实验2 误差传播

参考答案

1. 采用long e格式计算2.6连续加上三个0.2、2.6连续加上四个0.6的结果，判断计算结果正确与否并解释原因。

|  |  |
| --- | --- |
| >> format long e | >> 2.6+0.6  ans = 3.200000000000000e+00 |
| >> 2.6+0.2  ans = 2.800000000000000e+00 | >> ans+0.6  ans = 3.800000000000000e+00 |
| >> ans+0.2  ans = 3.000000000000000e+00 | >> ans+0.6  ans = 4.400000000000000e+00 |
| >> ans+0.2  ans = 3.200000000000001e+00 | >> ans+0.6  ans = 5 |

对精确的有理小数0.2的反复加运算产生了错误的数据位数，导致结果错误；对0.6的反复加运算不仅没有丢失精度，而且最后结果是完全正确的整数5.这说明在MATLAB中数值计算采用的是有限精度（尾数和阶码位数有限）的存储，由数字的浮点数形式的存储引起算术运算中的舍入误差，舍入的位数是由机器精度决定的。

1. 对下列3个差分方程计算出前10个数值近似值。在每种情况下引入一个小的初始误差。如果没有初始误差，则每个差分方程将生成序列。编程计算每个序列，构造如下两张表，绘制每个序列的误差序列图形。从中能得到什么结论？
2. ，其中*n* =1,2,…
3. ，其中*n* =2,3,…
4. ，其中*n* =2,3,…

构造表1和表2：

表1 序列以及近似值{*rn*},{*pn*}和{*qn*}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *n* | *xn* | *rn* | *pn* | *qn* |
| 0 | 1.000000000000000 | 0.994000000000000 | 1.000000000000000 | 1.000000000000000 |
| 1 | 0.500000000000000 | 0.497000000000000 | 0.497000000000000 | 0.497000000000000 |
| 2 | 0.250000000000000 | 0.248500000000000 | 0.245500000000000 | 0.242500000000000 |
| 3 | 0.125000000000000 | 0.124250000000000 | 0.119750000000000 | 0.109250000000000 |
| 4 | 0.062500000000000 | 0.062125000000000 | 0.056875000000000 | 0.030625000000000 |
| 5 | 0.031250000000000 | 0.031062500000000 | 0.025437500000000 | -0.032687500000001 |
| 6 | 0.015625000000000 | 0.015531250000000 | 0.009718750000000 | -0.112343750000001 |
| 7 | 0.007812500000000 | 0.007765625000000 | 0.001859375000000 | -0.248171875000003 |
| 8 | 0.003906250000000 | 0.003882812500000 | -0.002070312500000 | -0.508085937500005 |
| 9 | 0.001953125000000 | 0.001941406250000 | -0.004035156250000 | -1.022042968750010 |
| 10 | 0.000976562500000 | 0.000970703125000 | -0.005017578125000 | -2.047021484375021 |

表2 误差序列{*xn*-*rn*},{ *xn*-*pn*}和{ *xn*-*qn*}

|  |  |  |  |
| --- | --- | --- | --- |
| *n* | *xn*-*rn* | *xn*-*pn* | *xn*-*qn* |
| 0 | 0.0060000000 | 0.0000000000 | 0.0000000000 |
| 1 | 0.003000000000000 | 0.003000000000000 | 0.003000000000000 |
| 2 | 0.001500000000000 | 0.004500000000000 | 0.007500000000000 |
| 3 | 0.000750000000000 | 0.005250000000000 | 0.015750000000000 |
| 4 | 0.000375000000000 | 0.005625000000000 | 0.031875000000000 |
| 5 | 0.000187500000000 | 0.005812500000000 | 0.063937500000001 |
| 6 | 0.000093750000000 | 0.005906250000000 | 0.127968750000001 |
| 7 | 0.000046875000000 | 0.005953125000000 | 0.255984375000003 |
| 8 | 0.000023437500000 | 0.005976562500000 | 0.511992187500005 |
| 9 | 0.000011718750000 | 0.005988281250000 | 1.023996093750010 |
| 10 | 0.000005859375000 | 0.005994140625000 | 2.047998046875021 |

绘制：图1 误差序列{*xn*-*rn*}；图2 误差序列{*xn*-*pn*}；图3 误差序列{*xn*-*qn*}。



程序代码：

% The propagation of errors

% Three ways to compute the sequence {1/2^n}

% The original errors are 0.006,0.003,0.003

x=zeros(11,1);r=x;p=x;q=x;

x(1)=1;r(1)=0.994;p(1)=1;p(2)=0.497;q(1)=1;q(2)=0.497;

for i=2:11

x(i)=1/2\*x(i-1);

r(i)=1/2\*r(i-1);

end

for i=3:11

p(i)=3/2\*p(i-1)-1/2\*p(i-2);

q(i)=5/2\*q(i-1)-q(i-2);

end

disp('x r p q')

disp([x r p q])

e1=abs(x-r); e2=abs(x-p); e3=abs(x-q);

disp('e1 e2 e3')

disp([e1 e2 e3])

subplot(2,2,1),plot(0:10,e1,'d')

subplot(2,2,2),plot(0:10,e2,'+')

subplot(2,2,3),plot(0:10,e3,'\*')

从上面的计算可以看到，初始误差通常通过一系列的计算进行传播。对于任何数值计算而言，都要尽量减少初始误差，因为初始条件下的小误差对最终结果产生的影响较小。另外，不同的计算方法导致的误差也不同，要选择合适的方法进行计算。

1. 设，则一元二次方程有两种等价的求根公式如下：
2. 
3. 

当时，上述两组公式总会存在“相近数相减”的运算，从而导致精度损失。所以在这种情况下应该重新构造合适的求根公式。利用上述两组公式，构造出求解*x*1和*x*2的适当公式，设计算法，编写MATLAB程序，并计算下列二次方程的根。

1. *x*2-1000.001*x*+1=0; b) *x*2-1000000.000001*x*+1=0

用公式i)、公式ii)、自编程序和Matlab自带的求多项式的根的函数roots求解上述方程的根，比较结果。

为防止出现“相近数相减”的运算，关键在于正确处理*b*和之间的运算，只要让它们同号（相加）就可以了。所以重新构造如下求根公式：



程序代码：

function x=unaquadeq(a,b,c)

% Solve a Unary quadratic equation

% Input - a, b, c are the coefficients of the equation by descending order

% Output - x is a vector involving the roots

if (a==0)|(b^2-4\*a\*c<=0), return, end

x1=(-b-sign(b)\*sqrt(b^2-4\*a\*c))/(2\*a);

x2=c/(a\*x1);

x=[x1,x2];

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 公式i) | 公式ii) | 自编函数 | roots函数 |
| a） | x =1.0e+03 \*  1.500000500000000  0.500001500000000 | x =1.0e+03 \*  1.000000000023647  0.000001000000000 | x =1.0e+03 \*  1.000000000000000  0.000001000000000 | ans =1.0e+03 \*  1.000000000000000  0.000001000000000 |
| b） | x =1.0e+06 \*  1.500000000000500  0.500000000001500 | x =1.0e+06 \*  0.999992385564610  0.000000000001000 | x =1.0e+06 \*  1.000000000000000  0.000000000001000 | ans =1.0e+06 \*  1.000000000000000  0.000000000001000 |

可见，重新构造的函数求根结果与roots函数求得的结果完全一样，而公式i)和公式ii)的误差比较大。