# 实验1.2 误差传播与算法稳定性

解：

（1）两种算法不同有效数字位数的结果见表1

表1 两种算法不同有效数字位数的结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5位有效数字 | | 6位有效数字 | | 7位有效数字 | |
| E.1.6 | E.1.7 | E.1.6 | E.1.7 | E.1.6 | E.1.7 |
| 3.678800e-01 | 3.678800e-01 | 3.678790e-01 | 3.678800e-01 | 3.678794e-01 | 3.678795e-01 |
| 2.642400e-01 | 2.642400e-01 | 2.642420e-01 | 2.642410e-01 | 2.642412e-01 | 2.642411e-01 |
| 2.072800e-01 | 2.072800e-01 | 2.072740e-01 | 2.072770e-01 | 2.072764e-01 | 2.072768e-01 |
| 1.708800e-01 | 1.708900e-01 | 1.709040e-01 | 1.708930e-01 | 1.708944e-01 | 1.708929e-01 |
| 1.456000e-01 | 1.455300e-01 | 1.454800e-01 | 1.455360e-01 | 1.455280e-01 | 1.455357e-01 |
| 1.264000e-01 | 1.267900e-01 | 1.271200e-01 | 1.267860e-01 | 1.268320e-01 | 1.267857e-01 |
| 1.152000e-01 | 1.125000e-01 | 1.101600e-01 | 1.125000e-01 | 1.121760e-01 | 1.125000e-01 |
| 7.840000e-02 | 1.000000e-01 | 1.187200e-01 | 1.000000e-01 | 1.025920e-01 | 1.000000e-01 |
| 2.944000e-01 | 1.000000e-01 | -6.848000e-02 | 1.000000e-01 | 7.667200e-02 | 1.000000e-01 |
| 1.944000e+00 | 0.000000e+00 | 1.684800e+00 | 0.000000e+00 | 2.332800e-01 | 0.000000e+00 |

根据表中不同有效位数对应结果的相互对比，我认为算法E.1.7结果更精确

（2）两种算法的优劣与我第一感觉吻合，以下是理论分析：

E.1.6误差：



E.1.7误差：



由以上两式可知，算法E.1.6的误差随着n的增大，逐渐增大，算法E.1.7的误差随着n的增大，逐渐减小，与实验结果相吻合。

（3）算法E.1.6中，n增大，增大，算法E.1.7中随着n减小而逐渐增大，所以算法E.1.6在某一步产生的误差对后面的影响是扩张的，而算法E.1.7在某一步产生的误差对后面的影响是衰减的。

（4）结合理论分析与计算实验，可得出算法E.1.7的稳定性优于算法E.1.6，而且算法E.1.6随着迭代步数n的增大，有发散的趋势。

本实验MATLAB程序如下：

function test1\_2

%数值实验1.2：误差传播与算法稳定性

%输入：递推公式选择与递推步数

%输出：各步递推数值及误差结果，以及递推值和误差与递推步数的关系图

promps={'请选择递推关系式，若选E.1.6，请输入1，否则输入2：'};

result=inputdlg(promps,'test1\_2',1,{'1'});

nb=str2num(char(result));

if((nb~=1)&&(nb~=2))

errordlg('请选择递推关系式，若选E.1.6，请输入1，否则输入2!');

return;

end

result=inputdlg({'请输入递推步数n:'},'test1\_2',1,{'10'});

steps=str2num(char(result));

if(steps<1)

errordlg('递推步数错误!');

return;

end

result=inputdlg({'请输入计算中所采用的有效数字位数:'},'test1\_2',1,{'5'});

sd=str2num(char(result));

format long

result=zeros(1,steps);

err=result;

func=result;

%用库函数quadl计算积分的近似值

for n=1:steps

fun=@(x)x.^n.\*exp(x-1);

func(n)=quadl(fun,0,1);

end

if(nb==1)

%用算法E.1.6计算

digits(sd);

result(1)=subs(vpa(1/exp(1)));

for n=2:steps

result(n)=subs(vpa(1-n\*result(n-1)));

end

err=abs(result-func);

elseif(nb==2)

%用算法E.1.7计算

digits(sd);

result(steps)=0;

for n=steps:-1:2

result(n-1)=subs(vpa((1-result(n))/n));

end

err=abs(result-func);

end

clf;

disp('递推值:');

fprintf('%e \n',result);

disp('误差:')

fprintf('%e \n',err);

plot([1:steps],result,'-');

grid on

hold on;

plot([1:steps],err,'r--');

xlabel('n');

ylabel('en-and err n--');

text(2,err(2),'\uparrow err(n)');

text(4,result(4),'\downarrow en');

# 实验3.1

解：

由程序计算结果得：

拟合曲线中的参数= {1.99911，-2.99767，-3.96825e-05，0.549119}，平方误差=2.17619e-05，离散数据的拟合函数的图形见图1。



图1 拟合函数

本实验MATLAB程序如下：

function test3\_1

x0=-1:0.5:2;

y0=[-4.447 -0.452 0.551 0.048 -0.447 0.549 4.552];

n=3;

alph=polyfit(x0,y0,n);

y=polyval(alph,x0);

r=(y0-y)\*(y0-y)';

x=-1:0.01:2;

y=polyval(alph,x);

plot(x,y,'k--');

xlabel('x');

ylabel('y');

hold on;

plot(x0,y0,'x');

title('Polynomial fitting of discrete data');

grid on;

disp(['Square error:',sprintf('%g',r)]);

disp(['parameter alph:',sprintf('%g\t',alph)])

# 实验5.1常微分方程性态和R-K法稳定性试验

解：

（1）考虑到图中显示问题，本实验取=2，0.5，-0.6，-3，四组计算结果见图2。



图2 四组值的计算结果

由图可知，4组值的结果稳定，由于取值较小，使得这四组位于稳定域内。

（2）经典R-K法的稳定域为：，取=-14，取=0.1与=0.2使得参数分别位于经典R-K法的稳定域内与稳定域外。两组图像见图3。取痊愈等距的10个点上的计算值，见表2。



图3两组值的计算结果

表2两组值的计算结果

|  |  |  |  |
| --- | --- | --- | --- |
| xn | h=0.1的解 | h=0.2的解 | 解析解 |
| 0.1 | 0.382733 |  | 0.346597 |
| 0.2 | 0.279938 | 1.222400 | 0.260810 |
| 0.3 | 0.322601 |  | 0.314996 |
| 0.4 | 0.406390 | 1.445302 | 0.403698 |
| 0.5 | 0.501807 |  | 0.500912 |
| 0.6 | 0.600511 | 1.668717 | 0.600225 |
| 0.7 | 0.700144 |  | 0.700055 |
| 0.8 | 0.800041 | 1.892656 | 0.800014 |
| 0.9 | 0.900012 |  | 0.900003 |

由表中结果可知，当h=0.1时，位于稳定域内，与解析结吻合较好，当h=0.2时，位于稳定域外，与解析解相差较大。

本实验MATLAB程序如下：

function test5\_1

%数值实验5\_1：常微分方程性态和R-K法稳定性实验

%输入：参数a，步长h

%输出：精确解和数值解图形对比

clf;

for i=1:4

result=inputdlg({'请输入[-50 50]间的参数a:'},'test5\_1',1,{'-40'});

a=str2num(char(result));

if((a<-50)||(a>50))

errordlg('请输入正确的参数a!');

return;

end

result=inputdlg({'请输入(0 1)之间的步长:'},'test5\_1',1,{'0.01'});

h=str2num(char(result));

if((h>=1)||(h<=0))

errordlg('请输入正确的(0 1)之间的步长!');

return;

end

x=0:h:1;

y=x;

N=length(x);

y(1)=1;

func=inline('1+(y-x).\*a');

for n=1:N-1

k1=func(a,x(n),y(n));

k2=func(a,x(n)+h/2,y(n)+k1\*h/2);

k3=func(a,x(n)+h/2,y(n)+k2\*h/2);

k4=func(a,x(n)+h,y(n)+k3\*h);

y(n+1)=y(n)+h\*(k1+2\*k2+2\*k3+k4)/6;

end

fprintf('%e ',y);

y0=exp(a\*x)+x;

plot(x,y0,'g+');

hold on;

plot(x,y,'b--');

xlabel('x');

ylabel('y');

end