱承担，这个工具箱的核心是 Maple，工具箱下还有很多 MATLAB 的函数，它们负责 MATLAB 与 Maple 之间的信息传递。当你在 MATLAB 中首次执行符号命令时，MATLAB 在后台启动 Maple 核心，然后通过相关 M 文件调用该核心进行符号运算。

这一机制很灵活，一方面可以通过 M 文件间接调用 Maple，此过程你不必了解 MATLAB 符号计算的内在机制，上手迅速，另一方面，也可以越过这些 M 文件提供的命令，直接从 MATLAB 调用 Maple，能够完成更复杂的工作，当然，这需要你同时熟悉 MATLAB 和 Maple 两个软件。

需要注意的是，自从 MATLAB 2008b（7.7 版本）起，符号数学工具箱默认的内核就不再采用 Maple，而是改为了 MuPAD，当然用户也可以选择采用 Maple 计算引擎。但是令人欣慰的是，基本的符号运算的语法并没有改变（这得益于介于 MATLAB 和符号工具箱内核间的 M 文件）。但是如果你需要在 MATLAB 应用 MuPAD 更强大的功能，则需要下载一个完整的 MuPAD。

MATLAB2010a 版本中，不再支持 Maple 引擎。

在 MATLAB 中实现符号计算功能主要有以下三种途径。

- 通过调用 MATLAB 自己开发的各种功能函数进行常用的符号运算。这些功能主要包括符号表达式与符号矩阵的基本操作、符号矩阵的运算、符号微积分运算、符号线性方程求解、符号微分方程求解、特殊数学符号函数、符号函数图形等，这些内容将在本章进行详细地介绍。对于众多喜爱和熟悉 MATLAB 的用户来说，这些操作十分简单，很容易学习和掌握。
- MATLAB 语言中的符号计算功能已经很强大了，但为了给一些特殊专业的人员提供方便，MATLAB 中还保留着与专用的符号数学工具的接口，以期实现更多功能。
- 对那些用惯了计算器的用户来说，MATLAB 同样是最佳的选择，因为 MATLAB 还提供了符号函数计算器（Function Calculator）功能。计算器上提供了不超过两个符号函

数的基本运算和微积分运算的功能，而且还有函数可视化图形窗。虽然功能比较简单，但使用方便，操作简易，可视化效果好。在本章也将对它进行必要的介绍。

另外，在本章的最后还将介绍使用 MATLAB 符号计算功能的诸多注意事项。

3.1 符号表达式的生成

在数值计算中，包括输入、输出及中间过程，变量都是数值变量。而在符号运算过程中，变量都以字符形式保存和运算，即使是数字也被当做字符来处理。

符号表达式包括符号函数和符号方程，两者的区别在于前者不包括等号，而后者必须带等号，但它们的创建方式是相同的。最简单易用的创建方法和 MATLAB 中的字符串变量的生成方法相同。

- 创建符号函数

【例如】

```
>> f='log(x)'
f =
    log(x)
```

- 创建符号方程

【例如】

```
>> eqation='a*x^2+b*x+c=0';
```

- 创建符号微分方程

【例如】

```
>> diffeq='Dy-y=x';
```

说明

- 注意由这种方法创建的符号表达式对空格是很敏感的。因此，不要在字符间乱加空格符，否则在其他地方调用此表达式的时候会出错。
- 由于符号表达式在 MATLAB 中被看成是 1×1 阶的符号矩阵，因此，它也可用 sym 命令来创建。

【例如】

```
>> f=sym('sin(x)')
f =
    sin(x)
>> f=sym('sin(x)^2=0')
f =
    sin(x)^2=0
```

另外一种创建符号函数的方法是用 syms 命令。

【例如】

```
>> syms x
>> f=sin(x)+cos(x)
f =
   sin(x)+cos(x)
```

说明　用此方法来创建的符号函数同其他方法创建的符号函数效果相同，但此方法不能用来创建符号方程。

3.2　符号和数值之间的转换

有时符号运算的目的是得到精确的数值解，这样就需要对得到的解析解进行数值转换。在 MATLAB 中这种转换主要由两个函数实现，即 digits 和 vpa。而这两个函数在实际中经常同变量替换函数 subs 配合使用。

- digits 函数

它的调用格式如下：

- digits(D)　函数设置有效数字个数为 D 的近似解精度。

- vpa 函数

它的调用格式如下：

- R = vpa(S)　符号表达式 S 在 digits 函数设置精度下的数值解。
- vpa(S, D)　符号表达式 S 在 digits(D)精度下的数值解。

- subs 函数

此函数更全面的使用方法及功能将在下一节中介绍。本节为了说明以上两个函数的效果，在此先对此函数的主要格式给予说明。它的常用调用格式如下：

- subs(S, OLD, NEW)　将符号表达式中的 OLD 变量替换为 NEW 变量。

【例 3.1】 求方程 $3x^2 - e^x = 0$ 的精确解和各种精度的近似解。

解：

```
>> s=solve('3*x^2-exp(x)=0')
s =
-2*lambertw(0, -3^(1/2)/6)
>> vpa(s)
ans =
0.91000757248870906065733829575937
>> vpa(s,6)
ans =
0.910008
```

【例 3.2】 设函数为 $f(x)= x- \cos(x)$。求此函数在 $x = \pi$ 点值的各种精度的数值近似形式。

解：

```
>> x=sym('x');                    %定义函数
>> f=x-cos(x)
f =
   x-cos(x)
>> f1=subs(f,'pi',x)              %字符替代
f1 =
    pi+1
>> digits(25)                     %各精度显示
>> vpa(f1)
ans =
4.14159265358979323846264
>> double(f1)
```

```
ans =
   4.1416
```

3.3 符号函数的运算

这里“函数的运算”是指针对函数的运算，本节主要介绍复合函数运算和反函数运算。

3.3.1 复合函数运算

若函数 $z = z(y)$的自变量 y 又是 x 的函数 $y = y(x)$，则求 z 对 x 的函数的过程称为复合函数运算。在 MATLAB 中，此过程可由功能函数 compose 来实现。

- compose　函数复合函数
  - compose(f, g)　返回当 $f = f(x)$和 g = g(y)时的复合函数 f(g(y))，这里 x 为 findsym 定义的 f 的符号变量，y 为 findsym 定义的 g 的符号变量。
  - compose(f, g, z)　返回的复合函数以 z 为自变量。
  - compose(f, g, x, z)　返回复合函数 f(g(z))，且使得 x 为 f 的独立变量。也就是说如果 f = cos(x/t)，则 compose(f, g, x, z)返回 cos(g(z)/t)，而 compose(f, g, t, z)返回 cos(x/g(z))。
  - compose(f, g, x, y, z)　返回复合函数 f(g(z))并使得 x 为 f 的独立变量，y 为 g 的独立变量。若 f = cos(x/t)且 g = sin(y/u)，compose(f, g, x, y, z)返回 cos(sin(z/u)/t)，而 compose(f, g, x, u, z)返回 cos(sin(y/z)/t)。

【例 3.3】 复合函数的运算示例。

```
>> syms x y z t u;
>> f = 1/(1 + x^2);
>> g = sin(y);
>> h = x^t;
>> p = exp(-y/u);
>> compose(f,g)
ans =
    1/(sin(y)^2 + 1)
>> compose(f,g,t)
ans =
     1/(sin(t)^2 + 1)
>> compose(h，g，x，z)
ans =
      sin(z)^t
>> compose(h,g,t,z)
ans =
    x^sin(z)
>> compose(h,p,x,y,z)
ans =
    (1/exp(z/u))^t
>> compose(h,p,t,u,z)
ans =
      x^(1/exp(y/z))
```

3.3.2　反函数的运算

反函数运算也是符号函数运算比较重要的一部分，在 MATLAB 中由函数 finverse 实现。

- finverse　反函数运算函数
  - g = finverse(f)　符号函数 f 的反函数。f 为一符号函数表达式，单变量为 x。则函数 g 也为一符号函数，且使得 g(f(x))= x。
  - g = finverse(f,v)　返回的符号函数表达式的自变量为 v，这里 v 为一符号，是表达式的向量变量。则 g 的表达式要使得 g(f(v))= v。当 f 包括不止一个变量时最好使用此型。

【例 3.4】　反函数运算示例。

```
>> syms x y;
>> f = x^2+y;
>> finverse(f,y)
ans =
-x^2+y
>> finverse(f)
Warning: Functional inverse is not unique.
> In F:\MATLAB\R2011b\toolbox\symbolic\symbolic\symengine.p>symengine at 54
  In sym.finverse at 41
ans =
(-y+x)^(1/2)
```

说明　此时，由于没有指明自变量，MATLAB 给出警告信息，且以默认变量 x 给出结果。

3.4　符号矩阵的创立

在 MATLAB 中创建符号矩阵的方法和创建数值矩阵的形式很相似，只不过要用到符号定义函数 sym，下面介绍使用此函数创建符号函数的几种形式。

3.4.1　使用 sym 函数直接创建符号矩阵

此方法和直接创建数值矩阵的方法几乎完全相同。矩阵元素可以是任何不带等号的符号表达式，各符号表达式的长度可以不同；矩阵元素之间可用空格或逗号分隔。

【例如】

```
a= sym('[1/s+x,sin(x) cos(x)^2/(b+x);9,exp(x^2+y^2),log(tanh(y))]')
a =
[ x + 1/s,        sin(x), cos(x)^2/(b + x)]
[      9, exp(x^2 + y^2),     log(tanh(y))]
```

3.4.2　用创建子阵的方法创建符号矩阵

此方法是仿照 MATLAB 的字符串矩阵的直接输入法设计的。这种方法不需要调用 sym 命令，但要保证同一列的各元素字符串具有相同的长度。为此，在较短字符串的前后可用空格符补充。

【例如】

```
>> ms=['[1/S, sin(x)]';'[1  , exp(x)]']
```

```
ms =
    [1/S, sin(x)]
    [1  , exp(x)]
>> b=[a;'[exp(-i),3,x^3+y^9]']
b =
    [  x + 1/s,        sin(x), cos(x)^2/(b + x)]
    [       9, exp(x^2 + y^2),     log(tanh(y))]
    [ 1/exp(i),            3,       x^3 + y^9]
```

3.4.3 将数值矩阵转化为符号矩阵

在 MATLAB 中，数值矩阵不能直接参与符号运算，必须先转化为符号矩阵。

注意 不论数值矩阵的元素原先是用分数还是用浮点数表示，转化后的符号矩阵都将以最接近的精确有理数形式给出。

【例如】

```
>> a=[2/3, sqrt(2), 0.222;1.4, 1/0.23, log(3)]
a =
   0.6667   1.4142   0.2220
   1.4000   4.3478   1.0986
>> b=sym(a)
b =
[ 2/3, 2^(1/2),                        111/500]
[ 7/5,  100/23, 2473854946935173/2251799813685248]
```

3.4.4 符号矩阵的索引和修改

MATLAB 的矩阵索引和修改同数值矩阵的索引和修改完全相同，即用矩阵的坐标括号表达式实现。

【例 3.5】 对上例中的矩阵 ***b*** 的索引和修改。

```
>> b(2,3)                  %矩阵的索引
ans =
    2473854946935173/2251799813685248
>> b(2,3)='99'             %矩阵的修改
b =
[    2/3，sqrt(2)，111/500]
[    7/5，100/23，  99    ]
```

3.5 符号矩阵的运算

3.5.1 基本运算

MATLAB 把符号矩阵的基本运算符与数值矩阵的运算符统一起来，大大方便了用户。

1．符号矩阵的四则运算

- 矩阵的加（+）、减（−）法

【例如】

```
>> a=sym('[1/x,1/(x+1);1/(x+2),1/(x+3)]');
>> b=sym('[x,1;x+2,0]');
>> b-a
ans =
[         x - 1/x, 1 - 1/(x + 1)]
[ x - 1/(x + 2) + 2,    -1/(x + 3)]
```

- 矩阵的乘（*）、除（/、\）法

【例如】

```
>> a\b
ans =
[ -x*(2*x^2 + 7*x + 6),  (x*(x^2 + 3*x + 2))/2]
[  2*(x + 1)^2*(x + 3), -(x*(x + 1)*(x + 3))/2]
```

- 矩阵的转置（'）

【例如】

```
>> a'
ans =
    [      1/conj(x), 1/(conj(x) + 2)]
    [ 1/(conj(x) + 1), 1/(conj(x) + 3)]
```

2. 符号矩阵的行列式运算

【例如】

```
>> det(a)
ans =
    2/(x*(x + 1)*(x + 2)*(x + 3))
```

3. 符号矩阵的逆

【例如】

```
>> inv(b)
ans =
    [ 0,  1/(x + 2)]
    [ 1, -x/(x + 2)]
```

4. 符号矩阵的秩

【例如】

```
>> rank(a)
ans =
    2
```

5. 符号矩阵的幂运算

【例如】

```
>> a^2
ans =
    [       1/((x + 1)*(x + 2)) + 1/x^2, 1/(x*(x + 1)) + 1/((x + 1)*(x + 3))]
    [ 1/(x*(x + 2)) + 1/((x + 2)*(x + 3)),   1/(x + 3)^2 + 1/((x + 1)*(x + 2))]
```

6．符号矩阵的指数运算

- 符号矩阵的“数组指数”运算由函数 exp 实现。

【例如】

```
>> exp(b)
ans =
    [    exp(x), exp(1)]
    [ exp(x + 2),     1]
```

- 符号矩阵的“矩阵指数”运算由函数 expm 来实现，此处不再举例。

3.5.2 矩阵分解

有关矩阵分解的函数也做了同样的简化，看来数值运算和符号运算在运算符上的大同化是大势所趋了。

1．符号矩阵的特征值分解函数 eig

【例如】

```
>> [x, y]=eig(b)
    x =
    [ (x/2 - (x^2 + 4*x + 8)^(1/2)/2)/(x + 2), (x/2 + (x^2 + 4*x + 8)^(1/2)/2)/(x + 2)]
    [                                      1,                                      1]
    y =
    [ x/2 - (x^2 + 4*x + 8)^(1/2)/2,                          0]
    [                          0, x/2 + (x^2 + 4*x + 8)^(1/2)/2]
```

2．符号矩阵的奇异值分解函数 svd

【例如】

```
>> syms t real       %定义 t 为符号变量
>> A = [0 1; -1 0];
>> E = expm(t*A)
E =
    [      1/(2*exp(t*i)) + exp(t*i)/2, i/(2*exp(t*i)) - (exp(t*i)*i)/2]
    [ - i/(2*exp(t*i)) + (exp(t*i)*i)/2,    1/(2*exp(t*i)) + exp(t*i)/2]
>> sigma = svd(E)
sigma =
    ((1/(2*exp(t*i)) + exp(t*i)/2)^2 + (i/(2*exp(t*i)) - (exp(t*i)*i)/2)^2)^(1/2)
    ((1/(2*exp(t*i)) + exp(t*i)/2)^2 + (i/(2*exp(t*i)) - (exp(t*i)*i)/2)^2)^(1/2)
>> simplify(sigma)
ans =
    1
    1
```

3．符号矩阵的约当标准型函数 jordan

【例如】

```
>> a=sym('[1 1 2;0 1 3;0 0 2]')
a =
    [ 1，1，2]
```

```
     [ 0, 1, 3]
     [ 0, 0, 2]
>> [x, y]=jordan(a)
x =
     [  5, -5, -5]
     [  3,  0, -5]
     [  1,  0,  0]
y =
     [ 2, 0, 0]
     [ 0, 1, 1]
     [ 0, 0, 1]
```

4．符号矩阵的三角抽取函数 diag，tril，triu

【例如】

```
>> z=sym('[x*y x^a sin(y);t^a log(y) b;y exp(t) x]');
>> triu(z)
ans =
     [ x*y,    x^a, sin(y)]
     [   0, log(y),     b]
     [   0,     0,     x]
>> diag(z)
ans =
      x*y
     log(y)
       x
>> tril(z,-1)
ans =
     [     0,     0,    0]
     [   t^a,     0,    0]
     [     y, exp(t),    0]
```

3.5.3　矩阵的空间运算

1．符号矩阵的列空间运算函数 colspace

【例如】

```
>> colspace(a)          %  “a”见上页
ans =
     [ 0, 1, 0]
     [ 0, 0, 1]
     [ 1, 0, 0]
```

2．符号矩阵的零空间运算函数 null

- Z=null(a)　由奇异值分解所得的零空间的正交基。
- Z=null(A,'r')　零空间的有理基，A×Z 为零。若 A 是一个整数元素的小矩阵，元素 r 为小整数的比。

【例如】

```
>> a=[     1     2     3
```

```
          1    2    3
          1    2    3];
>> null(a)
ans=
       0   0.9636
  -0.8321  -0.1482
   0.5547  -0.2224
>> null(a,'r')
ans=
         -2   -3
          1    0
          0    1
```

3.5.4 符号矩阵的简化

符号工具箱中还提供了符号矩阵因式分解、展开、合并、简化及通分等符号操作函数。下面将一一进行介绍。

1．因式分解

factor　符号因式分解函数，调用格式：

- factor(S)　输入变量 S 为一符号矩阵，此函数将因式分解此矩阵的各个元素。如果 S 包含的所有元素为整数，则计算最佳因式分解式。为了分解大于 2^{52} 的整数，可使用 factor(sym('N'))。

【例如】

- 符号表达式的分解

```
>> syms x
>> factor(x^9-1)
ans =
    (x - 1)*(x^2 + x + 1)*(x^6 + x^3 + 1)
```

- 大整数的分解

```
>> factor(sym('12345678901234567890'))
ans =
    2*3^2*5*101*3541*3607*3803*27961
```

2．符号矩阵的展开

expand　符号矩阵的展开函数，调用格式如下：

- expand(S)　对符号矩阵的各元素的符号表达式进行展开。此函数经常用在多项式的表示式中，也常用于三角函数、指数函数和对数函数的展开中。

【例如】

```
>> syms x y
```

- 多项式的展开

```
>> expand((x+1）^3)
ans=
     x^3+3*x^2+3*x+1
```

● 三角函数展开

```
>> expand(sin(x+y))
ans=
   cos(x)*sin(y) + cos(y)*sin(x)
```

3．同类式合并

collect　合并系数函数。它的调用格式如下：

➢ collect(S,v)　将符号矩阵 S 中的各元素的 v 的同幂项系数合并。

➢ collect(S)　对由 findsym 函数返回的默认变量进行同类项合并。

【例如】

```
>> syms x y;
>> collect(x^2*y + y*x - x^2 - 2*x)
ans=
     (y-1)*x^2+(y-2)*x
```

4．符号简化

在 MATLAB 中进行符号简化可由函数 simple 和 simplify 实现。

simple　用于寻找符号矩阵或符号表达式的最简型。它的调用格式如下：

➢ simple(S)　对表达式 S 尝试多种不同的算法简化，以显示 S 表达式的长度最短的简化形式。若 S 为一矩阵，则结果是全矩阵的最简型，而非每个元素的最简型。

➢ [R，HOW] = simple(S)　返回的 R 为简化型，HOW 为简化过程中使用的主要方法。

【例 3.6】 通过计算可得表 3.1，可见此函数所用到的方法十分广泛。

表 3.1　符号简化函数示例表

S	R	HOW
cos(x)^2+sin(x)^2	1	combine(trig)
2*cos(x)^2−sin(x)^2	3*cos(x)^2−1	simplify
cos(x)^2−sin(x)^2	cos(2*x)	combine(trig)
cos(x)+(−sin(x)^2)^(1/2)	cos(x)+i*sin(x)	radsimp
cos(x)+i*sin(x)	exp(i*x)	convert(exp)
(x+1)*x*(x−1)	x^3−x	collect(x)
x^3+3*x^2+3*x+1	(x+1)^3	factor
cos(3*acos(x))	4*x^3−3*x	expand

simplify　符号简化函数。它的调用格式如下：

➢ simplify(S)　简化符号矩阵的每一个元素。

【例如】

```
>> syms x
>> simplify(sin(x)^2 + cos(x)^2)
ans=
    1
>> syms c alpha beta
>> simplify(exp(c*log(sqrt(alpha+beta))))
ans =
    (alpha + beta)^(c/2)
```

5．分式通分

numden　求解符号表达式的分子和分母。其常用调用格式如下：

- [N, D] = numden(A)　把 A 的各元素转换为分子和分母都是整系数的最佳多项式型。

【例如】

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

6．符号表达式的“秦九昭型”重写

horner　“秦九昭型”多项式表达式函数，调用格式：

- horner(P)　将符号多项式转换成嵌套形式表达式。

【例如】

```
>> horner(x^3-6*x^2+11*x-6)
ans=
    x*(x*(x - 6) + 11) - 6
```

3.6　符号微积分

微积分是大学教学、科研及工程应用中最重要的基础内容之一。有了 MATLAB 的符号工具箱，用户可以轻松地完成各种微积分计算。

3.6.1　符号极限

极限是微积分学的基础和出发点，因此，在介绍微积分之前先介绍极限的求解方法是必要的。在 MATLAB 中，极限的求解可由 limit 函数来实现。

limit　符号表达式的极限。其调用格式如下：

- limit(F, x, a)　计算符号表达式 F 在 x→a 条件下的极限值。
- limit(F, a)　计算符号表达式中由 findsym(F)返回的独立变量趋向于 a 的极限值。
- limit(F)　计算 a = 0 时的极限。
- limit(F, x, a, 'right')或 limit(F, x, a, 'left')　其中 right 或 left 用来指定取极限的方向。

【例如】

```
>> syms x a t h;
>> limit(sin(x)/x)
ans =
    1
>> limit((1+2*t/x)^(3*x),x,inf)
ans =
  exp(6*t)
>> limit(1/x,x,0,'right')
ans=
  inf
```

3.6.2　符号积分

1．积分函数 int

积分函数的调用格式如下：

- int(S)　计算符号表达式对 findsym 返回的符号自变量 S 的不定积分。S 为符号矩阵或符号数量。如果 S 为常数，则积分针对 x。
- int(S, v)　计算符号表达式 S 对符号自变量 v 的不定积分。v 是一数量符号量。
- int(S, a, b)　计算符号表达式 S 对默认符号变量从 a 到 b 的定积分。a 和 b 为双精度或符号数量。
- int(S, v, a, b)　计算符号表达式 S 对变量 v 从 a 到 b 的定积分。

【例如】

```
>> syms x x1 alpha u t;
>> A = [cos(x*t), sin(x*t);-sin(x*t), cos(x*t)];
>> int(A,t)
ans =
  [ sin(t*x)/x, -cos(t*x)/x]
  [ cos(t*x)/x,  sin(t*x)/x]
>> int(x1*log(1+x1),0,1)
ans =
    1/4
```

2．符号合计函数 symsum

符号合计函数的调用格式如下：

- symsum(S)　计算符号表达式对由 findsym 函数返回的符号变量的不定和。
- symsum(S, v)　计算符号表达式 S 对变量 v 的不定和。
- symsum(S, a, b)和 symsum(S, v, a, b)　计算从 a 到 b 的有限和。

【例如】

```
>> syms  k n
>> simple(symsum(k))
ans=
   k^2/2 - k/2
>> simple(symsum(k^2,0,n))
ans=
   n^3/3 + n^2/2 + n/6
>> symsum(k^2,0,10)
ans=
    385
```

3.6.3　符号微分和差分

1．微分和差分函数 diff

diff　微分和差分函数，包括数值差分和符号微分。其常用调用格式如下：

- diff(S)　对由 findsym 返回的自变量，求符号表达式 S 的微分。
- diff(S, 'v')或 diff(S, sym('v'))　对自变量 v，求符号表达式 S 的微分。

- diff(S, n)　对正整数 n，对符号表达式 S 微分 n 次。

diff(S, 'v', n)和 diff(S, n, 'v')这两种格式都可以被识别。

【例如】

```
>> x = sym('x');
>> t = sym('t');
>> diff(sin(x^2))
ans=
    2*x*cos(x^2)
>> diff(t^6,6)
ans=
    720
```

2．梯度函数 gradient

gradient　近似梯度函数。调用格式如下：

- [FX, FY] = gradient(F)　返回矩阵 F 的数值梯度，FX 相当于 dF/dx，为 x 方向的差分值。FY 相当于 dF/dy，为 y 方向的差分值。各个方向的点间隔设为 1。当 F 为向量时，DF = gradient(F)为一维梯度。
- [FX, FY] = gradient(F, H)　当 H 为数量时，用 H 作为各方向的点间隔。
- [FX, FY] = gradient(F, HX, HY)　当 F 为二维时，使用 HX 和 HY 指定点间距。HX 和 HY 可为数量和向量，如果 HX 和 HY 是向量，则它们的维数必须和 F 的维数一致。
- [FX, FY, FZ] = gradient(F)　返回三维的梯度。
- [FX, FY, FZ] = gradient(F, HX, HY, HZ)　使用 HX，HY 和 HZ 指定间距。

【例 3.7】 利用函数 gradient 绘制一个矢量图。

```
>> [x, y] = meshgrid(-2:.2:2,-2:.2:2);
>> z = x .* exp(-x.^2 - y.^2);
>> [px,py] =gradient(z,.2,.2);
>> contour(z)
>> hold on
>> quiver(px,py)
>> hold off
```

得到的图形如图 3.1 所示。

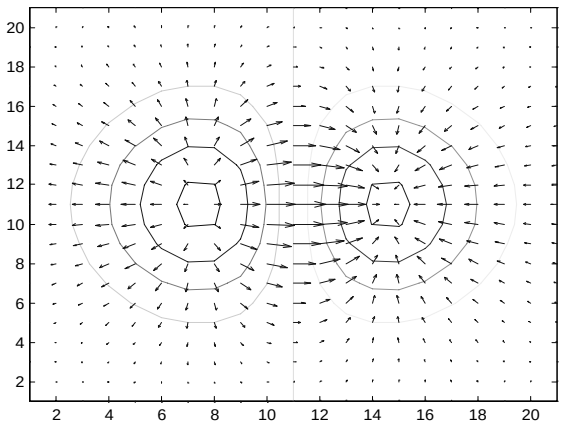

图 3.1　梯度函数绘制矢量图

3．多元函数的导数

在多元函数中，仿照单元函数的极限、可微的概念引入了 Frechet 导数。多元函数的 Frechet 导数在非线性方程的求解和变分原理中有极其重要的应用，在 MATLAB 中，此问题的实现由函数 jacobian 完成。

- jacobian(f, v)　计算数量或向量 f 对向量 v 的 Jacobi 矩阵。注意当 f 为数量时，函数返回 f 的梯度。

【例 3.8】 求各函数的 Jacobi 矩阵。

（1）$\begin{cases} x^2+y^2=4 \\ x^2-y^2=1 \end{cases}$

（2）$\begin{cases} 3x_1-\cos(x_1x_2)-0.5=0 \\ x_1^{\,2}-81(x_2+0.1)^2+\sin x_3+1.06=0 \\ \mathrm{e}^{-x_1x_2}+20x_3+(10\pi/3+1)=0 \end{cases}$

解：

（1）

```
>> x=sym(['x']);
>> y=sym(['y']);
>> z=sym(['z']);
>> jacobian([x^2+y^2;x^2-y^2], [x y])
ans =
    [  2*x, 2*y]
    [  2*x, -2*y]
```

（2）

```
>> f2='[3*x-cos(x*y)-0.5;x^2-81*(y+0.1)^2+sinz+1.06;exp(-x*y)+20*z+(10*pi/3-1)]';
>> jacobian(f2, [x y z])
ans =
    [ y*sin(x*y) + 3.0,      x*sin(x*y),  0]
    [           2*x, - 162.0*y - 16.2,  0]
    [     -y/exp(x*y),     -x/exp(x*y), 20]
```

另外，在求微函数中还有 del2 函数，用来离散拉普拉斯因子，此函数将在下面的求解方程组中介绍。

3.7　符号代数方程求解

3.7.1　线性方程组的符号解法

在符号数学工具箱中还提供了线性方程的符号求解函数，如 solve。此方法可得到方程组的精确解。所得的解析解可由函数 vpa 转换成浮点近似数值。

【例 3.9】 试对数值方程组 $\begin{cases} 10x-y=9 \\ -x+10y-2z=7 \\ -2y+10z=6 \end{cases}$ 用符号求解函数来求解。

```
>> [x,y,z]=solve('10*x-y=9','-x+10*y-2*z=7','-2*y+10*z=6')
x =
473/475
y =
91/95
z =
376/475
>> vpa([x,y,z])
ans =
[ 0.99578947368421052631578947368421, 0.95789473684210526315789473684211,
0.79157894736842105263157894736842]
```

3.7.2 非线性方程的符号解法

非线性方程的符号求解由函数 fsolve 实现。其调用格式如下：

- X=fsolve('fun', X0)　fun 为所要求解的函数名，通常以 M 文件的形式给出，返回函数值 F=fun(X)。X0 为求解方程的初始向量或矩阵。
- X=fsolve('fun', X0, options)　options 为选择参数输入向量，如 options(2）表示求解的精度要求；options(3)表示在解处的函数值精度要求；详细内容可查看 help foptions。
- X=fsolve('fun', X0, optionS, 'gradfun')　gradfun 为输入函数在 X 处的偏导数。
- X=fsolve('fun', X0, optionS, 'gradfun', P1, P2，…)　参数 P1，P2 等为问题定性参数，直接赋给函数 fun 和 gradfun，即 fun(X, P1, P2, …)和 gradfun(X, P1, P2，…)。此时若 options 和 gradfun 使用默认值，则要输入空矩阵。
- [X, options]=fsolve('fun', X0，…)　返回使用的优化方法的参数。例如，options(10) 表示函数估值的次数；默认算法为二次三次混合搜索的 Gauss-Newton 方法，若使用 Levenberg-Marquardt 算法，要设置 options(5)=1。

【例 3.10】 用 fsolve 函数求解下面的非线性方程。

$$\begin{cases} x_1 - 0.7\sin x_1 - 0.2\cos x_2 = 0 \\ x_2 - 0.7\cos x_1 + 0.2\sin x_2 = 0 \end{cases}$$

解：首先编制函数文件 fc.m 如下。

```
fc.m
function y=fc(x)
y(1)=x(1)-0.7*sin(x(1))-0.2*cos(x(2));
y(2)=x(2)-0.7*cos(x(1))+0.2*sin(x(2));
y=[y(1) y(2)];
```

在 MATLAB 命令窗口中输入：

```
>> x0=[0.5 0.5];
>> fsolve('fc',x0)
ans =
   0.5265   0.5079
```

可见计算极其简便。

3.8　符号微分方程求解

常微分方程的符号解由函数 dsolve 来计算，其常用调用格式如下：

- dsolve('equ1'，'equ2'，…）

以代表微分方程及初始条件的符号方程为输入参数，多个方程或初始条件可在一个输入变量内联立输入，且以逗号分隔。默认的独立变量为 t，也可把 t 变为其他的符号变量。字符 D 代表对独立变量的微分，通常指 d/dt。紧跟一数字的 D 代表高阶微分，如 D2 为 d^2/dt^2。紧跟此微分操作符的任何符号都可作为被微变量，如 D3y 代表对 y(t）的 3 阶微分。注意，用户所定义的符号变量不能再包括字符 D。初始条件可以由方程的形式给出（如 y(a)=b 或 Dy(a)=b）。这里 y 为被微变量而 a 和 b 为常数。如果初始条件的数目少于被微变量的数目，则结果中要包含不定常数 C1，C2 等。

此函数有三种可能的输出类型：

- 一个方程和一个输出，则结果返回一符号向量中非线性方程的联立解；
- 多个方程和多个输出，则结果以字母顺序排序且赋给输出量；
- 多个方程和单输出，则结果返回解的结构。

【例如】

```
>> dsolve('Dx = -a*x')
ans =
     C4/exp(a*t)
>> y = dsolve('Dy = 1/sqrt(y)', 'y(0) = 1')
 y =
     ((3*t)/2 + 1)^(2/3)
```

3.9　符号函数的二维图

3.9.1　符号函数的简易绘图函数 ezplot

函数的调用格式如下：

- ezplot(F)　绘制 f(x)的函数图，这里 f 为代表数学表达式的包含单个符号变量 x 的字符串或符号表达式。x 轴的近似范围为[−2*pi, 2*pi]。
- ezplot(f, xmin, xmax)或 ezplot(f, [xmin, xmax])　使用输入参数来代替默认横坐标范围[−2*pi，2*pi]。
- ezplot(f, [xmin xmax], fig)　指定绘图的图窗号以代替当前图窗。

【例 3.11】 绘出误差函数的图形。

```
ezplot('erf(x)')                    %或 ezplot erf(x)
```

这两种输入得到相同的函数图，如图 3.2 所示。

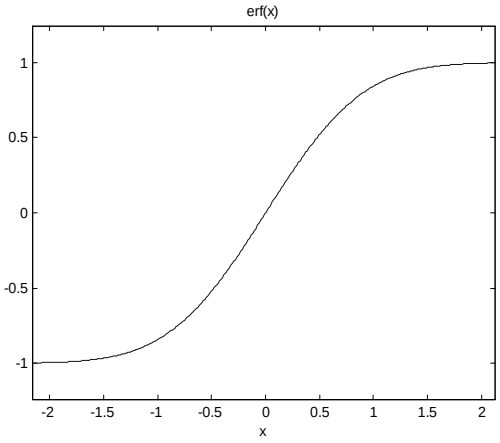

图 3.2　误差图

3.9.2　绘制函数图函数 fplot

函数的调用格式如下：

- fplot(fun, lims)　绘制由字符串 fun 指定函数名的函数在 x 轴区间为 lims=[xmin xmax]的函数图。若 lims=[xmin xmax ymin ymax]，则 y 轴也被输入限制。fun 必须为一个 M 文件的函数名或对变量 x 的可执行字符串，此字符串被送入函数 eval 后执行。函数 fun(x)必须要返回一针对向量 x 的每一元素的结果行向量。
- fplot(fun, limS, tol)　其中 tol<1 用来指定相对误差精度，默认值为 tol=0.002。
- fplot(fun, limS, n)　其中 n≥1，指定以最少 n+1 个点来绘制函数图，默认 n=1。最大步长被约束为不小于(1/n)×(xmax−xmin)。
- fplot(fun, limS, 'LineSpec')　以指定线型绘制图形。
- [x, y] = fplot(fun, limS, …)　只返回用来绘图的点的向量值，而不绘出图形。用户可自己用 plot(x, y)来输出图形。

【例如】

```
>> x = 0:.05:1;
>> fplot('[tan(x),sin(x),cos(x)]',2*pi* [-1 1 -1 1])
```

得到的图形如图 3.3 所示。

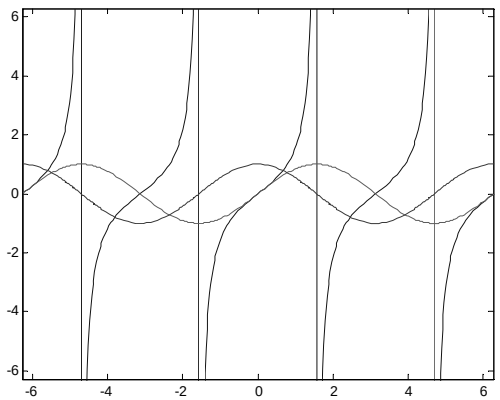

图 3.3　fplot 函数的绘图