计算方法-第一次上机作业

PB19051183 吴承泽

实验源码：

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

#define ERROR -1
#define EPSILON 1E-5
#define MAXREPT 100

//函数1

double Function1(double x)
{
    return pow((x - 1), 3)  - x * x + x;
}

double Derivative_Function1(double x)
{
    return 3 * (x - 1) * (x - 1) - 2 * x + 1;
}

//函数2

double Function2(double x)
{
    return pow(sin(x), 3) + pow(cos(x), 3);
}

double Derivative_Function2(double x)
{
    return 3 * sin(x) * sin(x) * cos(x) - 3 * cos(x) * cos(x) * sin(x);
}

    /*
    二分法求解函数的根
    参数分别为函数Function调用函数指针，表征函数；a，b分别为二分法初始上下界
    若解存在，则返回值为近似解，否则返回-1
    */

double Bisection_method(double Function(double ), double a, double b)
{
    int i = 0;

    if(Function(a) * Function(b) >= 0)
        return ERROR;

    while(fabs(a - b) > EPSILON)
    {
        double mid = (a + b) / 2;
        i++;
```

```c
        printf("iteration %d : f(%.12f) = %.12f\n", i, mid, Function(mid));
        //printf("%.12f\n",fabs(Function(mid)));

        if(fabs(Function(mid)) < EPSILON)
            return mid;

        if(Function(a) * Function(mid) < 0)
        {
            b = mid;
        }
        else if(Function(mid) * Function(b) < 0)
        {
            a = mid;
        }
    }
    return (a + b) / 2;
}

    /*
    牛顿迭代法求近似根
    参数分别为：Function调用预设的函数的计算公式；Derivative_Function调用预设的导函数；x0
为初始迭代值
    若解存在，则返回值为近似解，否则返回-1
    */

double Newtons_method(double Function(double ), double
Derivative_Function(double ), double x0)
{
    int k;
    for(k = 1; k < MAXREPT; k++)
    {
        double x1 = x0 - Function(x0) / Derivative_Function(x0);

        printf("iteration %d : f(%.12f) = %.12f\n", k, x1, Function(x1));
        //printf("%.12f\n",fabs(Function(x1)));
        if(fabs(x1 - x0) < EPSILON)
        {
            return x1;
        }

        x0 = x1;

    }
    return ERROR;
}


int main()
{
    printf("\nFunction1 in Bisection method in [2,3]\n");
    Bisection_method(Function1, 2, 3);
    printf("\nFunction1 in Bisection method in [2.2,3]\n");
    Bisection_method(Function1, 2.2, 3);
    printf("\nFunction1 in Bisection method in [2,2.8]\n");
    Bisection_method(Function1, 2, 2.8);

    printf("\nFunction2 in Bisection method in [2,3]\n");
    Bisection_method(Function2, 2, 3);
    printf("\nFunction2 in Bisection method in [2.2,3]\n");
    Bisection_method(Function2, 2.2, 3);
    printf("\nFunction2 in Bisection method in [2,2.8]\n");
    Bisection_method(Function2, 2, 2.8);

    printf("\nFunction1 in Newton's method with initial value 2.25\n");
    Newtons_method(Function1,Derivative_Function1,2.25);
    printf("\nFunction1 in Newton's method with initial value 2.5\n");
    Newtons_method(Function1,Derivative_Function1,2.5);
    printf("\nFunction1 in Newton's method with initial value 2.75\n");
    Newtons_method(Function1,Derivative_Function1,2.75);

    printf("\nFunction2 in Newton's method with initial value 2.25\n");
    Newtons_method(Function2,Derivative_Function2,2.25);
    printf("\nFunction2 in Newton's method with initial value 2.5\n");
    Newtons_method(Function2,Derivative_Function2,2.5);
    printf("\nFunction2 in Newton's method with initial value 2.75\n");
    Newtons_method(Function2,Derivative_Function2,2.75);


    return 0;
}
```

实验结果：

输出说明：Function1为第一个函数 Function2为第二个函数

```
PS C:\VscodeWorkspace\计算方法> cd "c:\VscodeWorkspace\计算方法\" ; if ($?) { gcc HW1-PB19051183-吴承泽.c -o HW1-PB19051183-吴承泽 } ; if ($?) { .\HW1-PB19051183-吴承泽 }

Function1 in Bisection method in [2,3]
iteration 1 : f(2.500000000000) = -0.375000000000
iteration 2 : f(2.750000000000) = 0.546875000000
iteration 3 : f(2.625000000000) = 0.025390625000
iteration 4 : f(2.562500000000) = -0.189208984375
iteration 5 : f(2.593750000000) = -0.085601806641
iteration 6 : f(2.609375000000) = -0.031040191650
iteration 7 : f(2.617187500000) = -0.003059864044
iteration 8 : f(2.621093750000) = 0.011106431484
iteration 9 : f(2.619140625000) = 0.004008568823
iteration 10 : f(2.618164062500) = 0.000470676459
iteration 11 : f(2.617675781250) = -0.001295512426
iteration 12 : f(2.617919921875) = -0.000412647685
iteration 13 : f(2.618041992188) = 0.000028956956
iteration 14 : f(2.617980957031) = -0.000191859722
iteration 15 : f(2.618011474609) = -0.000081454972
iteration 16 : f(2.618026733398) = -0.000026249905
iteration 17 : f(2.618034362793) = 0.000001353301

Function1 in Bisection method in [2.2,3]
iteration 1 : f(2.600000000000) = -0.064000000000
iteration 2 : f(2.800000000000) = 0.792000000000
iteration 3 : f(2.700000000000) = 0.323000000000
iteration 4 : f(2.650000000000) = 0.119625000000
iteration 5 : f(2.625000000000) = 0.025390625000
iteration 6 : f(2.612500000000) = -0.019904296875
iteration 7 : f(2.618750000000) = 0.002592529297
iteration 8 : f(2.615625000000) = -0.008693450928
iteration 9 : f(2.617187500000) = -0.003059864044
iteration 10 : f(2.617968750000) = -0.000236019611
iteration 11 : f(2.618359375000) = 0.001177666605
iteration 12 : f(2.618164062500) = 0.000470676459
iteration 13 : f(2.618066406250) = 0.000117291667
iteration 14 : f(2.618017578125) = -0.000059373161
iteration 15 : f(2.618041992188) = 0.000028956956
iteration 16 : f(2.618029785156) = -0.000015208677
iteration 17 : f(2.618035888672) = 0.000006873996

Function1 in Bisection method in [2,2.8]
iteration 1 : f(2.400000000000) = -0.616000000000
iteration 2 : f(2.600000000000) = -0.064000000000
iteration 3 : f(2.700000000000) = 0.323000000000
iteration 4 : f(2.650000000000) = 0.119625000000
iteration 5 : f(2.625000000000) = 0.025390625000
iteration 6 : f(2.612500000000) = -0.019904296875
iteration 7 : f(2.618750000000) = 0.002592529297
iteration 8 : f(2.615625000000) = -0.008693450928
iteration 9 : f(2.617187500000) = -0.003059864044
iteration 10 : f(2.617968750000) = -0.000236019611
iteration 11 : f(2.618359375000) = 0.001177666605
iteration 12 : f(2.618164062500) = 0.000470676459
iteration 13 : f(2.618066406250) = 0.000117291667
iteration 14 : f(2.618017578125) = -0.000059373161
iteration 15 : f(2.618041992187) = 0.000028956956
iteration 16 : f(2.618029785156) = -0.000015208677
iteration 17 : f(2.618035888672) = 0.000006873996

Function2 in Bisection method in [2,3]
iteration 1 : f(2.500000000000) = -0.299844768317
iteration 2 : f(2.250000000000) = 0.223165248351
iteration 3 : f(2.375000000000) = -0.039880755668
iteration 4 : f(2.312500000000) = 0.092542655874
iteration 5 : f(2.343750000000) = 0.026395343550
iteration 6 : f(2.359375000000) = -0.006746823284
iteration 7 : f(2.351562500000) = 0.009825759345
iteration 8 : f(2.355468750000) = 0.001539526758
iteration 9 : f(2.357421875000) = -0.002603673093
iteration 10 : f(2.356445312500) = -0.000532074436
iteration 11 : f(2.355957031250) = 0.000503726461
iteration 12 : f(2.356201171875) = -0.000014173989
iteration 13 : f(2.356079101563) = 0.000244776245
iteration 14 : f(2.356140136719) = 0.000115301129
iteration 15 : f(2.356170654297) = 0.000050563570
iteration 16 : f(2.356185913086) = 0.000018194790
iteration 17 : f(2.356193542480) = 0.000002010400

Function2 in Bisection method in [2.2,3]
iteration 1 : f(2.600000000000) = -0.492187517364
iteration 2 : f(2.400000000000) = -0.092777038572
iteration 3 : f(2.300000000000) = 0.118893227107
iteration 4 : f(2.350000000000) = 0.013140077883
iteration 5 : f(2.375000000000) = -0.039880755668
iteration 6 : f(2.362500000000) = -0.013375563053
iteration 7 : f(2.356250000000) = -0.000117754084
iteration 8 : f(2.353125000000) = 0.006511320866
iteration 9 : f(2.354687500000) = 0.003196802903
iteration 10 : f(2.355468750000) = 0.001539526758
iteration 11 : f(2.355859375000) = 0.000710886608
iteration 12 : f(2.356054687500) = 0.000296566291
iteration 13 : f(2.356152343750) = 0.000089406105
iteration 14 : f(2.356201171875) = -0.000014173989
iteration 15 : f(2.356176757813) = 0.000037616058
iteration 16 : f(2.356188964844) = 0.000011721034
iteration 17 : f(2.356195068359) = -0.000001226477

Function2 in Bisection method in [2,2.8]
iteration 1 : f(2.400000000000) = -0.092777038572
iteration 2 : f(2.200000000000) = 0.324669272034
iteration 3 : f(2.300000000000) = 0.118893227107
iteration 4 : f(2.350000000000) = 0.013140077883
iteration 5 : f(2.375000000000) = -0.039880755668
iteration 6 : f(2.362500000000) = -0.013375563053
iteration 7 : f(2.356250000000) = -0.000117754084
iteration 8 : f(2.353125000000) = 0.006511320866
iteration 9 : f(2.354687500000) = 0.003196802903
iteration 10 : f(2.355468750000) = 0.001539526758
iteration 11 : f(2.355859375000) = 0.000710886608
iteration 12 : f(2.356054687500) = 0.000296566291
iteration 13 : f(2.356152343750) = 0.000089406105
iteration 14 : f(2.356201171875) = -0.000014173989
iteration 15 : f(2.356176757812) = 0.000037616058
iteration 16 : f(2.356188964844) = 0.000011721034
iteration 17 : f(2.356195068359) = -0.000001226477

Function1 in Newton's method with initial value 2.25
iteration 1 : f(2.973684210526) = 1.819233853331
iteration 2 : f(2.703724937665) = 0.338962423673
iteration 3 : f(2.624907210346) = 0.025049946351
iteration 4 : f(2.618083761004) = 0.000180087254
iteration 5 : f(2.618033991389) = 0.000000009547
iteration 6 : f(2.618033988750) = 0.000000000000

Function1 in Newton's method with initial value 2.5
iteration 1 : f(2.636363636364) = 0.067618332081
iteration 2 : f(2.618381618382) = 0.001258201619
iteration 3 : f(2.618034117409) = 0.000000465493
iteration 4 : f(2.618033988750) = 0.000000000000

Function1 in Newton's method with initial value 2.75
iteration 1 : f(2.633333333333) = 0.056259259259
iteration 2 : f(2.618277331747) = 0.000880651474
iteration 3 : f(2.618034051805) = 0.000000228134
iteration 4 : f(2.618033988750) = 0.000000000000

Function2 in Newton's method with initial value 2.25
iteration 1 : f(2.358228990429) = -0.004315811855
iteration 2 : f(2.356194476157) = 0.000000029774
iteration 3 : f(2.356194490192) = 0.000000000000

Function2 in Newton's method with initial value 2.5
```

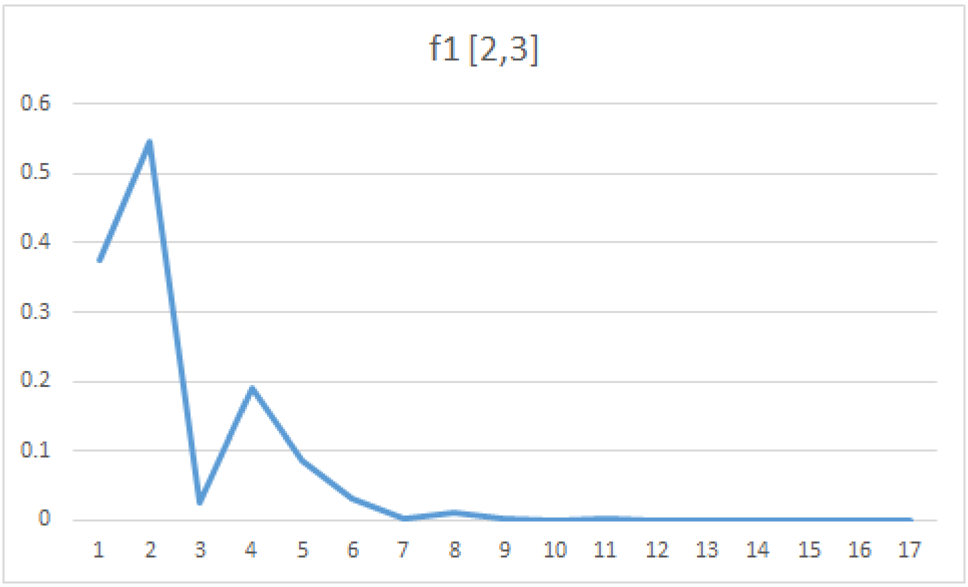
```
iteration 1 : f(2.351059749501) = 0.010892190568
iteration 2 : f(2.356194715836) = -0.000000478663
iteration 3 : f(2.356194490192) = 0.000000000000

Function2 in Newton's method with initial value 2.75
iteration 1 : f(2.218879782885) = 0.286746773861
iteration 2 : f(2.360650728280) = -0.009452952079
iteration 3 : f(2.356194342700) = 0.000000312878
iteration 4 : f(2.356194490192) = 0.000000000000
```

结果分析：

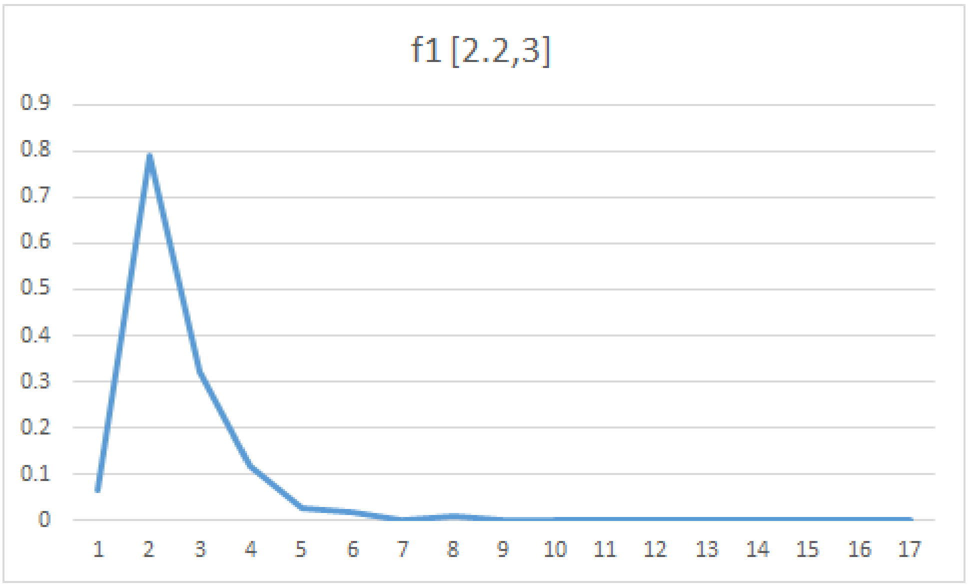
二分法：

1、f(x) = (x - 1)^3 - x ^2 + x

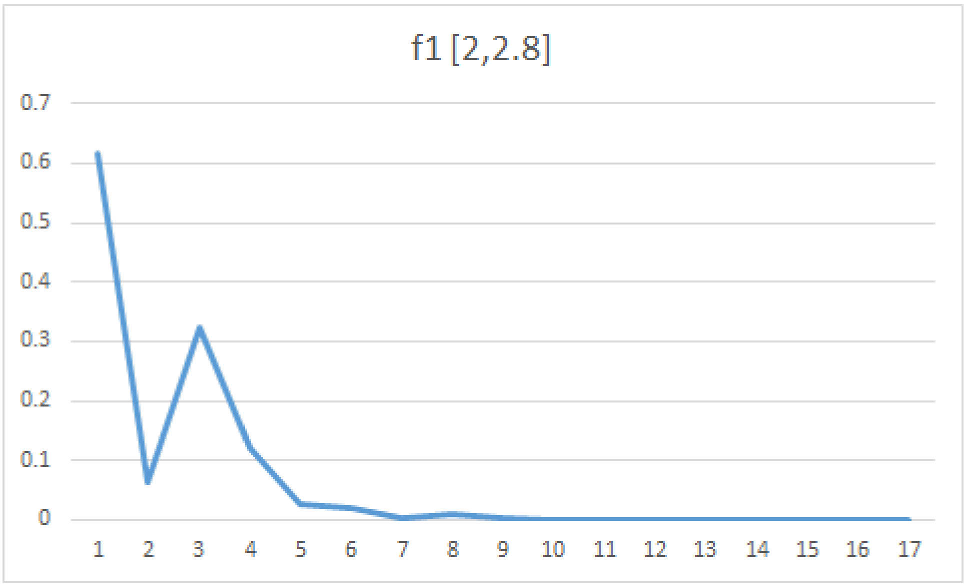
区间为[2,3]时



区间为[2.2,3]时



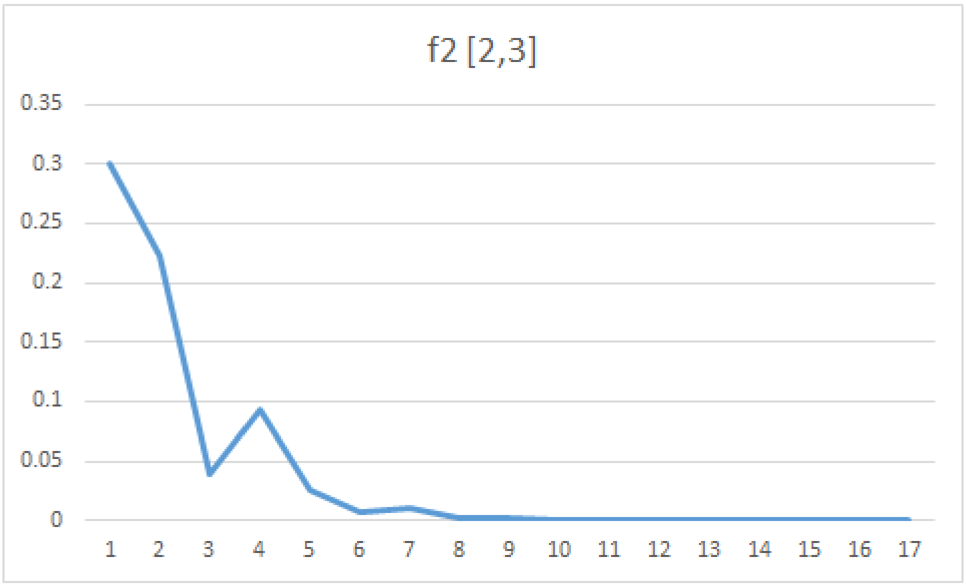
区间为[2,2.8]时



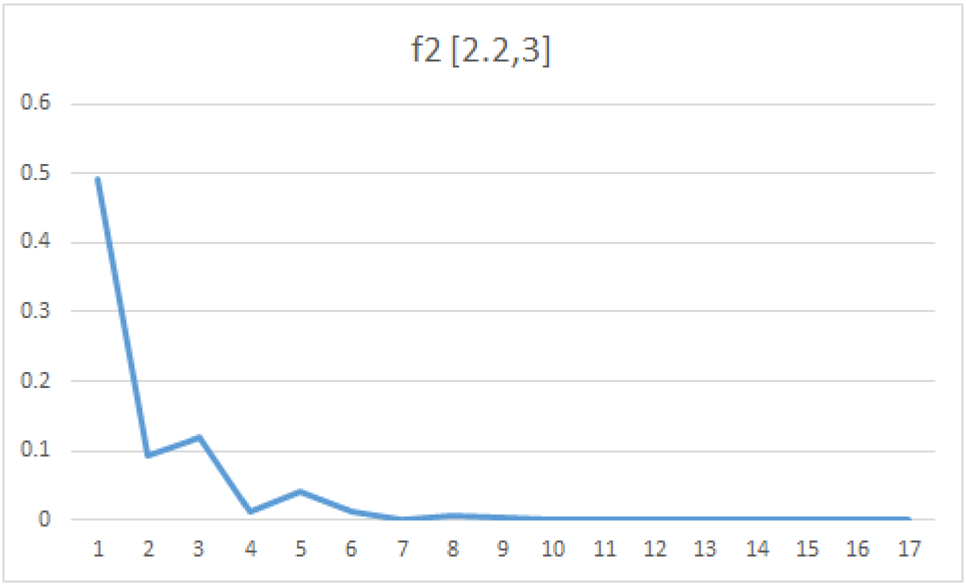
在该函数中用二分法求根时，不同的区间组合迭代次数都是大约在17次达到既定的精度，收敛速度在迭代次数增加时逐渐降低，在9~17次的迭代中，|f(x)|都在0.01以下，并逐渐趋近于0

2、f(x) = (sin(x)) ^ 3 + (cos(x)) ^ 3

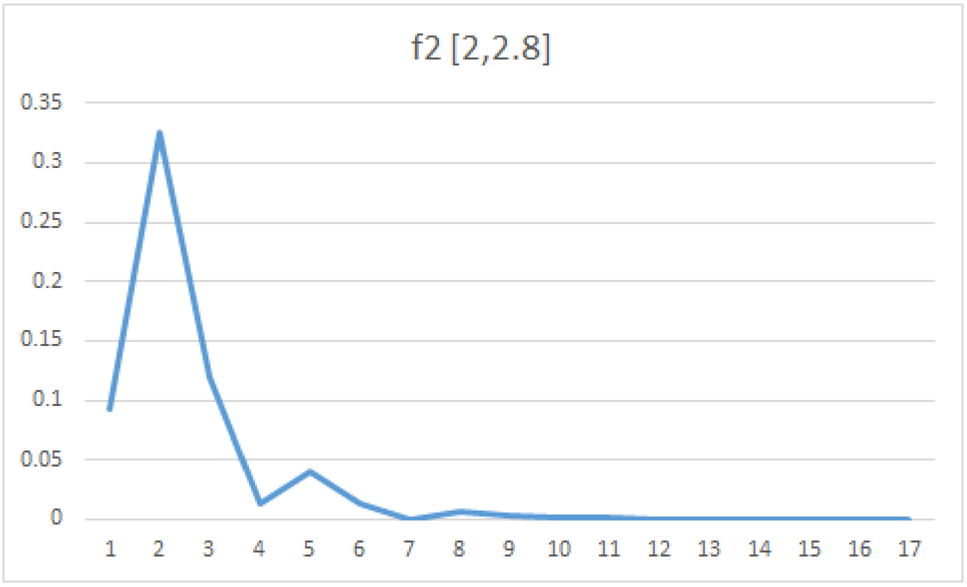
区间为[2,3]时



区间为[2.2,3]时



区间为[2,2.8]时

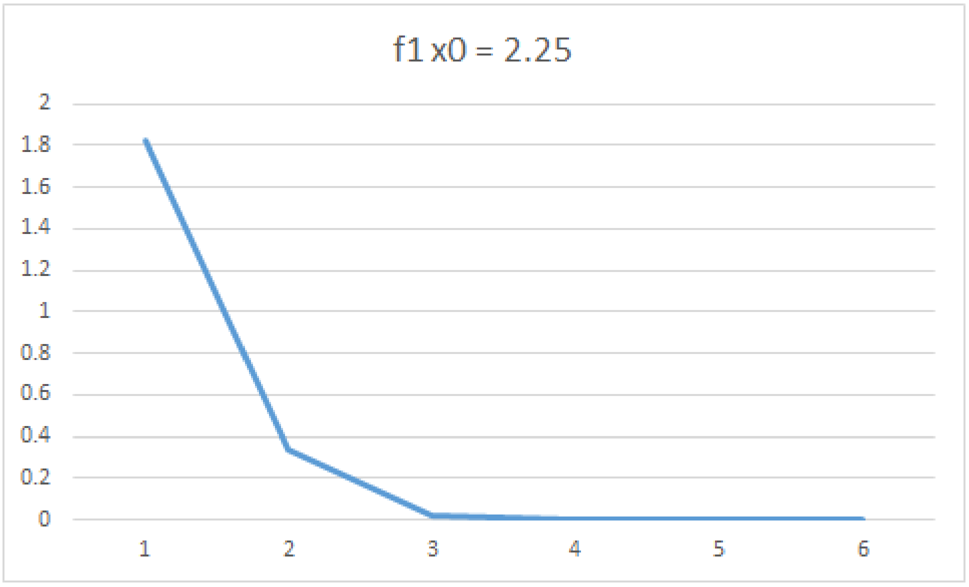


在该函数中用二分法求根时，不同的区间组合迭代次数都是大约在17次达到既定的精度，收敛速度在迭代次数增加时逐渐降低，在10~17次的迭代中，|f(x)|都在0.01以下，并逐渐趋近于0

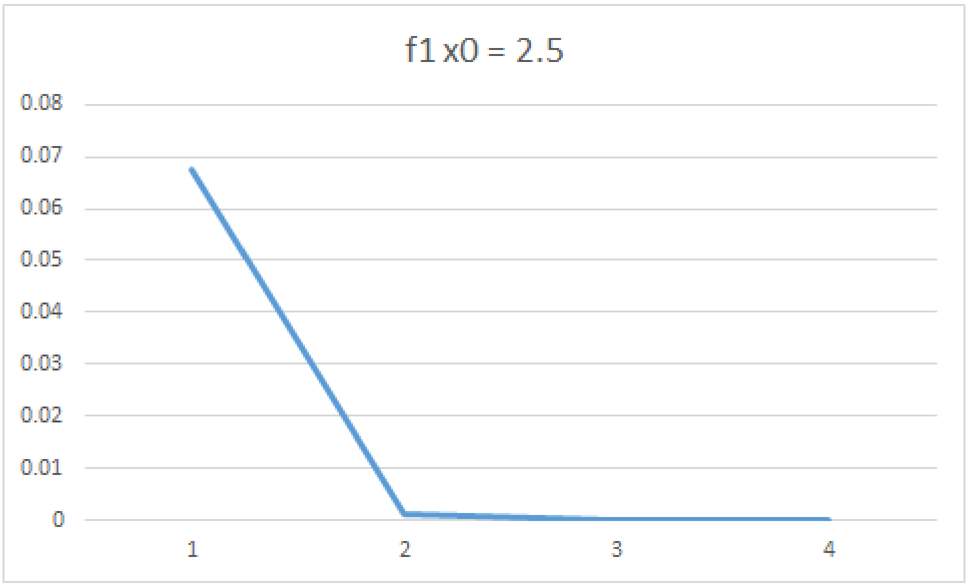
牛顿法：

1、f(x) = (x - 1)^3 - x ^2 + x

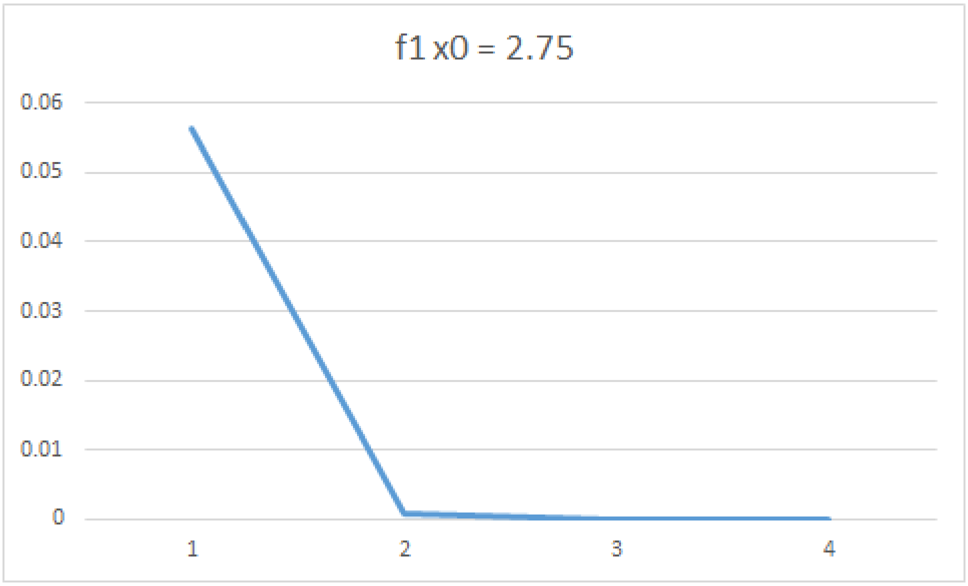
当初值x0 = 2.25时



当初值x0 = 2.5时



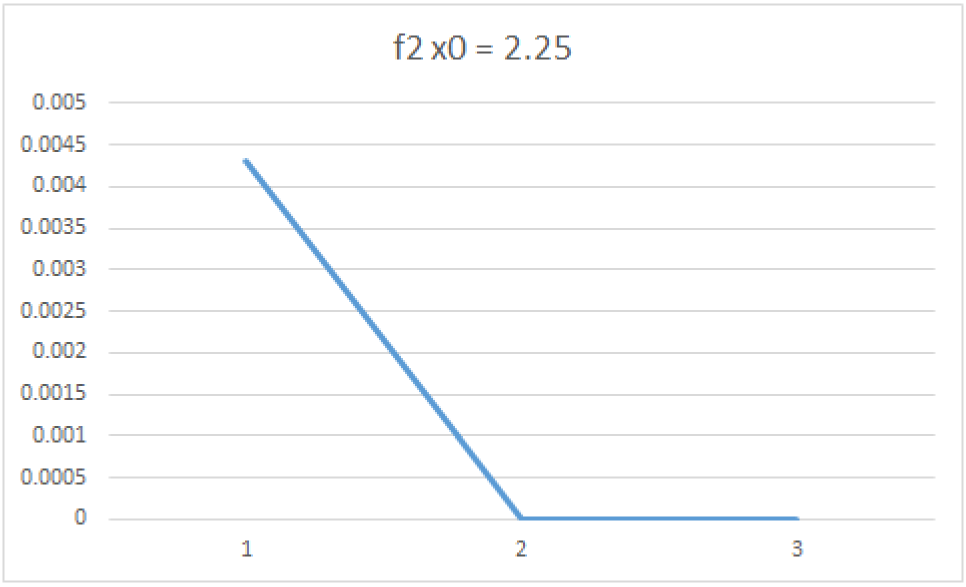
当初值x0 = 2.75时



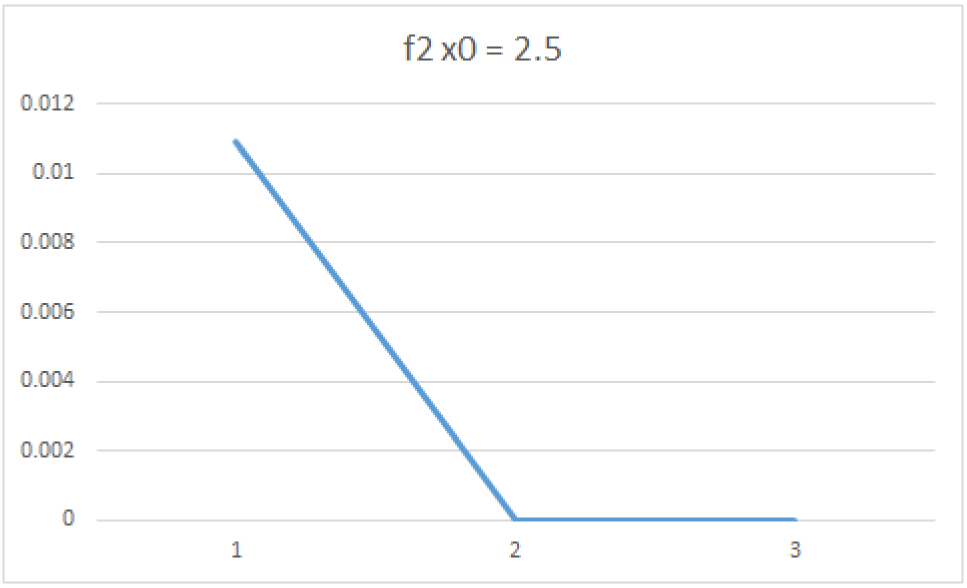
可以看出，该函数牛顿法的迭代次数相较二分法较少，大多数情况在第2或3次迭代时差不多|f(x)|达到0.01，而取的x0越接近实际根的值，迭代次数越少。

2、f(x) = (sin(x)) ^ 3 + (cos(x)) ^ 3

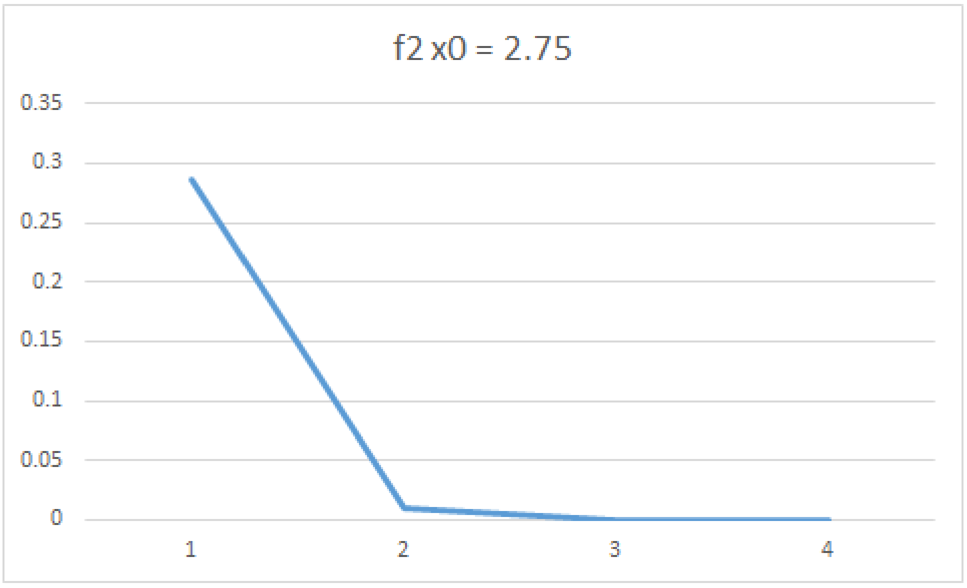
当初值x0 = 2.25时



当初值x0 = 2.5时



当初值x0 = 2.75时



可以看出，该函数牛顿法的迭代次数相较二分法较少，大多数情况在第2或3次迭代时差不多|f(x)|达到0.01，而取的x0越接近实际根的值，迭代次数越少。

结论：牛顿法普遍相较于二分法迭代次数较多，且牛顿法迭代得到的近似根更稳定使f(x)趋于0