数值分析第四次实验

16020041057 吴敬宇　　16090022049 曾趸

一． 实验名称：利用二分法和牛顿公式求解方程的根

二． 实验目的：

a. 验证牛顿公式的局部收敛性；

b. 比较二分法与牛顿公式的收敛速度；

c. 验证求解结果的正确性。

三． 实验内容及关键语句描写

本次实验要求使用牛顿法及二分法求解方程的根。

分别编写函数 Binary(min, max, times) 和 Newton(x0, times)实现以上两种方法。其中 Binary 函数为二分法函数，Newton 函数为牛顿法。下图为 Binary 函数：

```
void Binary(double min,double max,double precis)
{

    double mid = 0;
    int i = 0;
    cout<<"二分法迭代如下："<<endl;
    cout<<" 次数      值"<<endl;

    for(i=1;precis<fabs((max-min));i++)
    {
        mid = (min + max)/2.0;
        if(f(min)*f(mid)<0)
            max = mid;
        else if(f(mid) == 0.00)
            break;
        else
            min = mid;
        cout<<setw(2)<<i<<""<<setw(10)<<mid<<endl;

    }
    cout<<"总共进行了"<<i<<"次迭代。"<<endl;
}
```

二分法需要一个求解区间，那么我们让 **min 为区间下限，max 为区间上限，设置的精度为 0.0001.**

二分原理就是对给定的方程 f(x)=0，在二分区间(min,max)上进行如下处理:

a. 选择 min 和 max 的中点 mid；

b. 判断 f(mid)是否为 0，如果是就返回 mid；

c. 判断 f(min)*f(mid)<0，若 true，则 max = mid；

d. 否则 min = mid；

e. 判断此时 max-min 是否小于精度，若小于则返回 mid；

f. 此外在函数中循环一次就将 mid 输出一次，并且计数器+1；

牛顿思路如下：

```
void Newton(double x0,double precis)
{
    int flag;
    double x1 = 0;
    int k = 1;
    cout<<"牛顿法迭代如下："<<endl;
    cout<<" 次数     值"<<endl;
    while(1)
    {

        if(p(x0) == 0)
        {
            flag = 1;
            break;
        }
        x1 = x0 - f(x0)/p(x0);
        if(fabs(x1-x0)<precis)
            break;
        else
        {
            cout<<setw(2)<<k<<""<<setw(10)<<x0<<endl;
        }
        k++;
        x0 = x1;
        if(k>1000)  break;

    }
    if(k<1000)
        cout<<"总共进行了"<<k-1<<"次迭代。"<<endl;
    else
        cout<<"牛顿序列发散！"<<endl;
}
```

函数参数：**x0 表示给定的初值，times 表示迭代次数；**

而牛顿迭代的原理就是 $$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

把已知的初始值 x0 带入等式的右边，得到 x1。然后再把 x1 带入右端得到 x2，由于牛顿迭代公式的局部收敛性，当带入一个合适的初始值时，如此下去便能求得精度很高的近似值。

如下为 main()主函数：

```
int main()
{
    cout<<"请输入函数的区间下限";
    cin>>min;
    cout<<endl;
    cout<<"请输入函数的区间上限";
    cin>>max;
    cout<<endl;
    cout<<"请设置精度";
    cin>>precis;
    cout<<endl;
    Binary(min,max,precis);
    cout<<endl;
    cout<<"请输入牛顿的初始值：";
    cin>>x0;
    Newton(x0,precis);
    return 0;
}
```

四． 实验结果

首先先用两种方法计算 y = x*x-2 的函数，因为答案较好得出约等于 1.414，我们接下来计算一下：

通过下图我们可以发现，利用二分方法的话，需要计算 18 次才能达到精度为 0.0001 的答案。答案 1.41424 是接近正确答案的，所以正确性可以验证。接下来我们再测验一下牛顿迭代法。

我们发现，使用牛顿迭代法，选择初值为 7，只需要迭代 5 步就可以的到最后的答案，并且答案的精度也与二分法相差无几。

```
请输入函数的区间下限0

请输入函数的区间上限7

请设置精度0.0001

二分法迭代如下：
 次数      值
 1        3.5
 2       1.75
 3      0.875
 4     1.3125
 5    1.53125
 6    1.42188
 7    1.36719
 8    1.39453
 9     1.4082
10    1.41504
11    1.41162
12    1.41333
13    1.41418
14    1.41461
15     1.4144
16    1.41429
17    1.41424
总共进行了18次迭代。
```

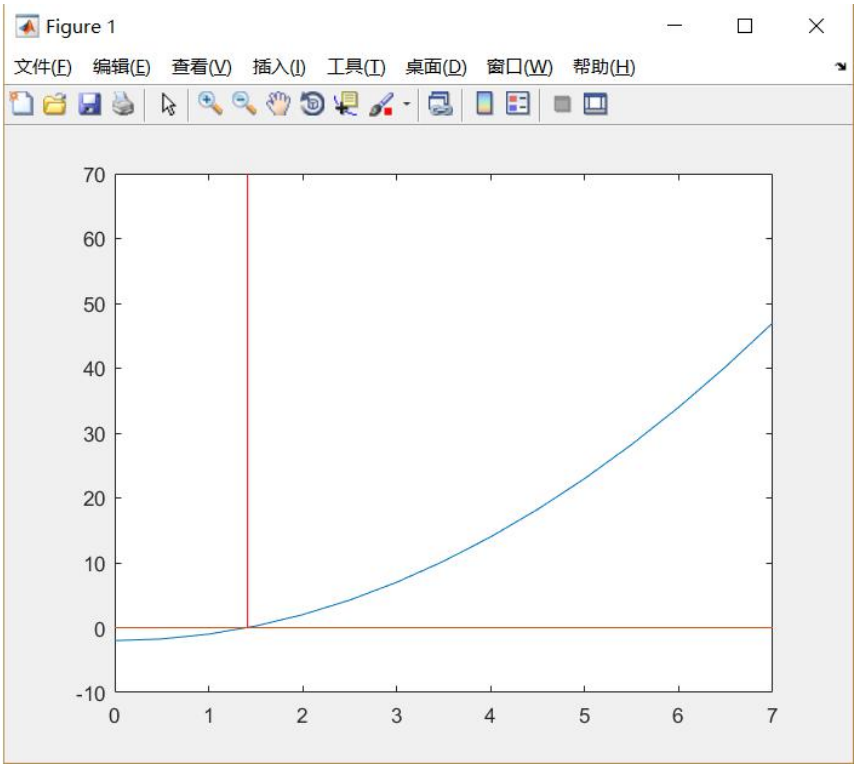
```
请输入牛顿的初始值：7
牛顿法迭代如下：
 次数      值
 1         7
 2   3.64286
 3   2.09594
 4   1.52508
 5   1.41824
总共进行了5次迭代。
```

我们再用 MATLAB 来验证一下结果是否正确：

```
>> syms x;
>> x = fsolve('x.*x-2',[0 7])

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>


x =

    1.4142    1.4142
```

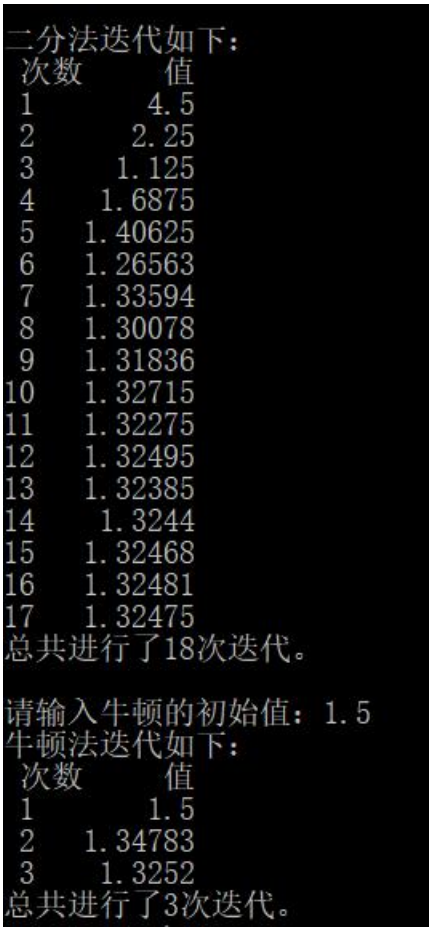
再用图像验证一次可以发现：

1.4142 这条线和 y=0 基本保持一致，可以确认，对于此函数的计算，二分法和牛顿法的计算结果均比较精准。



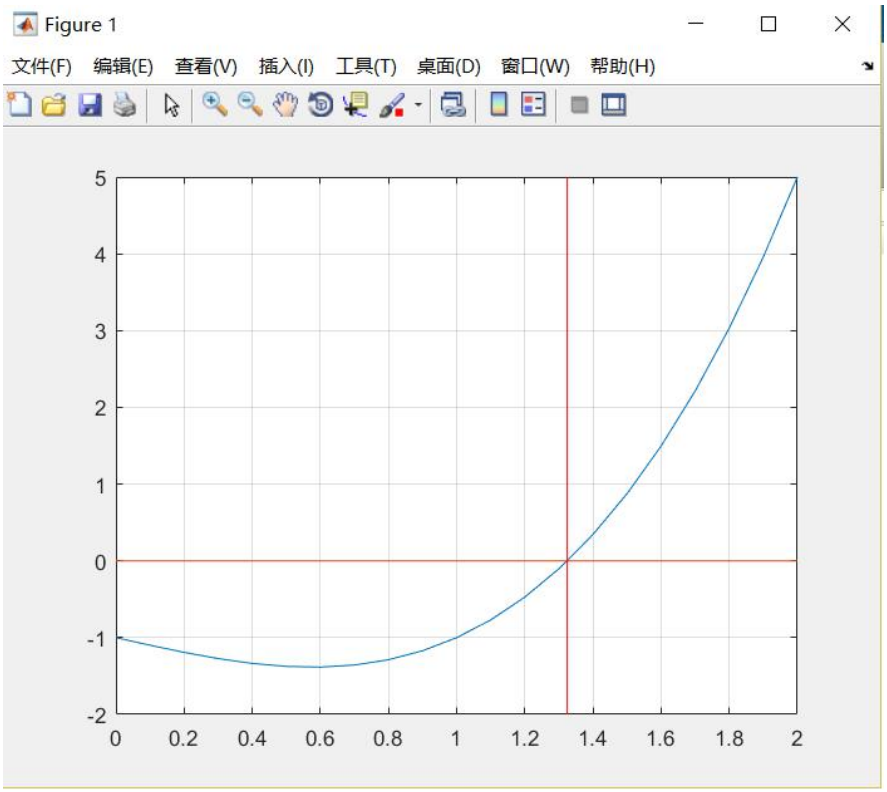
接下来我们再换一个例子，让 y = x*x*x-x-1

我们看一下两个方法的迭代次数以及精度。从下图可以发现，牛顿方法只需要三步就可以求出相应的解，但是二分法却需要 18 步，这两个例子都体现了牛顿方法迭代次数和精度均由于二分法。



让我们继续用 MATLAB 来验证一下：

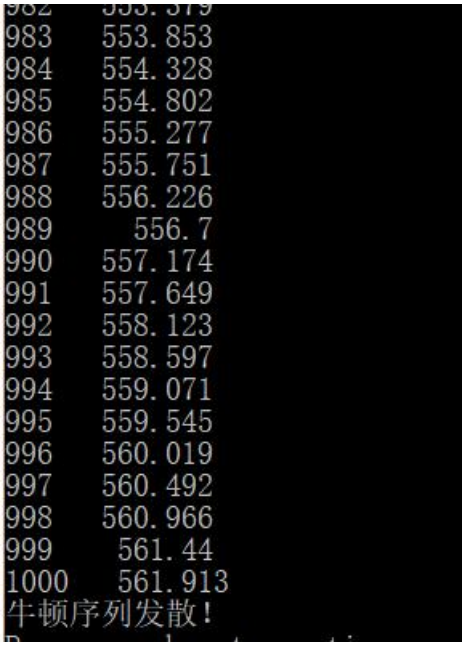
```
>> syms x
>> x = fsolve('x.*x.*x-x-1',[0 5])

No solution found.

fsolve stopped because the problem appears regular as measured by the gradient,
but the vector of function values is not near zero as measured by the
default value of the function tolerance.

<stopping criteria details>


x =

   -0.5774    1.3247
```



通过这两张图我们也可以发现，答案在精度范围内是正确的，验证了编写程序的正确性和准确性。

由此，编写程序的二分法以及牛顿法的正确性可以确定为正确，接着由实验结果可以得出，在相同情况下，牛顿的迭代次数和精度均高于二分法。实验目的的 b 和 c 得出结论。

又由于牛顿方法在特定情况下具有局部收敛性，即得到的数列是发散的。那么接下来我们选择如下的情况来展示一下:



我们可以发现，在进行了 1000 次迭代后，牛的你序列仍然没有得到精度为 0.0001 的答案。所以我们可以近似确定，牛顿序列是发散的，也就验证了它的局部收敛性。

五． 心得体会

本次实验通过编写关于牛顿迭代以及二分法求函数的根的程序，让我对于这两种求根的方法的过程有了更加深刻的印象，对于这两个方法的优劣势有了更加直观的印象，对于以后的使用有了更多的思路和方法。

在编写程序的时候，对于如何验证牛顿公式的收敛性有很大的难度，最后在同学们的合作之下，通过公式的推导求出来了相应的函数，完成整个实验。我也深刻的体会到合作的重要性，以及为什么老师让我们两个人一组而不是单人单组。相信在以后的学习过程中我一定会更加注重团队的合作，让整个实验变得更加流畅。