```
[a0 , an , bn] = mlfourier(g , 4)
a0 = 0
an = 0
bn = -2 * cos(pi * n)/pi/n + 2/pi/n
[a0 , an , bn] = mlfourier(h , 4)
a0 = 2
an = 4 * (cos(pi * n) + pi * n * sin(pi * n))/pi^2/n^2 - 4/pi^2/n^2
bn = -4 * ( - sin(pi * n) + pi * n * cos(pi * n))/pi^2/n^2
[a0 , an] = mcos(f)
a0 = pi
an = (2 * (cos(pi * n) + pi * n * sin(pi * n))/n^2 - 2/n^2)/pi
[a0 , an] = mcos(g)
a0 = 2
an = 2 * sin(pi * n)/pi/n
[a0 , an] = mcos(h)
a0 = pi
an = (2 * (cos(pi * n) + pi * n * sin(pi * n))/n^2 - 2/n^2)/pi
bn = msin(f)
bn = -2 * ( - sin(pi * n) + pi * n * cos(pi * n))/n^2/pi
bn = msin(g)
bn = ( - 2 * cos(pi * n)/n + 2/n)/pi
bn = msin(h)
bn = -2 * ( - sin(pi * n) + pi * n * cos(pi * n))/n^2/pi
```

在 MathCAD 中也可定义类似函数. 请同学们先定义出这些函数,然后以 $f(t)=t, g(t)=t^2$ 和 $h(t)=e^t$ 为例在 MathCAD 中求出傅立叶级数的系数.

第六节　数值微积分

一、数值微分

在实际应用中,通常要根据已知的数据点,求某点处的一阶或高阶导数;或者求任意函数在任一点处的任意阶导数,这就要用到数值微分. 数值微分的基本思路是先用逼近或拟合等方法将已知数据在一定范围内满足的近似函数求出,再用特定方法对此近似函数求微分. 如果函数事先给出,则无需第一步. 通常可用多项式求导法和中心差分法求数值微分.

1. 多项式求导法求微分

若已知函数在某些节点处的值，只要用曲线拟合法得到一个函数的近似多项式(即用多项式近似代替函数)，再对该多项式求微分，然后对微分后的多项式求值，便可方便地求出在拟合范围内的任一点处的任意阶微分. 从理论上讲，多项式可求任意阶微分，因而可求出任一点处任意阶导数的近似值，但随着求导阶数的增加，计算误差就会越来越大，甚至可能出现求出的结果与真实值相差甚远的现象，且误差难于估计. 因此，一般用该方法仅限于求低阶数值微分.

在 MATLAB 中，用多项式求微分时会用到以下几个有关多项式的函数

```
polyfit(x, y, n)
```

利用数据点向量 x 和 y 拟合一个 n 次多项式，返回多项式系数向量，系数由高次方朝低次方顺序排列.

```
polyval(P, x0)
```

计算多项式 P 在点 $x0$ 处的值.

```
polyder(P)
```

返回多项式 P 求导后的系数向量.

根据上述几个函数，我们可以编写一个利用多项式方法求任何函数在任一点处的任意阶导数的近似值，函数使用格式为

```
fpder(f, x0, n, m, d, k)
```

其中 f 为欲求导函数的符号表达式；$x0$ 为固定点，即求函数在该点处的导数；n 表示求导阶数，默认为一阶导数；m 为多项式最高次数，表示用 m 次多项式近似模拟给定函数，默认值为 6；d 为半区间长度，即在$[x0-d, x0+d]$区间内拟合多项式，默认值为 1；k 为区间采样点数，即在闭区间$[x0-d, x0+d]$上采 $k+1$ 个点来拟合多项式，默认值为 20. 前两个参数是必需的，不能省略. 函数如下

```
function y = fpder(f, x0, n, m, d, k)
if nargin<6
   k = 20;
end
if nargin<5
   d = 1.0;
end
if nargin<4
   m = 6;
end
if nargin == 2
```

```
    n = 1;
  end
  x = x0 - d:2 * d/k:x0 + d;
  y = eval(f);
  p = polyfit(x, y, m);
  pp = p;
  for n1 = 1:n
     pp = polyder(pp);
  end
  y = polyval(pp, x0);
```

例如

```
fpder('sin(x)' pi/4, 3, 10, 0.5, 10)        ans = -0.7071
fpder('exp(-x)' 1)                          ans = -0.3679
fpder('exp(-x)' 3)                          ans = -0.0498
fpder('exp(-x)' 3, 2)                       ans = 0.0498
fpder('exp(-x.^2)' 3, 2)                    ans = 0.0042
fpder('exp(-x.^2)' 1, 5, 10, 0.5)           ans = 2.9393
```

也可以先定义函数再求导数,例如

```
f = 'exp(-x). * log(sin(x.^(1/3)/5))'
fpder(f, pi/4)                              ans = 0.6508 + 0.3184i
fpder(f, pi/4, 2)                           ans = -0.2391 - 2.8557i
f = 'exp(-x). * sin(x)'
fpder(f, pi, 2)                             ans = 0.0864
```

注意上述定义函数时用到的运算符是带点的,否则得不到结果.

2. 中心差分法求微分

如果已知函数,则用中心差分法可求该函数在任意点处的导数.如果函数未知,则可用多项式先拟合函数,再利用中心差分法求该函数在拟合区间内任一点处的导数.在 MATLAB 中可编写如下函数,用于求任意函数在任意点处的导数

```
function y = fcder(f, x0, h, tol, m)
if nargin<5
   m = 1000;
end
if nargin<4
   tol = 1.0e-4;
end
```

```
if nargin<3
   h=0.05;
end
x=x0;
ff=eval(f);
for n1=1:m
    x=x0+h;ff1=eval(f);
    x=x0-h;ff2=eval(f);
    gg=(ff1-ff2)/(2*h);
    if abs(gg-ff)<tol
       y=gg;
       return;
   end
   h=h*0.5;
   ff=gg;
end
y=gg;
```

该函数使用的一般格式为

$$\text{fcder}(f, x0, h, tol, m)$$

其中 f 为求导函数的符号表达式；$x0$ 为固定点，即求函数 f 在该点处的导数；这两个参数是必须输入的；h 为差分步长，一般取 0.01 到 0.1 为宜，默认为 0.05；tol 为允许误差，一般应在 10^{-4} 以下，默认为 10^{-4}；m 为最大循环次数，该项是为了避免因达不到误差要求陷入死循环而设置的. 例如

```
fcder('sin(x)',pi/4)                          ans=0.7071
fcder('sin(x)',pi/4,1,0.1)                    ans = 0.7059
fcder('sin(x)',pi/4,0.01,0.1)                 ans = 0.7071
fcder('sin(x)',pi/4,1,0.01)                   ans = 0.7053
fcder('sin(x)',pi/4,1,0.001)                  ans = 0.7070
fcder('sin(x)',pi/4,0.01,1e-8)                ans = 0.7071
fcder('sin(x)',pi/4,0.01,1e-2)                ans = 0.7071
fcder('exp(-x)',1,0.01,1e-2)                  ans = -0.3679
fcder('exp(-x)',1,0.1,1e-2)                   ans = -0.3680
fcder('exp(-x)',1,1,1e-2)                     ans = -0.3688
fcder('exp(-x)',1,1,1e-8)                     ans = -0.3679
fcder('exp(-x.^2)',1,1,1e-8)                  ans = -0.7358
```

```
fcder('exp(-x.^2)',3,1,1e-8)                    ans = -7.4046e-004
fcder('exp(-x.^2)',3,0.01,1e-8)                 ans = -7.4046e-004
fcder('exp(-x.^2)',1,0.01,1e-8)                 ans = -0.7358
fcder('exp(-x).*log(sin(x.^(1/3)/5))',pi/4)     ans = 0.9644
fcder('exp(-x).*sin(x)',pi)                     ans = -0.0432
```

比较上述运行结果可知，步长与容许误差对求导结果影响较大，若选取不当，将得不到正确的结果，特别是步长的选取不宜过大，因为步长对结果影响最大.另外，比较多项式求导法和中心差分求导法可知，当函数稍微复杂一些，特别是在奇异点附近，利用多项式求导法可能会得出错误结论，但用中心差分法却得到了正确结果，例如上例中，对函数

$$f(x)=\exp(x)\ln\left(\sin\left(\frac{\sqrt[3]{x}}{5}\right)\right)$$

求 $\pi/4$ 处的一阶导数值，用多项式法求出一个复数结果，显然是错误的，但用中心差分法得到了正确结果，该点处导数值应该是 0.964 412 176.但多项式法可求高阶导数，而且很方便，而中心差分法则比较麻烦.在 MathCAD 中编写数值微分函数较繁，这里略去不提.

二、数值积分

同数值微分比较，数值积分相对要简单得多.这是因为积分描述了一个函数的整体或宏观性质，而微分则描述一个函数在一点处的斜率，这是函数的微观性质.因而积分对函数的形状在小范围内的改变不敏感；而微分却很敏感，一个函数很小的变化，往往会导致相邻两点斜率有较大的改变.

在对已知函数求积分时，理论上可以利用 Newton-Leibniz 公式求解.但这种方法往往并不实用，因为实际中遇到的大多数函数是找不到原函数的，有些即使能找到，函数的表达式也十分复杂，故用 Newton- Leibniz 公式求解会变的很繁琐，给精确计算带来不便.对于工程上以数据表给出的表函数，只能利用经验函数或多项式将其拟合成解析函数，然后再求积分.数值积分就是对那些不能求积分或不易求积分的函数提供的一种较精确的积分方法.

在 MathCAD 中，如果被积函数是不可积的，则软件会自动利用数值方法求解.如函数 $\exp(-x^2)$ 是不可积函数，但在 MathCAD 中可求出积分如下

$$\int_{-\infty}^{0}\exp(-x^2)\,dx \rightarrow \frac{1}{2}\cdot\sqrt{\pi}=0.886$$

$$\int_{-1}^{1}\exp(-x^2)\,dx \rightarrow \sqrt{\pi}\cdot\mathrm{erf}(1)=1.494$$

$$\int_{0}^{1}\frac{1}{\sqrt{1+x^4}}dx=0.927$$

在 MATLAB 中,有两个函数是专门求数值积分的函数,它们分别是 quad( )函数和 quad8( )函数,其用法如下

```
quad('函数名串',a,b,tol,trace)
```

其中"函数名串"是 MATLAB 内建函数的函数名或用户自定义函数的函数名,如 sin(x)用 'sin',exp(x)函数用 'exp' 等;a 为积分下限;b 为积分上限;*tol* 为数值方法允许误差,省略时默认为 10^{-3};*trace* 为非零值时,以动态图形的形式实现积分的整个过程.quad8( )函数的用法与 quad( )函数的用法完全相同.但在功能上有所区别:quad( )函数用 Simpson 法求数值积分,而 quad8( )函数则用 Newton - Cotes 法求数值积分,前者属低阶法,后者则属高阶法.在处理有软奇点函数的积分时,quad8( )比 quad( )要好,例如用 quad( )函数有

```
quad('sin',0,pi)                    ans = 2.0000
quad('sin',0,pi,1e-6)               ans = 2.0000
quad('sin',0,pi,1e-6,1)             ans = 2.0000
quad('erf',0,1)                     ans = 0.4861
quad('erf',-inf,1)                  ans = -Inf
quad('log',0.1,1)                   ans = -0.6697
```

再定义两个函数 f1(x)和 f2(x)如下

```
function y = f1(x)
y = 1./(1 + x.^4);
function y = f2(x)
y = log(1 + tan(x));
```

则有

```
quad('f1',0,1)                      ans = 0.8670
quad('f2',0,pi/4)                   ans = 0.2722
```

同样,也可用 quad8( ) 函数求其结果

```
quad8('sin',0,pi)                   ans = 2.0000
quad8('sin',0,pi,1e-6)              ans = 2.0000
quad8('sin',0,pi,1e-6,1)            ans = 2.0000
quad8('erf',0,1)                    ans = 0.4861
quad8('erf',-inf,1)                 ans = NaN
quad('log',0.1,1)                   ans = -0.6697
quad8('f1',0,1)                     ans = 0.8670
quad8('f2',0,pi/4)                  ans = 0.2722
```

比较 quad8( )函数与 quad( )函数的执行结果可知,绝大多数基本一致,但有两点略有差异:一是观察由 quad('sin',0,pi,1e-6,1)和由 quad8('sin',0,pi,

le－6,1)两个函数给出的动态图形显示可知,前者求解过程要比后者慢得多,可见用高阶法要比用低阶法效率高,特别是当函数较复杂时更是如此;二是由quad('erf',－inf,1)得出的结果为－∞,而用quad8('erf',－inf,1)得出的结果为非数值NaN,即无法得出结果.

另外,我们还可以编写一些其它常用方法(如高斯十点法,抛物公式法和复合辛普生法等)的函数,这些函数都是基于被积函数已知的情况下而编写的.若被积函数为一些观测数据点,则需先拟合其近似分布函数,然后定义该函数,再利用下述方法求数值积分.

1. 高斯十点法函数

用Gauss十点法求数值积分的函数用法如下

```
gaussled(f,a,b)
```

其中:f 为被积函数函数名,是MATLAB内置函数名或用户自定义函数名;a 与 b 分别为积分下限和上限.函数如下

```
function y = gaussled(f,a,b)
n = 10;
z = [ - 0.9739065285, - 0.8650633677, - 0.6794095683, ...
    - 0.4333953941, - 0.1488743990, - 0.1488743990, ...
      0.4333953941, 0.6794095683, 0.8650633677, ...
      0.9739065285];
w = [0.0666713443, 0.1494513492, 0.2190863625, ...
    0.2692667193, 0.2955242247, 0.2955242247,...
      0.2692667193, 0.2190863625, 0.1494513492, ...
      0.0666713443];
gg = 0;
for i = 1:n
  yy = (z(i) * (b - a) + a + b)/2;
  gg = gg + w(i) * feval(f,yy);
end
y = gg * (b - a)/2;
```

例如

```
gaussled('sin',0,pi)          ans = 2.0000
gaussled('exp',0,pi)          ans = 21.0867
gaussled('log',0,1)           ans = -1.0386
gaussled('log',1,2)           ans = 0.3716
gaussled('f1',0,1)            ans = 0.8768
```

```
gaussled('f2',0,pi/4)          ans = 0.2609
```

2. 抛物公式法(Gauss 法)函数

用抛物公式法求数值积分函数用法如下

```
gauss(f,a,b,n)
```

其中:f 为被积函数名;a 为积分下限;b 为积分上限;n 为积分区间划分段数,必须为偶数.该积分分段越多,计算就越精确,但耗费机时也就越长.函数如下

```
function y = guass(f,a,b,n)
h = (b-a)/n;
y = 0;
for i = 0:(n/2-1)
  y = y + h*(feval(f,a+h*(1-1/sqrt(3)+2*i))...
      +feval(f,a+h*(1+1/sqrt(3)+2*i)));
end
```

例如

```
gauss('sin',0,pi,1000)         ans = 2.0000
gauss('sin',0,pi,100)          ans = 2.0000
gauss('sin',0,pi,10)           ans = 1.9999
gauss('cos',0,pi,1000)         ans = -1.1276e-016
gauss('exp',0,pi,1000)         ans = 22.1407
gauss('log',0,1,1000)          ans = -0.9998
gauss('log',1,2,1000)          ans = 0.3863
gauss('f1',0,1,1000)           ans = 0.8670
gauss('f2',0,pi/4,1000)        ans = 0.2722
```

3. 复合辛普生法函数

复合辛普生法求数值积分函数的用法如下

```
comsimpson(f,a,b,n)
```

其中 f,a,b,n 意义同抛物公式法,但这里 n 不必为偶数.函数如下

```
function y = comsimpson(f,a,b,n)
s = feval(f,a) - feval(f,b);
h = (b-a)/(2*n);x = a;
for i = 1:2:(2*n-1)
    x = x + h;
    s = s + 4*feval(f,x);
    x = x + h;
    s = s + 2*feval(f,x);
```

```
    end
    s = s * h/3.0;
    y = s;
```

例如

```
comsimpson('sin',0,pi,1000)          ans = 2.0000
comsimpson('sin',0,pi,100)           ans = 2.0000
comsimpson('sin',0,pi,10)            ans = 2.0000
comsimpson('sin',0,pi,1)             ans = 2.0944
comsimpson('cos',0,pi,1000)          ans = 6.8853e-014
comsimpson('exp',0,pi,1000)          ans = 22.1407
comsimpson('log',0,1,1000)           ans = -Inf
comsimpson('log',1,2,1000)           ans = 0.3863
comsimpson('f1',0,1,1000)            ans = 0.8670
comsimpson('f2',0,pi/4,1000)         ans = 0.2722
```

比较上述三种方法得出的结果，请同学们思考不同方法间的差异及同一种方法中区间分段数不同对精度的影响.

另外，MATLAB 还提供了一个用梯形法求数值积分的函数 trapz( )，该函数不对解析函数作数值积分，仅对数据表作数值积分，它有以下几种用法

```
trapz(Y)
```

当 Y 为一向量时，以单位等距节点计算数值积分，Y 的各分量为对应节点处的值；当 Y 为一矩阵时，以每一列作为一向量计算数值积分，得到的结果为一行向量，若节点间距不为 1 时，给计算结果乘以间距即可.

```
trapz(X,Y)
```

X 为节点；Y 为节点对应的值，即计算以 X 为节点，对应函数值为 Y 的数值积分；当 X 与 Y 为元素个数相同的向量时，求该向量的数值积分；当 X 为一向量，Y 为一矩阵时，X 的维数必须等于 Y 的行数，即计算 Y 的每一列对应于同一 X 的数值积分. 也可以在 trapz(Y)或 trapz(X,Y)后再加一参数 DIM，变为 trapz(Y,DIM)或 trapz(X,Y,DIM)，其中 DIM 为整数，表示矩阵的维，当 DIM 为 1 时按列计算，当 DIM 为 2 时按行计算. 例如

```
X =     1     3     7     9     10
Y =  1.2000    1.5000    1.9000    2.8000    4.5000
y =     1     2     3     4
        5     6     7     8
x =     1     3     7    10
z =     1     3
```

```
trapz(Y)          ans = 9.0500
trapz(X,Y)        ans = 17.8500
trapz(y)          ans =    3    4    5    6
trapz(y,1)        ans =    3    4    5    6
trapz(y,2)        ans = 7.5000
                        19.5000
trapz(x,y,2)      ans = 23.5000
                        59.5000
trapz(z,y)        ans =    6    8   10   12
```

三、微分方程数值解

一般情况下,除了简单的微分方程外,要找出微分方程解的解析表达式是极其困难的,有时甚至是不可能的.微分方程数值解法是能够计算出解在若干个离散点上近似结果的一种通用方法.

在 MATLAB 中提供了一些求解微分方程(组)的数值方法函数:ode45( ),ode23( ),ode113( ),ode15s( ) 和 ode23s( ).其中 ode23( )为低阶法解非刚性微分方程;ode23s( )为高阶法解刚性微分方程;ode45( )为中阶法解非刚性微分方程;ode113( )为变序法解非刚性微分方程;ode15s( )为变序法解刚性微分方程,其使用格式是类似的.以 ode23( )函数为例,其格式为

```
ode23('F',Tspan,Y0,OPTIONS)
```

其中"*F*"为函数名,即要将微分方程存入一个函数文件中,然后调用;*Tspan* 为一向量 $[t_0, t_m]$ 或 $[t_0, t_1, t_2, \cdots, t_n]$,前者是求解区间,后者是指定要求解的点,必须按升序或降序顺序排列;Y_0 为初值条件,即 $y|_{t=t_0} = y_0$;*OPTIONS* 可以指定一系列可选项包括允许绝对误差、允许相对误差、是否作图等,详细内容可参阅有关手册,当 *OPTIONS* 省略时,使用软件默认值.当微分方程为方程组时,F 与 Y_0 均为向量,F 是微分方程

$$\frac{dy}{dt} = F(t, y)$$

中的 $F(t,y)$ 函数,若微分方程为高阶方程,可将其化为一阶微分方程组.例如,求微分方程

$$\begin{cases} \dfrac{dy}{dt} = t^2 + \dfrac{y}{t}, \\ y|_{t=1} = \dfrac{1}{2} \end{cases}$$

在区间 [1,1.10] 的数值解;再求点 [1,1.02 ,1.06,1.08,1.10] 处的近似值.

编写下述函数,并以 df1 为文件名存盘:

```
function F = df1(t , y)
F = t^2 + y/t;
```

然后分别用函数 ode23( ),ode45( )和 ode113( )在区间 [1,1.1] 上求方程的数值解.用[T, Y]=函数(参数)可给出数值解,T 为自变量值,Y 为对应于 T 的因变量值.若直接使用函数,可画出数值解的图形(图 3-11、图 3-12 和图 3-13).

```
ode23(' df1 ',[1 , 1.1], 0.5)
[T , Y] = ode23(' df1 ',[1 , 1.1], 0.5)
T = 1.0000   1.0100   1.0200   1.0300   1.0400   1.0500   1.0600   1.0700
    1.0800   1.0900   1.1000
Y = 0.5000   0.5152   0.5306   0.5464   0.5624   0.5788   0.5955   0.6125
    0.6299   0.6475   0.6655
```

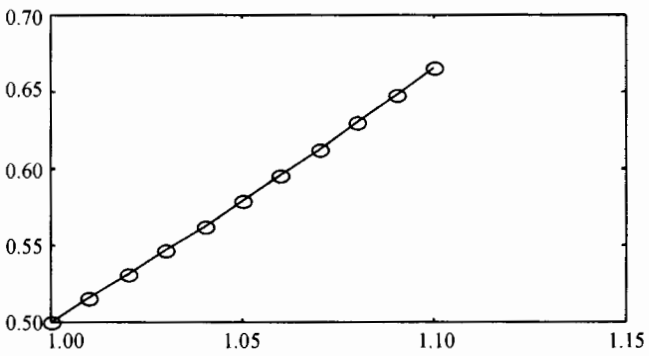

图 3-11　低阶法解微方程

```
ode45(' df1 ',[1 , 1.1], 0.5)
[T , Y] = ode45(' df1 ',[1 , 1.1], 0.5)
T = 1.0000   1.0025   1.0050   1.0075   1.0100   1.0125   1.0150   1.0175
    1.0200   1.0225   1.0250   1.0275   1.0300   1.0325   1.0350   1.0375
    1.0400   1.0425   1.0450   1.0475   1.0500   1.0525   1.0550   1.0575
    1.0600   1.0625   1.0650   1.0675   1.0700   1.0725   1.0750 1.0775
    1.0800   1.0825   1.0850   1.0875   1.0900   1.0925 1.0950   1.0975
    1.1000
Y = 0.5000   0.5038   0.5075   0.5113   0.5152   0.5190   0.5228   0.5267
    0.5306   0.5345   0.5384   0.5424   0.5464   0.5504   0.5544   0.5584
    0.5624   0.5665   0.5706   0.5747   0.5788   0.5830   0.5871   0.5913
```

```
0.5955   0.5997   0.6040   0.6082   0.6125   0.6168   0.6211   0.6255
0.6299   0.6342   0.6386   0.6431   0.6475   0.6520   0.6565   0.6610
0.6655
```

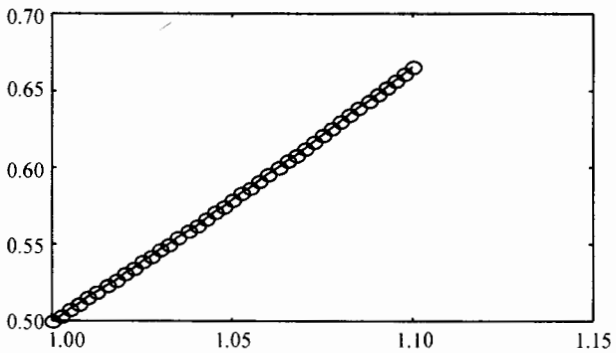

图 3-12 中阶法解微分方程

```
    ode113('df1',[1,1.1],0.5)
[T,Y]=ode113('df1',[1,1.1],0.5)
T =1.0000   1.0026   1.0079   1.0179   1.0279   1.0379   1.0479   1.0579
   1.0679   1.0779   1.0879   1.0979   1.1000
Y =0.5000   0.5040   0.5120   0.5273   0.5430   0.5590   0.5754   0.5920
   0.6089   0.6262   0.6438   0.6617   0.6655
[T,Y]=ode23('df1',[1,1.02,1.04,1.06,1.08,1.1],0.5)
T = 1.0000   1.0200   1.0400   1.0600   1.0800   1.1000
Y = 0.5000   0.5306   0.5624   0.5955   0.6299   0.6655
[T,Y]=ode45('df1',[1,1.02,1.04,1.06,1.08,1.1],0.5)
T = 1.0000   1.0200   1.0400   1.0600   1.0800   1.1000
Y = 0.5000   0.5306   0.5624   0.5955   0.6299   0.6655
```

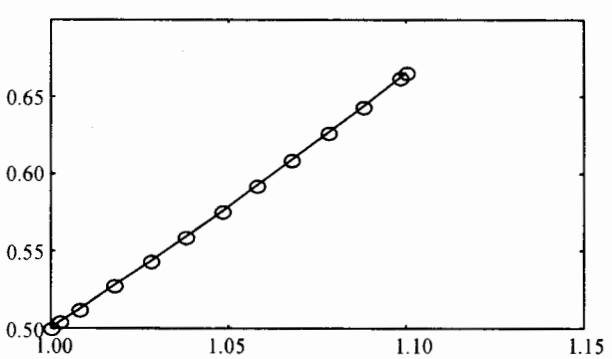

图 3-13 变序法解微分方程

由上述图形及结果可以看出三种不同方法的区别.在指定点处求出的函数值基本相同.

再分别求解如下四个微分方程在指定区间的函数值,先建立函数 df2,df3,df4 和 df5.

```
function F = df2(t,y)
F = y;
function F = df3(t,y)
F = (y-2*t)/(y+t^2);
function F = df4(t,y)
F = y+sin(t)/t;
function F = df5(t,y)
F = y-2*t/y;
```

则有

```
[T,Y] = ode23('df2',[0,2],1)
T = 0  0.0800  0.2800  0.4800  0.6800  0.8800  1.0800  1.2800 1.4800
    1.6800  1.8800  2.0000
Y = 1.0000  1.0833  1.3231  1.6159  1.9735  2.4103  2.9438  3.5954
    4.3912  5.3631  6.5501  7.3852
[T,Y] = ode23('df3',[1,2],1)
T = 1.0000  1.1000  1.2000  1.3000  1.4000  1.5000  1.6000  1.7000
    1.8000  1.9000  2.0000
Y = 1.0000  0.9458  0.8840  0.8157  0.7420  0.6637  0.5819  0.4973
    0.4107  0.3228  0.2341
[T,Y] = ode23('df4',[1,2],0.5)
T = 1.000  1.0298  1.1298  1.2298  1.3298  1.4298  1.5298  1.6298
    1.7298  1.8298  1.9298  2.000
Y = 0.500  0.5405  0.6832  0.8375  1.0043  1.1848  1.3802  1.5920
    1.8216  2.0711  2.3422  2.5467
[T,Y] = ode23('df5',[0,1],1)
T = 0  0.0800  0.1800  0.2800  0.3800  0.4800  0.5800  0.6800 0.7800
    0.8800  0.9800  1.0000
Y = 1.000  1.0770  1.1662  1.2490  1.3267  1.4000  1.4697  1.5363
    1.6001  1.6614  1.7206  1.7322
```

请读者自己用其他方法求解上述四个方程,并作图观察有何区别.

对于微分方程组和高阶微分方程的情形,请读者思考.

MathCAD 提供了两个用 Runge-Kutta 法求微分方程数值解的函数，它们分别为

rkfixed（初值，起始点，终止点，节点数，函数名）

Rkadapt（初值，起始点，终止点，节点数，函数名）

这两个函数的参数是相同的，前者用固定步长求解，后者用变步长求解，例如解微分方程

$$\begin{cases}\dfrac{dy}{dt}=\sin t+\exp(-t),\\ y(0)=0\end{cases}$$

在区间 [0,10] 上的数值解，节点数取 10，则有

$v_0 := 0$　　$x1 := 0$　　$x2 := 10$　　$n := 10$

$f(t,y) := \sin(t) + e^{-t}$　　$z := rkfixed(v,x1,x2,n,f)$

$$z^T = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & 1.092 & 2.282 & 2.941 & 2.636 & 1.71 & 1.038 & 1.246 & 2.146 & 2.912 & 2.84 \end{bmatrix}$$

其中结果 z^T 中的第 0 行（转置前的第 0 列）为节点值，第 1 行为对应于节点的函数值. 再如

$v_0 := 0.5$　　$x1 := 1$　　$x2 := 1.1$　　$n := 5$

$f(x,y) := x^2 + \dfrac{y}{x}$　　$z := rkfixed(v,x1,x2,n,f)$

$$z^T = \begin{bmatrix} 1 & 1.02 & 1.04 & 1.06 & 1.08 & 1.1 \\ 0.5 & 0.5306 & 0.5624 & 0.5955 & 0.6299 & 0.6655 \end{bmatrix}$$

亦可用变步长法求解

$z := Rkadapt(v,x1,x2,n,f)$

$$z^T = \begin{bmatrix} 1 & 1.02 & 1.04 & 1.06 & 1.08 & 1.1 \\ 0.5 & 0.5306 & 0.5624 & 0.5955 & 0.6299 & 0.6655 \end{bmatrix}$$

$v_0 := 1$　　$t1 := 1$　　$t2 := 1.7$　　$m := 7$

$f(x,y) := \dfrac{y-2x}{x_x^2}$　　$z := rkfixed(v,t1,t2,m,f)$

$$z^T = \begin{bmatrix} 1 & 1.1 & 1.2 & 1.3 & 1.4 & 1.5 & 1.6 & 1.7 \\ 1 & 0.8958 & 0.7856 & 0.6723 & 0.558 & 0.4439 & 0.331 & 0.2199 \end{bmatrix}$$

$z := Rkadapt(v,t1,t2,m,f)$

$$z^T = \begin{bmatrix} 1 & 1.1 & 1.2 & 1.3 & 1.4 & 1.5 & 1.6 & 1.7 \\ 1 & 0.8958 & 0.7856 & 0.6723 & 0.558 & 0.4439 & 0.331 & 0.2199 \end{bmatrix}$$

还可用求出的结果直接作图. 如

$v_0 := 0.5$　　$t1 := 1$　　$t2 := 2$　　$m := 6$

$f(x,y) := y + \frac{\sin(x)}{x}$　　$z := rkfixed(v,t1,t2,m,f)$

$$z^T = \begin{bmatrix} 1 & 1.1667 & 1.3333 & 1.5 & 1.6667 & 1.8333 & 2 \\ 0.5 & 0.7387 & 1.0104 & 1.3203 & 1.6745 & 2.0803 & 2.5467 \end{bmatrix}$$

$v_0 := 1$　　$x1 := 0$　　$x2 := 1$　　$m := 5$

$f(x,y) := y - 2x$　　$z := rkfixed(v,x1,x2,m,f)$

$$z^T = \begin{bmatrix} 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 1 & 1.1786 & 1.3082 & 1.3779 & 1.3745 & 1.2817 \end{bmatrix}$$

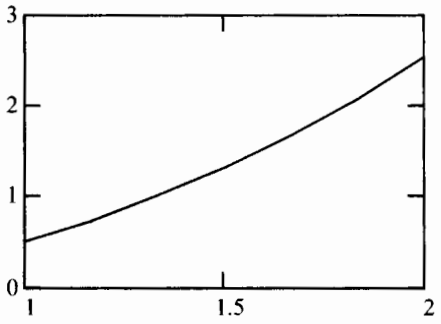

图 3－14　y ' = y + sinx/x 数值解

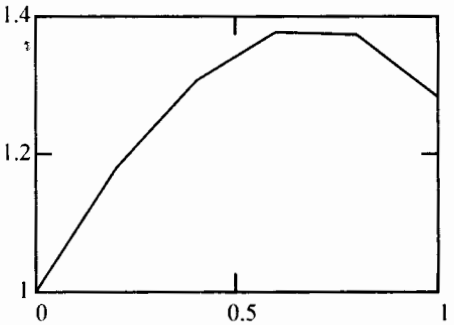

图 3－15　y ' = y－2x 数值解

MathCAD 中的 Runge-Kutta 法仅能求解线性微分方程. MathCAD 还提供了另外一个函数 Bulstoer( ),该函数与函数 Rkfixed( )的用法完全相同,但求解微分方程的方法是用 Bulirsch-Stoer 法. 对于刚性的微分方程,MathCAD 还提供了 stiffb( )和 stiffr( )函数解微分方程,前者用 Bulirsch-Stoer 方法,后者用 Rosenbrock 方法. 对于高阶微分方程,可将其化为一阶微分方程组,这时只需将前面的初值和函数改为向量形式即可求解. 例如要求解微分方程

$$\begin{cases} y'' - 2y' + 200y = 20x^2\exp(x), \\ y'(0) = 1,\ y(0) = 0, \end{cases}$$

令 $y_1 = y'$,则 $y'_1 - 2y_1 + 200y = 20x^2\exp(x)$,方程化为

$$\begin{cases} y'_0 = y_1, \\ y'_1 = 2y_1 - 200y_0 + 20x^2\exp(x), \\ y_1(0) = 1,\ y_0(0) = 0. \end{cases}$$

则有(图 3－16):

$$f(x,y):=\begin{bmatrix} y_1 \\ 2y_1 - 200y_0 + 20x^2\cdot\exp(x) \end{bmatrix} \qquad v:=\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$x1:=0 \qquad x2:=10 \qquad n:=1000$

$z:=\text{Bulstoer}(v,x1,x2,n,f)$

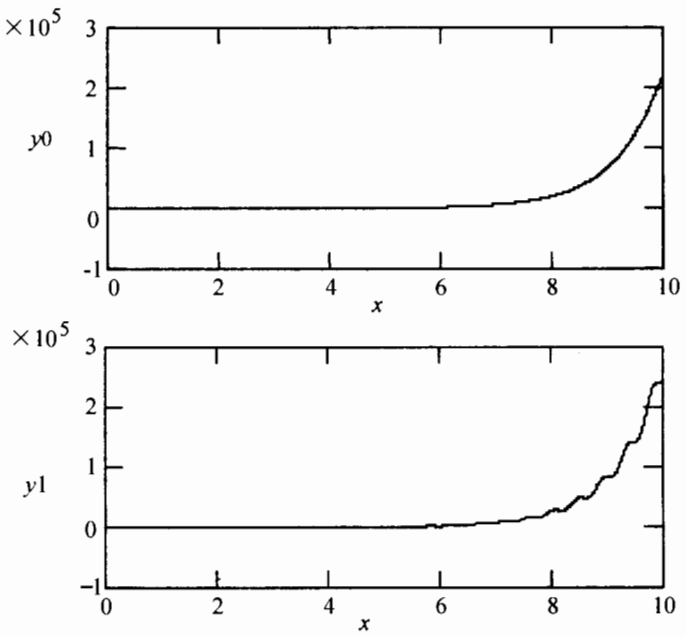

图 3－16 微分方程 $y''-2y'+200y=20x^2\exp(2)$的特解

第七节 微积分应用模型

一、血吸虫病的数学模型

要建立血吸虫病的数学模型，就必须对血吸虫的生活史有全面的了解.血吸虫在终宿主体内(如人、家畜、野生动物等)配对产生虫卵，虫卵通过循环系统或组织排出体外.如果虫卵落入水中便孵出毛蚴，毛蚴侵入适宜螺类宿主(钉螺)，在螺类宿主中发育成胞蚴，再由胞蚴发育成尾蚴.螺类于水中释放尾蚴，尾蚴再于水中入侵终宿主，进一步在终宿主体内成长、发育、交配、排卵，因而血吸虫病传播形成一回路系统(图 3－17).

Hairston 基于寄生虫的寿命表，提出了血吸虫的纯生殖率的估计，其数学表达式为