上机题第二题实验报告

王伟任　　计 33　　2013011333

一、题目要求及分析

第二章上机题 2：编程实现阻尼牛顿法，要求：（1）设定阻尼因子的初始值 λ_0 及解的误差阈值ε ；（2）阻尼因子 λ 用逐次折半法更新；（3）打印每个迭代步的最终 λ 值及近似解。用所编程序求解：（1）$x^3 - x - 1 = 0$ ，取$x_0 = 0.6$ ；（2）$-x^3 + 5x = 0$ ，取$x_0 = 1.2$ 。

牛顿迭代法使用近似解处函数值的切线与横轴的交点横坐标作为新的近似解，以此来不断逼近准确解。其近似解的迭代公式为 $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ 。而当初始值x_0偏离准确解较远时，这样的算法有可能发散，因此需要引入阻尼因子 λ 来缩小解的改变量，即 $x_{k+1} = x_k - \lambda\frac{f(x_k)}{f'(x_k)}$ ，并且要求新的近似解的函数值更加接近 0，否则将 λ 减小重新计算新的近似解。这就是阻尼牛顿法。

二、实验结果及分析

实验中需要设定阻尼因子的初始值λ_0和解的误差阈值ε。由阻尼牛顿法的要求，阻尼因子应该较为接近 1，因此初始值设置为 0.9；误差阈值设为10^{-10}，当函数值的绝对值小于该阈值时，即可认为该近似解已经足够接近准确解，可以当做答案。

第一个方程的解：

The equation is : X^3-X-1=0
e = 0.0000000001; l = 0.900000
current ans = 1.57312500000000188294; l = 0.056250
current ans = 1.36766295241534008298; l = 0.056250
current ans = 1.32634168448867439949; l = 0.056250
current ans = 1.32472040873855290144; l = 0.056250
current ans = 1.32471795725034646729; l = 0.056250
ans = 1.32471795725034646729

第二个方程的解：

The equation is : -X^3+5X=0
e = 0.0000000001; l = 0.900000
current ans = -1.62705882352940922608; l = 0.450000
current ans = -1.91981873621319465428; l = 0.225000
current ans = -2.33638857170398805962; l = 0.225000
current ans = -2.24218008415447211945; l = 0.225000
current ans = -2.23609287909452136844; l = 0.225000
current ans = -2.23606797791574729573; l = 0.225000
current ans = -2.23606797749978980505; l = 0.225000
ans = -2.23606797749978980505

由实验结果可以看出，阻尼牛顿法起作用的关键点就在由初始解确定第一个近似解的过程，因为初始解往往有可能偏离准确解较远，因此需要借助阻尼牛顿法来对这一步骤做一定的限制。而之后的阶段就基本满足了收敛的条件，不再需要减小阻尼因子甚至不需要阻尼因

子的帮助了。

三、实验代码

采用 C++语言实现。

```
#include <cstdio>
#include <cmath>

using namespace std;

const double e = 0.0000000001;
double l = 0.9 * 2;       //在这里乘 2 只是为了之后迭代的时候方便
int a3, a2, a1, a0;

void print(){          //为了更加直观在输出窗口输出原方程
      printf("The equation is : ");
      if ((a3 != 1)&&(a3 != -1)&&(a3 != 0)) printf("%dX^3", a3);
      else if (a3 == 1) printf("X^3");
      else if (a3 == -1) printf("-X^3");
      if ((a2 > 0)&&(abs(a3)>0)) printf("+");
      if ((a2 != 1)&&(a2 != -1)&&(a2 != 0)) printf("%dX^2", a2);
      else if (a2 == 1) printf("X^2");
      else if (a2 == -1) printf("-X^2");
      if ((a1 > 0)&&(abs(a3)+abs(a2)>0)) printf("+");
      if ((a1 != 1)&&(a1 != -1)&&(a1 != 0)) printf("%dX", a1);
      else if (a1 == 1) printf("X");
      else if (a1 == -1) printf("-X");
      if ((a0 > 0)&&(abs(a3)+abs(a2)+abs(a1)>0)) printf("+");
      if (a0 != 0) printf("%d", a0);
      printf("=0\n");
}

double function(double x){
      return a3*x*x*x+a2*x*x+a1*x+a0;
}

double dfunction(double x){
      return 3*a3*x*x+2*a2*x+a1;
}

int main(){
      printf("a3=1, a2=0, a1=-1, a0=-1, x0=0.6\n");
      printf("a3=-1, a2=0, a1=5, a0=0, x0=1.2\n");
      scanf("%d %d %d %d", &a3, &a2, &a1, &a0);
      print();
```

```
    printf("e = %.10f; l = %f\n", e, l / 2);
    double x0, x1, s;
    scanf("%lf", &x0);
    while (fabs(function(x0)) > e){
        s = function(x0) / dfunction(x0);
        x1 = x0 - s;
        while (fabs(function(x1)) >= fabs(function(x0))){
            l = l / 2;
            x1 = x0 - l * s;
        }
        x0 = x1;
        printf("current ans = %.20f; l = %f\n", x0, l);
    }
    printf("ans = %.20f\n", x0);
    return 0;
}
```