评分：

SHANGHAI UNIVERSITY

**课程论文**

**COURSE PAPER**

**学 院 理学院**

**专 业 数学与应用数学**

**学 号 17122906**

**学生姓名 金意诚**

**课程 计算实习B1**

**Mathematica软件在求解方程近似根中的应用**

金意诚

（上海大学 数学系）

**摘 要：**本文分别按牛顿迭代法和对分区间法，利用Mathematica编程计算同一方程的近似根，通过比较结果，得出牛顿迭代法收敛速度快于对分区间法，且误差较小，并对牛顿迭代法作进一步优化，设计可视化界面，为解没有求根公式的方程提供帮助。

**关键词：**Mathematica软件；牛顿迭代法；对分区间法；动态可视化

**Application of Mathematica in Solving Approximate Root of Function**

JIN Yicheng

（Department of Mathematics，Shanghai University）

**Abstract:** In this paper, according to the principle of Newton iteration theory and the principle of Opposite Dividing Method for Intervals respectively, we calculate the approximate root of the same function, by comparing the results, Newton derived the convergence rate faster and a smaller error than that of sub-interval method. For further optimization, we design a visual interface to help solve equations without root formula.

**Keyword:** Mathematica software; Newton iteration theory; sub-interval method; dynamic visualization

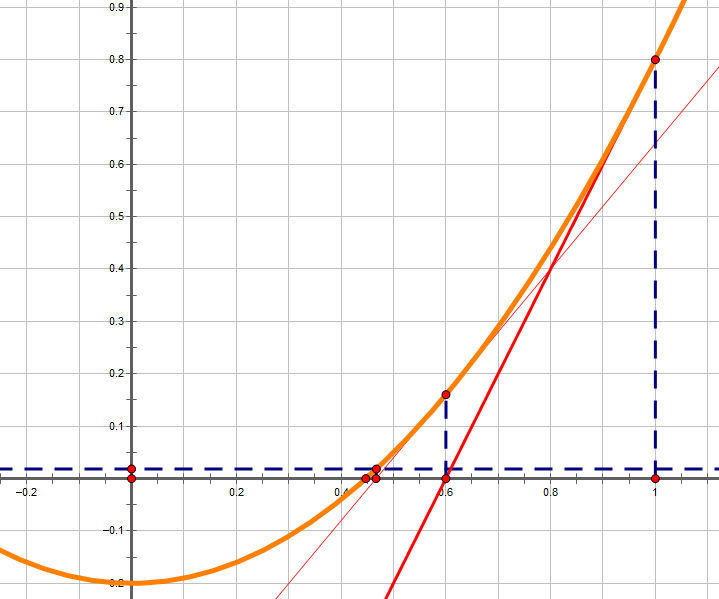
**0. 引言**

Mathematica软件是Wolfram Research公司1988年开发的一款科学计算类软件，具有集数值和符号计算、编程语言及图形绘制等为一身的强大功能。在高等数学、微分方程和数学物理实验等大学数学课程中Mathematica软件得到了很好的应用，对科学、工程和数学问题中需要求解方程时提供很大的帮助。本文从牛顿迭代法和对分区间法在Mathe-matica求解方程近似根的方法出发，比较这两种方法的优缺点，并对牛顿迭代法作进一步优化，设计可视化界面。通过对牛顿迭代法的编程，以动画形式实现了求解过程，动态过程可手动操控，同时也便于判断牛顿收敛法是否适用与使用者所选的方程。

**1. 利用牛顿迭代法求方程的根**

**1.1 理论依据**

牛顿迭代法又称为牛顿-拉夫逊方法，它是牛顿在17世纪提出的一种在实数域和复数域上近似求解方程的方法。牛顿法利用线性化的思想，以切线方程的根代替方程的根，从而产生一系列不断接近根的近似值设曲线可微。



**图1 牛顿迭代法的求值过程**

**Fig. 1 The evaluation process of Newton iteration method**

设r是的根，选取作为r的初始近似值，过点,作曲线的切线L，L的方程为，求出L与轴交点的横坐标：

为r的一次近似值。过点作曲线的切线，并求该切线与轴交点的横坐标

为r的二次近似值。重复以上过程，得r的近似值序列，其中

为r的n+1次近似值，上式称为牛顿迭代公式。

用牛顿迭代法解非线性方程，是把非线性方程线性化的一种近似方法。把在点的某邻域内展开成泰勒级数

取其线性部分（即泰勒展开式的前两项），并令其等于0，即，以此作为非线性方程的近似方程，若，则其解为

这样，得到牛顿迭代法的一个迭代关系式

**1.2 牛顿迭代法的Mathematica程序**

以为例，选取初值，计算函数的所有近似零点，误差不超过

1.2.1 程序代码

1. f[x\_]=Ex-x2;
2. x=-2;
3. data={{"n","x(n)"},{0,x}};
4. Do[x=x-  f[x] ;AppendTo[data,{n,x}],{n,1,8}];
5. f'[x]
6. N[TableForm[data],5]

1.2.2 程序运行结果

1. n  x(n)
2. 0.0000 -2.0000
3. 1.0000  -1.06545
4. 2.0000  -0.746073
5. 3.0000  -0.704169
6. 4.0000  -0.703468
7. 5.0000  -0.703467
8. 6.0000  -0.703467
9. 7.0000  -0.703467
10. 8.0000  -0.703467

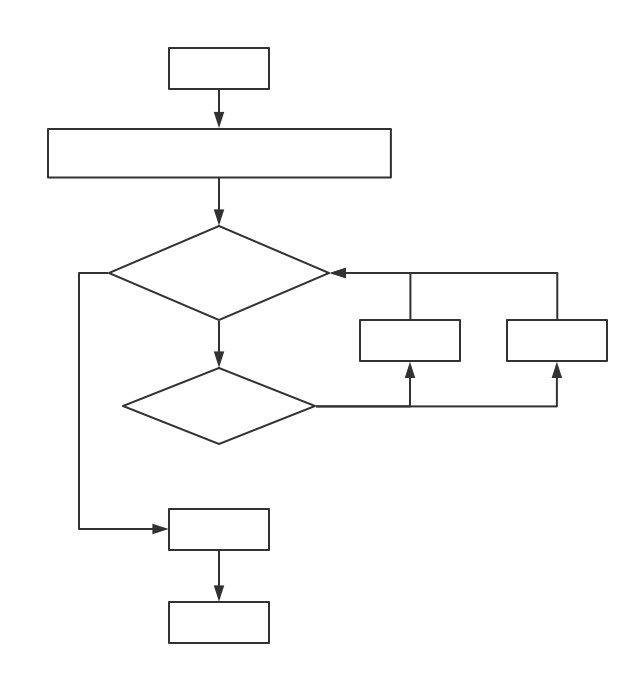
用时：0.02秒

**2. 利用对分区间法求方程的根**

**2.1 理论依据**

设函数在闭区间上连续，且在端点处取非零异号的函数值和，即。由零点存在定理，方程在区间内必有根。将区间一分为二，首先检查中点是否为根，是则结束计算，否则留下端点处函数值异号的区间。该区间一定包含方程的根，再将该区间对分，再检查中点是否为根，如此往复，每一次对分，包含根的区间长度都将缩短一半，直至包含根的区间长度小于给定的误差，取该区间的中点作为根的近似值。

**2.2 算法流程**



或

结束

输出

否

是

否

是

开始

赋初值：

**2.3 对分区间法的Mathematica程序**

1. y[x\_] := E^x - x^2;
2. x1 = -2;
3. x2 = 0;
4. eps = 0.000001;
5. nmax = 100;
6. ans = {{"n", "x"}};
7. For[i = 1, i < nmax, i++, x = (x1 + x2)/2.;
8. ans = Append[ans, {i, x}];
9. If[Abs[y[x]] <= eps, Break[]];
10. If[y[x1]\*y[x] < 0, x2 = x, x1 = x];];
11. ans // MatrixForm

**2.4 运行结果**

1. n  x
2. 1  -1.
3. 2  -0.5
4. 3  -0.75
5. 4  -0.625
6. 5  -0.6875
7. 6  -0.71875

用时：0.02秒

1. 7  -0.703125
2. 8  -0.710938
3. 9  -0.707031
4. 10  -0.705078
5. 11  -0.704102
6. 12  -0.703613
7. 13  -0.703369
8. 14  -0.703491
9. 15  -0.70343
10. 16  -0.703461
11. 17  -0.703476
12. 18  -0.703468
13. 19  -0.703465
14. 20   -0.703466
15. 21  -0.703467

**3. 牛顿迭代法与对分区间法的比较**

**3.1 收敛速度**

牛顿法迭代5次后，得到结果：-0.703467，误差：；对分区间法在迭代 21 次后，得到结果：-0.703468，误差：，由结果可以看出，牛顿法收敛速度明显快于对分区间法，且误差较小。

**3.2 适用范围**

牛顿法和对分区间法都需要通过计算或图形来估计方程根r的位置，初始近似值的选取要尽可能接近r。此外，使用牛顿法时，要求收敛。

**4. 牛顿迭代法的可视化设计**

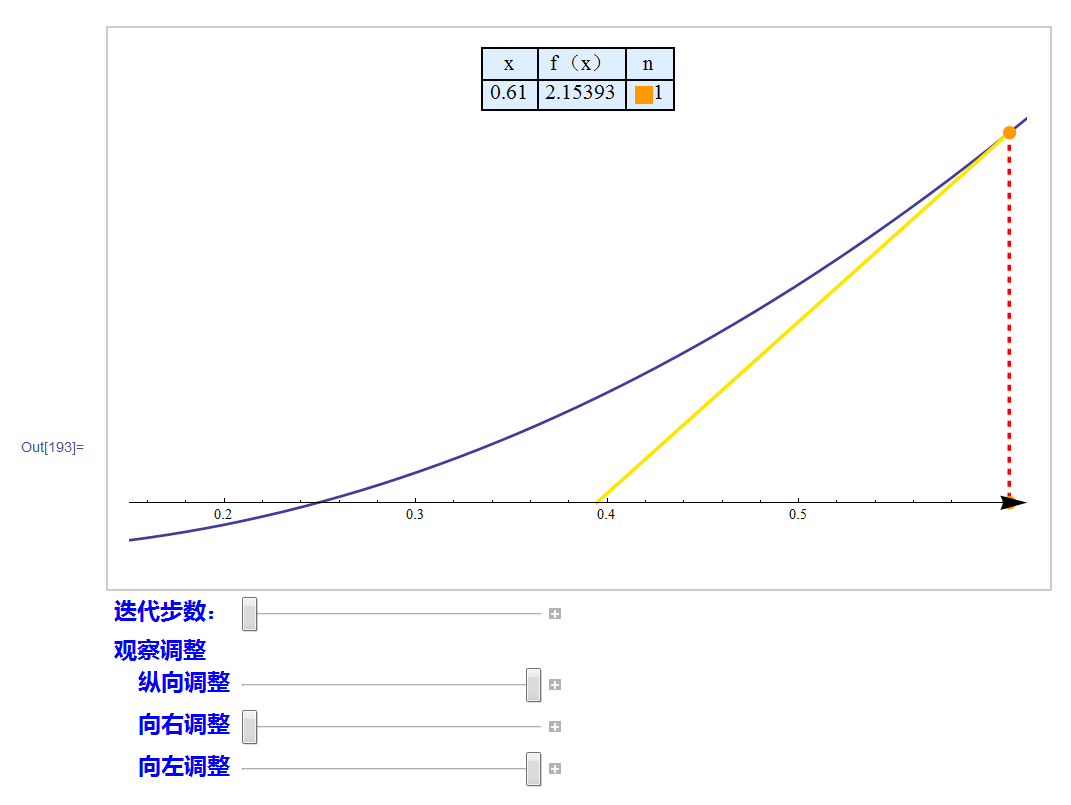
以方程为例

**4.1 程序代码**

1. Manipulate[
2. f[x\_] := 8\*x^2 - Sin[x] - 0.25;
3. cishu = 150; wucha = 0.000001; jingdu = 0.000001;
4. x0 = 0.61; z = 10; x2 = {0.61};
5. For[j = 1, j < cishu, j++,
6. x1 = x0 - f[x0]/z;
7. x2 = Append[x2, x1];
8. If[Abs[x0 - x1] > jingdu,
9. x0 = x1, Break[]];];
10. p = Plot[f[x], {x, 0.15, 0.65}

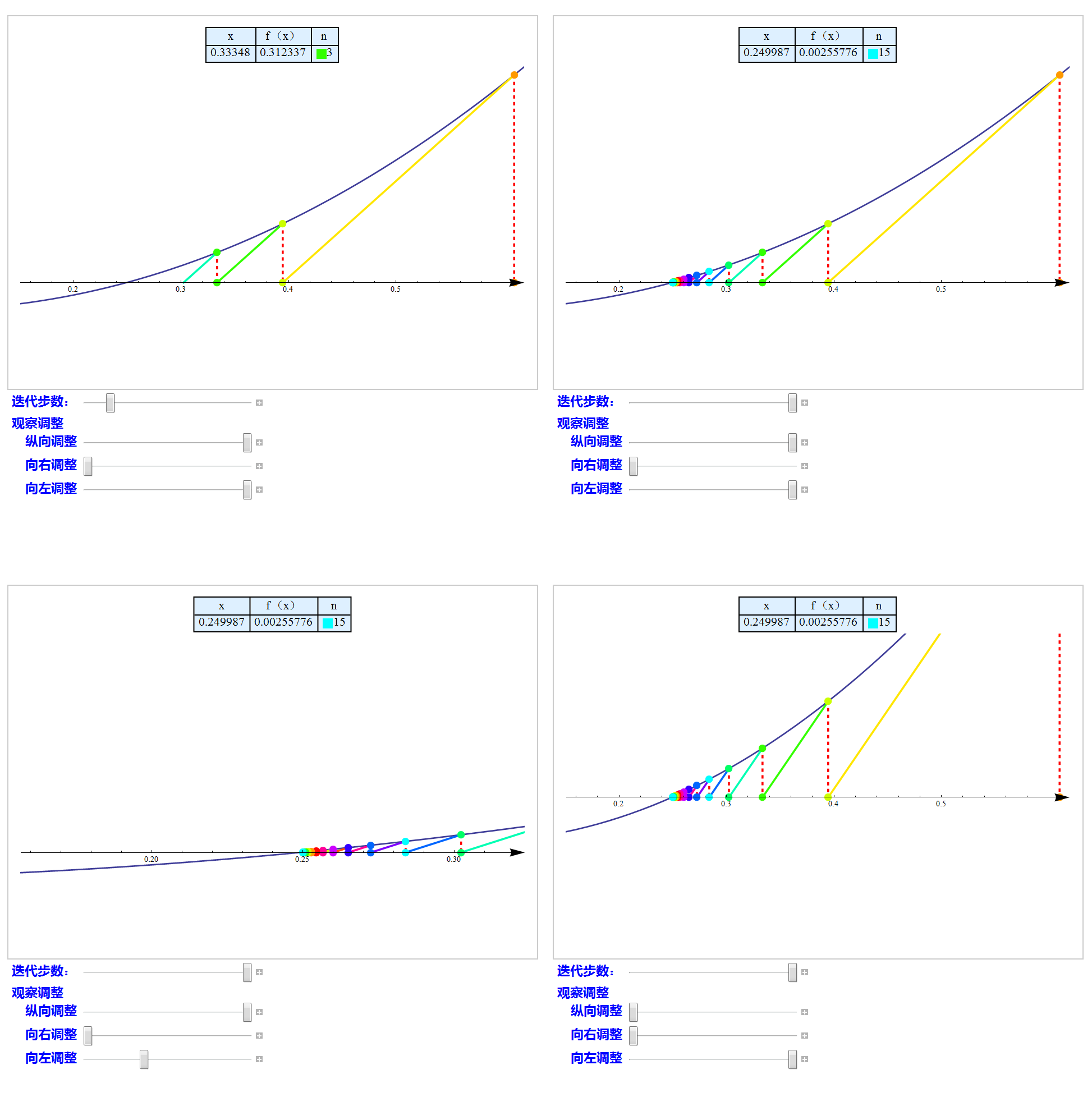
**4.2 可视化设计**

1. , PlotStyle -> Thickness[0.003]
2. , PlotLabel -> Style[Grid[{{"x", "f（x）", "n"}, {x2[[t]], f [x2[[t]]]
3. , Row[{Graphics[{Hue[0.1 t], Rectangle[]}, ImageSize -> 13], t}]}}
4. , Frame -> All, Background -> LightBlue], 14],
5. AxesStyle -> Arrowheads[0.03]];
6. fig = Table[
7. Graphics[{{Thickness[0.004], Hue[0.15 j],
8. Line[{{x2[[j]], f[ x2[[j]]]}, {x2[[j + 1]], 0}}]}
9. , {Red, Dashed, Thickness[0.004],
10. Line[{{x2[[j]], f [x2[[j]]]}, {x2[[j]], 0}}]}
11. , {Hue[0.1 j], PointSize[0.015],
12. Point[{x2[[j]], 0}]}, {Hue[0.1 j], PointSize[0.015],
13. Point[{x2[[j]], f [x2[[j]]]}]}}], {j, 1, 15}];
14. Show[{p, Table[fig[[s]], {s, 1, t}]}, ImageSize -> 600,
15. PlotRange -> {{c, d}, {-0.9, e}}]
16. , {{t, 1, Style["迭代步数：", Blue, Bold, 15]}, 1, 15, 1}
17. , Style["观察调整", Blue, Bold, 15]
18. , {{e, 2.2, Style["纵向调整", Blue, Bold, 15]}, 1, 2.2, 0.1}
19. , {{c, 0.16, Style["向右调整", Blue, Bold, 15]}, 0.16, 0.61, 0.04}
20. , {{d, 0.61, Style["向左调整", Blue, Bold, 15]}, 0.16, 0.61, 0.04},
21. ControlPlacement -> Bottom]



**图2 牛顿迭代法可视化面板初始状态**

**Fig. 2 Newton iteration method visual panel initial state**

随着迭代步数的增加，平行直线与轴的交点逐步靠近曲线 与轴的交点（方程的根），在满足精度的条件下，可得方程的根。界面上方的表格内，n 为迭代次数，为迭代点的值，迭代点对应的函数值为，这些值均实时变化。界面左下方，设置观察调整按钮，可纵向调整、向右调整和向左调整，便于调整观察角度。

**移动滑块“向左调整”，界面向左移动**

**移动滑块“纵向调整”，界面向上移动**

**移动滑块“迭代步数”，迭代步数增加**

**图3 牛顿迭代法可视化面板操作图例**

**Fig. 3 Newton iteration method visual panel operation legend**

上面的程序中，若迭代步数n = 13 ，此时得 = 0. 250882 ，0. 0052763 ，求出了满足精度的方程的根。程序中的方程也可以换成其他的类型，或者设计具有下拉菜单功能的控件， 通过选择不同的方程，动态演示每个方程求根的可视化过程，达到人机交互的目的。

**5. 对牛顿迭代法和对分区间法的进一步优化**

如何选取初值是使用牛顿迭代法中最大的问题，对于方程，选取任何初值，均收敛，而对于方程，若初值选取错误，牛顿迭代法将无法使用，且使用对分区间法效果也不显著（见下表）。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 二分法程序结果 | | |  | 牛顿迭代法程序结果（初值=0） | | |
| 迭代数 | **区间值** | **区间值** |  | **迭代数** | **迭代初始值** | **迭代输出值** |
| 1 | 0.00000 | 500.00000 |  | **1** | 0.00000 | **不存在** |
| 2 | 0.00000 | 250.00000 |  | **2** | **不收敛** | **不收敛** |
| 3 | 0.00000 | 125.00000 |  | **牛顿迭代法程序结果（初值=10）** | | |
| 4 | 0.00000 | 75.00000 |  | **迭代数** | **迭代初始值** | **迭代输出值** |
| … |  |  |  | **1** | 10.00000 | -5.15714 |
| 20 | 52.40631 | 52.40727 |  | **…** |  |  |
| 21 | 52.40679 | 52.40679 |  | **26** | 52.40693 | 52.40693 |

尝试使用对分区间法直至包含解的区间长度小于一定长度，将区间值作为初值代入牛顿迭代法中（见下表）。

|  |  |  |
| --- | --- | --- |
| 牛顿迭代法和二分法结合的程序结果 | | |
| 迭代数 | **迭代初始值** | **迭代输出值** |
| 1 | 50.00000 | 52.64400 |
| 2 | 52.40571 | 52.40872 |
| 3 | 52.40673 | 52.40873 |

显然，通过将两种方法结合起来解决复杂方程的求解问题，取得了显著地效果。迭代次数大大减少，极大降低了计算的复杂性。如何针对不同的方程构造不同的迭代公式，如何降低对初值的依赖性和提高收敛的速度，都是方程求根问题中经常考虑的问题。利用牛顿迭代法和二分法各自的特点，并把两种方法结合起来，先通过二分法找到合适的初始值，然后再用牛顿法进行迭代运算，大大节约了时间，降低了计算复杂性。

**6. 结语**

方程的解法主要有对分区间法和牛顿迭代法，对分区间法对函数的要求低，计算简单但收敛慢；对于牛顿迭代法，若初值选取得好则收敛快，将两种方法结合，既减少了计算量，也加快了收敛速度。

通过实例可以看出利用 Mathematica 软件可以快速准确的求解方程的近似根，为我们解不存在求根公式的方程提供帮助，程序编写结束后，将完成的 Mathematica Notebook 文档保存为.nb 格式，便可在 Mathematica环境下运行；程序中涉及的图形可利用函数 Export 将其输出，图形以.gif 格式默认保存在“我的文档”中。Mathematica 软件可实现数学问题的直观性、互动性和交互性，对加快解决问题的速度和对数学问题的理解是大有帮助的。

**参考文献（References）**

［1］ 马千里. Mathematica软件在微积分教学中的应用［J］. 陕西师范大学学报（自然科学版），2009（S1）：135-136

［2］ 金哲，张凤兰. Mathematica与实验数据分析［J］. 延边大学学报（自然科学版），2003，29（1）：36-39.

［3］ 郑永凡，王艳青.基于Mathematica的交互动态可视化设计及其应用［J］. 辽宁大学学报（自然科学版），2010，37（4）：324-328.

［4］ 王福贵，侯建文. 基于 Mathematica 的线性方程组的可视化求解［J］. 计算机与现代化，2013（5）：149-154.

［5］ Maeder, Roman E. Programming in mathematica. Addison-Wesley Longman Publishing Co., Inc., 1991.