数值微分的数值计算方法实验

姓名：王旋宇
学号：2012049010022
班级：数理基科班
老师：赖生建

2014-11-18

1 实验目的

1. 了解精度为$O(h^2)$和$O(h^4)$的中心差分公式。并会优化步长。会使用理查森外推得到需要精度的数据。并编程实现

2. 掌握牛顿多项式微分，并编程实现基于N+1个点的差分求解。

2 实验原理

2 .1 导数的近似值

如果函数$f(x)$在点x的左边和右边的值可以计算，则最佳二点公式包含x 两边的两个对称的横坐标

定理1 (精度为$O(h^2)$的差分公式). 设$f \in C^3[a,b]$，且$x-h, x, x+h \in [a,b]$，则

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} \tag{1}$$

而且存在数$c = c(x) \in [a,b]$，满足

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} + E_{trunc}(f,h) \tag{2}$$

其中

$$E_{trunc}(f,h) = -\frac{h^2 f^{(3)}(c)}{6} = O(h^2) \tag{3}$$

项$E(f,h)$称为截断误差。

定理2 (精度为$O(h^4)$的差分公式). 设$f \in C^5[a,b]$，且$x-2h, x-h, x, x+h, x+2h \in [a,b]$，则

$$f'(x) \approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} \tag{4}$$

而且存在数$c = c(x) \in [a,b]$，满足

$$f'(x) \approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} + E_{trunc}(f,h) \tag{5}$$

其中

$$E_{trunc}(f,h) = -\frac{h^4 f^{(5)}(c)}{30} = O(h^4) \tag{6}$$

项$E(f,h)$称为截断误差。

定理3 . *[理查森外推]* 设$f'(x)$的两个精度为$O(h^{2k})$ 的近似值分别为$D_{k-1}(h)$和$D_{k-1}(2h)$，而且他们满足

$$f'(x_0) = D_{k-1}(h) + c_1h^{2k} + c_2h^{2k+2} + \cdots \tag{7}$$

和

$$f'(x_0) = D_{k-1}(2h) + 4^kc_1h^{2k} + 4^{k+1}c_2h^{2k+2} + \cdots \tag{8}$$

这样可以得到近似值的表达式

$$f'(x_0) = D_k(h) + O(h^{2k+2}) = \frac{4^kD_{k-1}(h) - D_{k-1}(2h)}{4^k - 1} + O(h^{2k+2}) \tag{9}$$

2 .2 牛顿多项式微分

根据点$t_0, t_1, \cdots, t_N$近似$f(t)$ 的N 次牛顿多项式P(t) 表示为

$$\begin{aligned} P(t) = a_0 &+ a_1(t - t_0) + a_2(t - t_0)(t - t_1) \\ &+ a_3(t - t_0)(t - t_1)(t - t_2) + \cdots + a_N(t - t_0)\cdots(t - t_{N-1}) \end{aligned} \tag{10}$$

P(t)的导数为

$$\begin{aligned} P'(t) =& a_1 + a_2((t - t_0) + (t - t_1)) \\ &+ a_3((t - t_0)(t - t_1) + (t - t_0)(t - t_2) + (t - t_1)(t - t_2)) \\ &+ \cdots + a_N \sum_{k=0}^{N-1} \prod_{j=0, j\neq k}^{N-1} (t - t_j) \end{aligned} \tag{11}$$

当在$t = t_0$处计算$P'(t)$时，式中有许多项为零，这样$P'(t_0)$ 可简化为

$$\begin{aligned} P'(t_0) =& a_1 + a_2(t_0 - t_1) + a_3(t_0 - t_1)(t_0 - t_2) + \cdots \\ &+ a_N(t_0 - t_1)(t_0 - t_2)(t_0 - t_3)\cdots(t_0 - t_{N-1}) \end{aligned} \tag{12}$$

3 实验内容

3 .1 P260

1. 使用极限的微分求解下列函数在x 处的导数近似值，精度为小数点后13位。

   (a) $f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$

   (b) $f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$

   (c) $f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$

(d) $f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$

(e) $f(x) = x^{x^x}; x = 0.0001$

2. 使用极限的微分求解，实现精度为$O(h^4)$ 的中心差分公式。用这个程序求解上题中给出的函数导数的近似值。精度为小数点后13位。

3. 利用外推法的微分求解第1题中函数导数的近似值。精度为小数点后13位。

3 .2 P270 1

利用牛顿多项式微分计算基于N+1个点的差分求解。

4 实验分析

4 .1 P260 1

由于误差分析中知道只有当步长取得一定值得时候误差才会最小，而在步长更长或者更短的时候误差都不是最小。所以程序设计中在每一次循环中步长缩小为之前的十分之一，但是需要判断本次循环的误差是否比上一次循环的误差大，是的话就要终止循环。同时当误差小于容差的时候也要停止循环。

4 .2 P260 3

由理查森外推的定理（(3)）中的(9) 可以得知在计算的时候利用递推的方法计算$D_k(h)$更加简洁。同时要求精度为13 位小数，由于步长小于1，所以设置k的值为6，这样$2k + 2 = 14 > 13$ 满足了精度要求。

4 .3 P268 1

在构造牛顿多项式求解导数近似值的时候，由式(12) 可知只要有了牛顿多项式的系数就可以了。由于牛顿多项式本身在构造的时候就不在意点的顺序，所以通过重新排列点的顺序可以得到同样的牛顿多项式。在计算第k 个点的导数值得时候就交换第一个和第k 个点的值，再次利用式(12) 计算就可以了。

5 实验结果

5 .1 P260 1

1. 对于第一个函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$，经过尝试采取初始步长为0.9 计算数据如下表

步长h	近似值	误差
0.9000000000	1404388653.0079610000000	1404388577.8344665000000
0.4500000000	379.8965386292859	304.7230439341123
0.4500000000	115.0463962209679	39.8729015257943
0.1125000000	85.0398645142786	9.8663698191050
0.1125000000	77.6331924026476	2.4596977074740
0.0281250000	75.7879853704329	0.6144906752592
0.0281250000	75.3270902221802	0.1535955270066
0.0070312500	75.2118918800969	0.0383971849233
0.0070312500	75.1830938853466	0.0095991901730
0.0017578125	75.1758944860871	0.0023997909135
0.0017578125	75.1740946424869	0.0005999473133
0.0004394531	75.1736446819909	0.0001499868173
0.0004394531	75.1735321918951	0.0000374967215
0.0001098633	75.1735040693487	0.0000093741751
0.0001098633	75.1734970387479	0.0000023435743
0.0000274658	75.1734952810542	0.0000005858806
0.0000274658	75.1734948418006	0.0000001466270
0.0000068665	75.1734947318255	0.0000000366519
0.0000068665	75.1734947043800	0.0000000092063
0.0000017166	75.1734947001346	0.0000000049610
0.0000017166	75.1734947001468	0.0000000049732

Table 1: 函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$计算结果

2. 对于第二个函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$，经过尝试采取初始步长为0.7 计算数据如下表

步长h	近似值	误差
0.7000000000	1.1655567992822	0.0630406240940
0.3500000000	1.2305168654049	0.0019194420288
0.3500000000	1.2300092018358	0.0014117784597
0.0875000000	1.2290030605441	0.0004056371680
0.0875000000	1.2287020319188	0.0001046085427
0.0218750000	1.2286237739733	0.0000263505972
0.0218750000	1.2286040234059	0.0000066000298
0.0054687500	1.2285990741569	0.0000016507808
0.0054687500	1.2285978361196	0.0000004127435
0.0013671875	1.2285975265651	0.0000001031890
0.0013671875	1.2285974491736	0.0000000257975
0.0003417969	1.2285974298251	0.0000000064490
0.0003417969	1.2285974249878	0.0000000016117
0.0000854492	1.2285974237804	0.0000000004042
0.0000854492	1.2285974234767	0.0000000001005
0.0000213623	1.2285974233958	0.0000000000197
0.0000213623	1.2285974233769	0.0000000000008
0.0000053406	1.2285974233990	0.0000000000229
0.0000053406	1.2285974234172	0.0000000000411

Table 2: 函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$计算结果

3. 对于第三个函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$，经过尝试采取初始步长为0.6 计算数据如下表

步长h	近似值	误差
0.6000000000	1.2497980870510	0.7017615218859
0.3000000000	2.0311024993890	0.0795428904522
0.3000000000	2.0120441269963	0.0604845180594
0.0750000000	1.9687028453655	0.0171432364286
0.0750000000	1.9559614488596	0.0044018399227
0.0187500000	1.9526671560056	0.0011075470687
0.0187500000	1.9518369360644	0.0002773271275
0.0046875000	1.9516289682009	0.0000693592640
0.0046875000	1.9515769504699	0.0000173415330
0.0011718750	1.9515639444274	0.0000043354906
0.0011718750	1.9515606928162	0.0000010838793
0.0002929687	1.9515598799069	0.0000002709701
0.0002929687	1.9515596766791	0.0000000677422
0.0000732422	1.9515596258740	0.0000000169371
0.0000732422	1.9515596131737	0.0000000042368
0.0000183105	1.9515596099958	0.0000000010589
0.0000183105	1.9515596091893	0.0000000002525
0.0000045776	1.9515596090332	0.0000000000963
0.0000045776	1.9515596089514	0.0000000000145
0.0000011444	1.9515596088725	0.0000000000643
0.0000011444	1.9515596088847	0.0000000000522
0.0000002861	1.9515596092485	0.0000000003116
0.0000002861	1.9515596092485	0.0000000003116

Table 3: 函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$计算结果

4. 对于第四个函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.1000000000	2.7381460011626	0.2273688280228
0.0500000000	2.9751474064600	0.0096325772746
0.0500000000	2.9727368672288	0.0072220380434
0.0125000000	2.9676320973711	0.0021172681858
0.0125000000	2.9660638036576	0.0005489744722
0.0031250000	2.9656533040973	0.0001384749119
0.0031250000	2.9655495249115	0.0000346957262
0.0007812500	2.9655235079298	0.0000086787444
0.0007812500	2.9655169991723	0.0000021699869
0.0001953125	2.9655153717016	0.0000005425162
0.0001953125	2.9655149648164	0.0000001356310
0.0000488281	2.9655148630900	0.0000000339046
0.0000488281	2.9655148376605	0.0000000084751
0.0000122070	2.9655148313168	0.0000000021314
0.0000122070	2.9655148297206	0.0000000005353
0.0000030518	2.9655148292841	0.0000000000987
0.0000030518	2.9655148291931	0.0000000000078
0.0000007629	2.9655148293386	0.0000000001533
0.0000007629	2.9655148294114	0.0000000002260

Table 4: 函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$计算结果

5. 对于第五个函数$f(x) = x^{x^x}; x = 0.0001$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.0000500000	1.0150760914859	0.0001300144143
0.0000250000	1.0151743931808	0.0000317127194
0.0000250000	1.0151982234375	0.0000078824627
0.0000062500	1.0152041380903	0.0000019678100
0.0000062500	1.0152056141223	0.0000004917779
0.0000015625	1.0152059829667	0.0000001229336
0.0000015625	1.0152060751675	0.0000000307327
0.0000003906	1.0152060982171	0.0000000076832
0.0000003906	1.0152061039794	0.0000000019208
0.0000000977	1.0152061054204	0.0000000004799
0.0000000977	1.0152061057804	0.0000000001198
0.0000000244	1.0152061058717	0.0000000000285
0.0000000244	1.0152061058946	0.0000000000056
0.0000000061	1.0152061059025	0.0000000000022
0.0000000061	1.0152061058985	0.0000000000017
0.0000000015	1.0152061059108	0.0000000000105
0.0000000015	1.0152061058800	0.0000000000202

Table 5: 函数$f(x) = x^{x^x}; x = 0.0001$计算结果

5 .2 P260 2

1. 对于第一个函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$，经过尝试采取初始步长为0.9 计算数据如下表

步长h	近似值	误差
0.9000000000	1404388653.0079610000000	1404388577.8344665000000
0.4500000000	-468129044.4739356000000	468129119.6474303000000
0.4500000000	26.7630154181952	48.4104792769784
0.1125000000	75.0376872787155	0.1358074164581
0.1125000000	75.1643016987707	0.0091929964029
0.0281250000	75.1729163596946	0.0005783354791
0.0281250000	75.1734585060962	0.0000361890774
0.0070312500	75.1734924327360	0.0000022624376
0.0070312500	75.1734945537639	0.0000001414097
0.0017578125	75.1734946863359	0.0000000088377
0.0017578125	75.1734946946174	0.0000000005563
0.0004394531	75.1734946951543	0.0000000000193
0.0004394531	75.1734946951966	0.0000000000230
0.0001098633	75.1734946951345	0.0000000000391
0.0001098633	75.1734946951453	0.0000000000283
0.0000274658	75.1734946952600	0.0000000000864
0.0000274658	75.1734946956751	0.0000000005015

Table 6: 函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$计算结果

2. 对于第二个函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$，经过尝试采取初始步长为0.7 计算数据如下表

步长h	近似值	误差
0.7000000000	1.1655567992822	0.0630406240940
0.3500000000	1.2521702207792	0.0235727974030
0.3500000000	1.2298399806461	0.0012425572700
0.0875000000	1.2286676801136	0.0000702567374
0.0875000000	1.2286016890437	0.0000042656676
0.0218750000	1.2285976879915	0.0000002646153
0.0218750000	1.2285974398834	0.0000000165073
0.0054687500	1.2285974244073	0.0000000010312
0.0054687500	1.2285974234405	0.0000000000644
0.0013671875	1.2285974233803	0.0000000000042
0.0013671875	1.2285974233764	0.0000000000003
0.0003417969	1.2285974233757	0.0000000000005
0.0003417969	1.2285974233753	0.0000000000008
0.0000854492	1.2285974233779	0.0000000000018
0.0000854492	1.2285974233755	0.0000000000006
0.0000213623	1.2285974233688	0.0000000000073
0.0000213623	1.2285974233707	0.0000000000055
0.0000053406	1.2285974234060	0.0000000000298
0.0000053406	1.2285974234233	0.0000000000472
0.0000013351	1.2285974232492	0.0000000001269
0.0000013351	1.2285974232821	0.0000000000940
0.0000003338	1.2285974239127	0.0000000005366
0.0000003338	1.2285974233167	0.0000000000594
0.0000000834	1.2285974233722	0.0000000000039
0.0000000834	1.2285974219308	0.0000000014453

Table 7: 函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$计算结果

3. 对于第三个函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$，经过尝试采取初始步长为0.6 计算数据如下表

步长h	近似值	误差
0.6000000000	1.2497980870510	0.7017615218859
0.3000000000	2.2915373035017	0.3399776945648
0.3000000000	2.0056913361987	0.0541317272619
0.0750000000	1.9542557514886	0.0026961425517
0.0750000000	1.9517143166910	0.0001547077541
0.0187500000	1.9515690583876	0.0000094494507
0.0187500000	1.9515601960840	0.0000005871471
0.0046875000	1.9515596455797	0.0000000366428
0.0046875000	1.9515596112262	0.0000000022893
0.0011718750	1.9515596090800	0.0000000001431
0.0011718750	1.9515596089458	0.0000000000089
0.0002929687	1.9515596089372	0.0000000000003
0.0002929687	1.9515596089365	0.0000000000004
0.0000732422	1.9515596089389	0.0000000000020
0.0000732422	1.9515596089401	0.0000000000032
0.0000183105	1.9515596089366	0.0000000000003
0.0000183105	1.9515596089203	0.0000000000166

Table 8: 函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$计算结果

4. 对于第四个函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.1000000000	2.7381460011626	0.2273688280228
0.0500000000	3.0541478748924	0.0886330457071
0.0500000000	2.9719333541517	0.0064185249663
0.0125000000	2.9659305074186	0.0004156782332
0.0125000000	2.9655410390865	0.0000262099011
0.0031250000	2.9655164709105	0.0000016417251
0.0031250000	2.9655149318496	0.0000001026642
0.0007812500	2.9655148356025	0.0000000064171
0.0007812500	2.9655148295864	0.0000000004010
0.0001953125	2.9655148292114	0.0000000000260
0.0001953125	2.9655148291883	0.0000000000029
0.0000488281	2.9655148291806	0.0000000000048
0.0000488281	2.9655148291844	0.0000000000010
0.0000122070	2.9655148292022	0.0000000000168
0.0000122070	2.9655148291931	0.0000000000078
0.0000030518	2.9655148291386	0.0000000000468
0.0000030518	2.9655148291810	0.0000000000044
0.0000007629	2.9655148293629	0.0000000001775
0.0000007629	2.9655148295084	0.0000000003230

Table 9: 函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$计算结果

5. 对于第四个函数$f(x) = x^{x^x}; x = 0.0001$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.0000500000	1.0150760914859	0.0001300144143
0.0000250000	1.0152071604124	0.0000010545122
0.0000250000	1.0152061668565	0.0000000609562
0.0000062500	1.0152061096412	0.0000000037409
0.0000062500	1.0152061061330	0.0000000002328
0.0000015625	1.0152061059148	0.0000000000146
0.0000015625	1.0152061059012	0.0000000000009
0.0000003906	1.0152061059003	0.0000000000000
0.0000003906	1.0152061059002	0.0000000000000

Table 10: 函数$f(x) = x^{x^x}; x = 0.0001$计算结果

5 .3 P261 3

1. 对于第一个函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$，经过尝试采取初始步长为0.0140625 计算数据如下表

步长h	近似值	误差
0.0140625000	46.9631282343973	28.2103664607763
0.0035156250	75.1734946669405	28.2103664607763
0.0035156250	75.1734946951752	0.0000000000015
0.0008789063	75.1734946951721	0.0000000000015
0.0008789063	75.1734946951726	0.0000000000010
0.0002197266	75.1734946951953	0.0000000000010

Table 11: 函数$f(x) = 60x^{45} - 32x^{33} + 233x^5 - 47x^2 - 77; x = 1/\sqrt{3}$计算结果

2. 对于第二个函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$，经过尝试采取初始步长为0.7 计算数据如下表

步长h	近似值	误差
0.3500000000	1.2400173943055	0.0114199709294
0.0875000000	1.2280390318868	0.0114199709294
0.0875000000	1.2285959193804	0.0000015039957
0.0218750000	1.2285974316187	0.0000015039957
0.0218750000	1.2285974233771	0.0000000000009
0.0054687500	1.2285974233761	0.0000000000009
0.0054687500	1.2285974233761	0.0000000000000
0.0013671875	1.2285974233761	0.0000000000000

Table 12: 函数$f(x) = tan\left(cos\left(\frac{\sqrt{5}+sin(x)}{1+x^2}\right)\right); x = \frac{1+\sqrt{5}}{3}$计算结果

3. 对于第三个函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$，经过尝试采取初始步长为0.6 计算数据如下表

步长h	近似值	误差
0.3000000000	2.3608008116819	0.4092412027450
0.0750000000	1.9789971259583	0.4092412027450
0.0750000000	1.9501003980584	0.0014592108785
0.0187500000	1.9515632129140	0.0014592108785
0.0187500000	1.9515596147538	0.0000000058169
0.0046875000	1.9515596089339	0.0000000058169
0.0046875000	1.9515596089368	0.0000000000000
0.0011718750	1.9515596089369	0.0000000000000

Table 13: 函数$f(x) = sin\left(cos\left(\frac{1}{x}\right)\right); x = \frac{1}{\sqrt{2}}$计算结果

4. 对于第四个函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.0500000000	2.9930984860332	0.0275836568478
0.0125000000	2.9655511013447	0.0275836568478
0.0125000000	2.9655148131543	0.0000000160311
0.0031250000	2.9655148291820	0.0000000160311
0.0031250000	2.9655148291854	0.0000000000001
0.0007812500	2.9655148291854	0.0000000000001

Table 14: 函数$f(x) = sin(x^3 - 7x^2 + 6x + 8); x = \frac{1-\sqrt{5}}{2}$计算结果

5. 对于第五个函数$f(x) = x^{x^x}; x = 0.0001$，经过尝试采取初始步长为0.1 计算数据如下表

步长h	近似值	误差
0.0000001000	1.0152061059005	0.0000000000002
0.0000000010	1.0152061059010	0.0000000000002

Table 15: 函数$f(x) = x^{x^x}; x = 0.0001$计算结果

5 .4 P268 1

距离$cos(x)$说明

0.000000	1.000000
0.100000	0.995000
0.200000	0.980100
0.300000	0.955300
0.400000	0.921100
0.500000	0.877600
0.600000	0.825300
0.700000	0.764800
0.800000	0.696700
0.900000	0.621600
1.000000	0.540300

Table 16: $cos(x)$的数据点

牛顿多项式系数表如下

a_0	1.000000
a_1	-0.005000
a_2	-0.004950
a_3	0.000000
a_4	0.000021
a_5	-0.000007
a_6	0.000002
a_7	-0.000001
a_8	0.000000
a_9	-0.000000
a_10	0.000000

Table 17: 牛顿多项式系数

算得$cos(x)$在x=0的时候为-0.005002。

6 实验结果分析

- 在使用中心差分公式的使用中，选择合适的初始值很重要，每次步长的变化也很重要。在很多时候，初始值不合适就会导致计算无结果，或者步长变化不当也会导致最后计算出坏的结果。所以寻找合适的初始值，还有合适的配套的步长变化很重要。现在摸索一个经验规律，大概初始值是素数的时候会更大的几率得出适合的近似值。
  在这次的实验中，利用中心差分公式计算导数值得时候几乎很难找到合适的初始值来达到小数点后13位的精确度，大部分情况是因外步长变化的时候误差由小变大而导致的循环结束。所以适当的时候需要理论分析出合适的步长再来尝试得到需要的精确度。

- 理查森外推公式比中心差分更好用。中心差分很难得到需要的精度的结果，但是外推公式很容易得到需要的精度。推测原因是使用外推公式时不用在意步长只要低精度数据足够多就可以推出高精度结果，但是中心差分公式对步长的要求很高，必须步长恰好合适才能得出足够的精度。
  关于这一点有一个猜测：切比雪夫节点、或者其它的构造节点方式是否可以提高中心差分公式的计算效率？

- 牛顿多项式微分在应用的时候保证以后的节点和第一个节点的距离逐渐增大，且为等距点，这样可以估算出随着多项式的误差。但是在实际应用中或许更应该使用不等距的切比雪夫节点，这样多项式更好的拟合，精度自然也就提高了，但是目前尚不知对于任意形式节点牛顿多项式微分的误差如何。

7 实验代码

7 .1 P260 1&2

main.c

```
  #include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x);
double value_of_f(double x, double h);

int main()
{
    freopen("out.txt","w",stdout);
    double Dk,Dk_next;
    double error,error_next;
    double h,h_next;
    double x;
    double tol;
    int i;
    double fen=2;
    double truev = 75.173494695173620;

    x = 1 / sqrt(3);//function 1
    //x = (1 + sqrt(5)) / 3;//function 2
    //x = 1 / sqrt(2);//function 3
    //x = (1 - sqrt(5)) / 2;//function 4
    //x = 0.0001;//function 5
    tol = 1e-13;
    h = h_next = 0.9;
    //h = h_next = 0.00005;

    Dk_next = (f(x + h) - f(x - h)) / (2 * h);
    printf("%.10lf  &  %.13lf  &  %.13lf\\\\\\hline\n"
    ,h,Dk_next,fabs(Dk_next-truev));
    h = h/fen;

    do{
        Dk = value_of_f(x,h);
        error = fabs(Dk - truev);

        printf("%.10lf  &  %.13lf  &  %.13lf\\\\\\hline\n"
        ,h,Dk,fabs(Dk-truev));

        h_next = h / fen;
        Dk_next = value_of_f(x,h_next);
        error_next = fabs(Dk_next - truev);
        printf("%.10lf  &  %.13lf  &  %.13lf\\\\\\hline\n"
        ,h,Dk_next,fabs(Dk_next-truev));
```

```
        h = h_next/fen;
        if(error < tol)printf("\ncondition:error < tolerance\n");
        if(error_next >= error)printf("\ncondition:error_next >= error\n");
    }while(error > tol && error_next < error);

    printf("the h is: %.13lf\nthe result is: %.14lf\n", h_next, Dk_next);
    return 0;
}

double f(double x)
{
    return (60 * pow(x,45) - 32 * pow(x,33) + 233 * pow(x,5)
     - 47 * pow(x,2) - 77);//function 1
    //return (tan( cos( (sqrt(5) + sin(x)) / (1 + pow(x,2))  ) ) );//function 2
    //return (sin(cos(1 / x)));//function 3
    //return sin(pow(x,3) - 7 * pow(x,2) + 6 * x + 8);//function 4
    //return pow(x,pow(x,x));//function 5
}

double value_of_f(double x, double h)
{
    return (f(x + h) - f(x - h)) / (2 * h);
    //return (-f(x + 2*h) + 8*f(x + h) - 8*f(x-h)+f(x-2*h)) / (12 * h);;
}
```

7 .2 P261 3

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double D(double x,double k, double h);
double f(double x);

int main()
{
    //freopen("out.txt","w",stdout);
    int k,count;
    double result,result_next,error,error_next,tol;
    double h,h_next,x;
    double fen=10;
    double truev = 1.015206105900236;
```

```
    k = 7;
    h = h_next = 0.000001;
    //x = 1 / sqrt(3);
    //x = (1 + sqrt(5)) / 3;//function 2
    //x = 1 / sqrt(2);//function 3
    //x = (1 - sqrt(5)) / 2;//function 4
    x = 0.0001;//function 5
    count = 0;
    tol = 1e-13;

    result_next = D(x,k,h);
    h = h/fen;

    do{
            result = D(x,k,h);
            error = fabs(result - truev);
            printf("%.10lf  &  %.13lf  &  %.13lf\\\\\\hline\n"
            ,h,result,fabs(result-truev));
            h_next = h/fen;
            result_next = D(x,k,h_next);
            error_next = fabs(result_next - truev);
            h = h_next/fen;
            printf("%.10lf  &  %.13lf  &  %.13lf\\\\\\hline\n"
            ,h,result_next,fabs(result-truev));
            count++;
    }while(error > tol && count < 50 && error_next < error);

    printf("result = %.14lf\n",result);
    return 0;
}

double D(double x,double k, double h)
{
    if(k == 0){
        return (f(x + h) - f(x - h)) / (2 * h);
    }
    if(k != 0){
        return (pow(4,k) * D(x,k-1,h) - D(x,k-1,2*h)) / (pow(4,k) - 1);
    }
}

double f(double x)
{
    //return (60 * pow(x,45) - 32 * pow(x,33) + 233 * pow(x,5)
                                        // - 47 * pow(x,2) - 77);//function 1
```

```
    //return (tan( cos( (sqrt(5) + sin(x)) / (1 + pow(x,2))  ) ) );//function 2
    //return (sin(cos(1 / x)));//function 3
    //return sin(pow(x,3) - 7 * pow(x,2) + 6 * x + 8);//function 4
    return pow(x,pow(x,x));//function 5
}
```

7 .3 P270 1

```
  main.c

  #include <stdio.h>
#include <stdlib.h>
#include "Newton.h"

int main()
{
    //freopen("in.txt","r",stdin);
    freopen("test.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    Nodes *nd;
    double **DD;
    double unknown;
    Vector *coe;
    Vector *res;
    nd = malloc(sizeof(Nodes));
    coe = malloc(sizeof(Vector));
    res = malloc(sizeof(Vector));

    InitNodesInfo(nd);
    ShowNodesInfo(nd);

    DD = malloc((nd->num)*sizeof(double*));
    CalcuDD(DD,nd);
    ShowDDtable(DD,nd);

    InitCoe(DD,coe,nd);
    ShowVector(coe);

    ShowPoly(coe,nd);

    //printf("Input the unknown:");
    //scanf("%lf", &unknown);
    //unknown = 1;
    //CalcuVal(coe,nd,res,unknown);
    CalcuDiffValue(coe,nd,res);
    return 0;
```

```
}
```

Newton.h

```
#ifndef NEWTON_H_INCLUDED
#define NEWTON_H_INCLUDED
/*************************************************************
Module Name: Newton.h
Module Date: 10/29/14
Module Auth: Xuanyu Wang
Description: Use Newton approximation method.
Revision History:
Date Rel Ver. Notes
month/day/year x.x [e.g.] Module created
*************************************************************/
/*----------------Includes--------------*/
#include <math.h>
/*---------Structures and Typedefs-------*/
struct Vec
{
    int num;//the number of vector's elements
    double *v;//the value of vector's element
};
typedef struct Vec Vector;
struct Nodes
{
    int num;//the number of nodes
    double *node;//the value of node
    double *node_value;//the value of f(node)
};
typedef struct Nodes Nodes;

/*************************************************************
 Function Name: InitNodesInfo(Nodes* nd)
 Function Description: assign the number of nodes. assign every
        node's info include node's value and the value of f(node)
Inputs: need the point of Nodes, and the point had been allocated room/
 Outputs: No output
*************************************************************/
void InitNodesInfo(Nodes* nd)
{
    int i;

    printf("Please input the number of nodes you have:");
    scanf("%d", &(nd->num));//get the number of nodes
    printf("nd->num = %d\n",nd->num);//verify the number
```

```
    //allocate the room for node and it's value
nd->node = (double*)malloc(nd->num * sizeof(double));
    nd->node_value = (double*)malloc(nd->num * sizeof(double));
    //assign the node's value and f(node)'s value
for (i = 0; i < nd->num; i++){
printf("the NO.%d node and it's value are:",i);
scanf("%lf%lf", (nd->node + i),(nd->node_value + i));
}
}

/************************************************************
 Function Name: ShowNodesInfo(Nodes* nd)
 Function Description: display the info of Nodes,including  the
                number of nodes,every node's info include node's
                value and the value of f(node)
Inputs: need the point of Nodes, and the point had been initialized in function
                InitNodesInfo(Nodes* nd)
 Outputs: No output
************************************************************/
void ShowNodesInfo(Nodes* nd)
{
    int i;
    printf("nd->num = %d\n",nd->num);
    //display the info of each node.
for (i = 0; i < nd->num; i++){
printf("the NO.%d node and it's value are :
        \t%lf  &  %lf\\\\\\hline\n",
        i, *((nd->node)+i),*(nd->node_value+i));
}
}

/************************************************************
 Function Name: ShowNodesInfo(Nodes* nd)
 Function Description: calculate the divided difference.
Inputs: need the point of Nodes and double**. points had been
                initialized in function InitNodesInfo(Nodes* nd)
 Outputs: No output
************************************************************/
void CalcuDD(double** DD,Nodes* nd)
{
    int i, n, j;
    n = nd->num;
    for(i = 0; i < n; i++){
            //allocate the room
        *(DD+i) = (double*)malloc((n-i)*sizeof(double));
        for(j = 0; j < n-i; j++){
```

```
            //the first-order divided difference would be assigned directly
            if(i == 0){
                *(*(DD+i)+j) = *((nd->node_value)+j);
            }
            //other divided difference would be calculated
            else{
                *(*(DD+i)+j) = (*(*(DD+i-1)+j+1) - *(*(DD+i-1)+j))
                            / (((nd->node)+i+i+j)-((nd->node)+i+j));
            }
        }
    }
}

/*************************************************************
 Function Name: ShowDDtable(double **DD,Nodes *nd)
 Function Description: display the divided difference table.
Inputs: need the point of Nodes and double**. points had been
            initialized in function InitNodesInfo(Nodes* nd).
 Outputs: No output
*************************************************************/
void ShowDDtable(double **DD,Nodes *nd)
{
    int i, n, j;
    n = nd->num;
    for(i = 0; i < n; i++){
        for(j = 0;j<n-i;j++){
            //printf("No.(%d,%d) = %.1lf\t\t",i, j, *(*(DD+i)+j));
            printf("%.10lf\t",*(*(DD+i)+j));
        }
        printf("\n");
    }
}

/*************************************************************
 Function Name: InitCoe(double **DD, Vector *coe, Nodes *nd)
 Function Description: As we had divided difference table,
                   we can assign a_k.
Inputs: need the point of Nodes and double**. points had been
            initialized in function InitNodesInfo(Nodes* nd).
            the coe is used to save coefficient.
 Outputs: No output
*************************************************************/
void InitCoe(double **DD, Vector *coe, Nodes *nd)
{
    int i;
    //initialize the coe
```

```
    coe->num = nd->num;
    coe->v = malloc((coe->num)*sizeof(double));
    //assign value to a_k
    for(i = 0; i < coe->num; i++){
        *((coe->v) + i) = *(*(DD+i)+0);
    }
}

/*************************************************************
 Function Name: ShowVector(Vec *vec)
 Function Description: display the vector's info.
Inputs: need the point of Vec, and the point had been initialized.
 Outputs: No output
*************************************************************/
void ShowVector(Vector *vec)
{
    int i;
    //for every display the number of it and the value of it.
    for(i = 0; i < vec->num; i++){
        //printf("The No.%d value is :%lf \n", i, *((vec->v)+i));
        printf("$a_%d$   &  %lf\\\\\\hline \n", i, *((vec->v)+i));
    }
}
/*************************************************************
Function Name: ShowPoly(Vector *coe, Nodes *nd)
Function Description: display the polynomial.
Inputs: need the point of Nodes, and the point had been initialized.
            need the point of coe, and the point had been initialized.
Outputs: No output
*************************************************************/
void ShowPoly(Vector *coe, Nodes *nd)
{
    int i, n, j;
    n = nd->num;
    for(i = 0; i < n; i++){
        //the 1st coefficient and the negative coefficient don;t need '+'
        if(i != 0 && *((coe->v)+i) > 0){
            printf("+");
        }
        //0 don't need to output
        if(*((coe->v)+i) != 0){
            printf("%lf",*((coe->v)+i));
            //for node which is negative don't need '+'
            for(j = 0; j < i; j++){
                //some number is too little might can't judge whether it's
                //positive or negative.
```

```
                if(*((nd->node)+j)*1000 >= 0){
                    printf("*(x-%lf)",*((nd->node)+j));
                }
                else if(*((nd->node)+j)*1000 < 0){
                    printf("*(x%lf)",-*((nd->node)+j));
                }
            }
        }
        if(*((coe->v)+i+1) != 0){
            //this section can show the polynomial in multiline
            //printf("\n");
        }
    }
    printf("\n");
}
/************************************************************
Function Name: CalcuVal(Vector *coe, Nodes *nd,Vector *res, double unknown)
Function Description: the result of substitution of unknown.
Inputs: need the point of Nodes, and the point had been initialized.
            need the point of coe, and the point had been initialized.
            and need the unknown.
Outputs: No output
************************************************************/
void CalcuVal(Vector *coe, Nodes *nd,Vector *res, double unknown)
{
    int i, n, j;
    double test = 0, temp = 1;
    n = nd->num;

    res->v = malloc((res->num)*sizeof(double));
    for(i = 0; i < n; i++){
        temp *= *((coe->v)+i);
        for(j = 0; j < i; j++){
            temp *= (unknown - *((nd->node)+j));
        }
        test += temp;
    }
    printf("result = %lf\n",test);
}

/************************************************************
Function Name: CalcuVal(Vector *coe, Nodes *nd,Vector *res, double unknown)
Function Description: the result of substitution of unknown.
Inputs: need the point of Nodes, and the point had been initialized.
            need the point of coe, and the point had been initialized.
            and need the unknown.
```

```
Outputs: No output
************************************************************/
void CalcuDiffValue(Vector *coe, Nodes *nd,Vector *res)
{
    int i, n, j;
    double test = 0, temp = 1;
    double unknown;
    unknown = *(nd->node+0);
    n = nd->num;

    res->v = malloc((res->num)*sizeof(double));
    for(i = 1; i < n; i++){
        temp *= *((coe->v)+i);
        for(j = 1; j < i; j++){
            temp *= (unknown - *((nd->node)+j));
        }
        test += temp;
    }
    printf("result = %lf\n",test);
}

#endif // NEWTON_H_INCLUDED
```