计算方法第五次上机作业程序文档

6.8.2021 yawning-lion

一、任务介绍

给定非线性方程组，给定迭代初值，设定精度，编制程序用牛顿法迭代求解。

二、公式说明

1、牛顿法

给定非线性方程组

$$\begin{cases} x_1^2+x_2^2+x_3^2-1=0 \\ 2x_1^2+x_2^2-4x_3=0 \\ 3x_1^2-4x_2+x_3^2=0 \end{cases}$$

记

$$\vec{x}=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\vec{f}(\vec{x})=\begin{bmatrix} x_1^2+x_2^2+x_3^2-1 \\ 2x_1^2+x_2^2-4x_3 \\ 3x_1^2-4x_2+x_3^2 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$f(x_1,x_2,x_3)$的雅可比矩阵

$$A(\vec{x})=\begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \\ 4x_1 & 2x_2 & -4 \\ 6x_3 & -4 & 2x_3 \end{bmatrix}$$

牛顿法迭代格式为

$$A(\vec{x}_0)\cdot(\vec{x}-\vec{x}_0)=-\vec{f}(\vec{x}_0)$$

解出$\vec{x}$作为下一次计算迭代格式中的$\vec{x}_0$.

2、高斯消元法

牛顿法迭代格式中，需要求解一个常系数线性方程组，求解由高斯消元法实现。公式不予详细说明，具体实现请点击此处.

三、程序说明

1、运行环境

程序编译环境为mingw-w64-v8.0.0,g++,IDE为vscode，程序文档利用markdown写作。

2、使用说明

本程序没有输入，点击运行即可求解给定的非线性方程组。如果想要求解其他非线性方程组，需要在程序内部修改。根据非线性方程组个数，方程内未知数个数修改，程序还可继续使用。

本程序输出为直接打印结果。分别打印近似解和迭代次数。

3、程序结构

本程序包含三个头文件

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

随后定义设定精度

```
# define EPS 0.0000001
```

定义用于生成雅可比矩阵的函数`RenewJ`，用于生成牛顿法中方程右端项的函数`RenewB`,定义用于求解迭代格式的函数`RenewX`,定义用于判断两次迭代结果是否在设定误差范围内的函数`ErrorCheck`。

然后进入主函数。分别定义雅可比矩阵维度`n=3`,`m=3`，雅可比矩阵`jocA`，方程解`arrX`，方程右端项`arrB`,存储上一迭代解的数组`lastX`，迭代次数`k`。

进入`while`循环，将当前的`arrX`值赋给`lastX`，然后分别用目前的迭代结果`arrX`来更新雅可比矩阵`jocA`，方程解`arrX`，方程右端项`arrB`，然后进入函数`RenewX`求解，进入函数`ErrorCheck`判别精度是否满足要求后，进入下一轮循环。

精度满足要求后，输出`arrX`和迭代次数`k`。

4、函数说明

`int RenewJ(double* jocA,double* arrX,int n,int m)`

利用手动寻址，对矩阵内每一个元素赋值。直接对内存操作，返回0无意义。

```
for(int i=0;i<m;i++) *(jocA+i)=2*arrX[i];
*(jocA+1*n+0)=4*arrX[0];
*(jocA+1*n+1)=2*arrX[1];
*(jocA+1*n+2)=-4;
*(jocA+2*n+0)=6*arrX[0];
*(jocA+2*n+1)=-4;
*(jocA+2*n+2)=2*arrX[2];
```

`int RenewB(double* arrX,double* arrB,int n)`

对数组内每一个元素赋值。返回0无意义。

```
arrB[0]=-(x1*x1+x2*x2+x3*x3-1);
arrB[1]=-(2*x1*x1+x2*x2-4*x3);
arrB[2]=-(3*x1*x1-4*x2+x3*x3);
```

`int RenewX(double* jocA,double* arrB,double* arrX,int n,int m)`

给定雅可比矩阵右端项后，利用 高斯消元法求解。

下为对增广矩阵做行变换，得到上三角形部分的代码

```
double w=0;
for(int i=0;i<n-1;i++)
{
    for(int j=i+1;j<n;j++)
    {
        w=(*(jocA+n*j+i))/(*(jocA+n*i+i));
        for(int k=i;k<m;k++)
        {
            *(jocA+n*j+k)=*(jocA+n*j+k)-*(jocA+n*i+k)*w;
        }
        arrB[j]=arrB[j]-arrB[i]*w;
    }
}
```

下为回代，求解出$\vec{x}-\vec{x}_0$部分的代码

```
double deltX[n];
for(int i=n-1;i>-1;i--)
{
    double temp=0;
    for(int j=i+1;j<n;j++)
    {
        temp+=*(jocA+n*i+j)*deltX[j];
    }
    deltX[i]=(arrB[i]-temp)/ *(jocA+n*i+i);
}
```

随后利用得到的解更新`arrX`为新的迭代结果。

```
for(int i=0;i<n;i++) arrX[i]+=deltX[i];
```

`int ErrorCheck(int n,double* lastX,double*arrX )`

在无穷范数的意义下计算两次迭代解误差，若精度满足要求返回1，反之返回0.

四、算例展示

在给定非线性方程组，设定精度的情况下，得到近似解

```
PS C:\Users\任大代表> cd "c:\Users\任大代表\Desktop\算方上机作业\1900011026
任绍持作业五\code\" ; if ($?) { g++ NCM_HW5.cpp -o NCM_HW5 } ; if ($?) { .\N
CM_HW5 }
solution=
0.7851969 0.4966114 0.3699228
```

保留七位小数，近似解

$$\vec{x} \approx \begin{bmatrix} 0.7851969 \\ 0.4966114 \\ 0.3699228 \end{bmatrix}$$

五、总结反思

可以看到本题中所给非线性方程组具有很好的性质，迭代5次就得到了近似解。如果所给矩阵性质不好需要迭代多次，本程序中对雅可比矩阵的精确更新可能就会造成时间消耗过大。

本程序的优点在于矩阵维度`n`,`m`可以自行赋值，为修改程序求解其他非线性方程组带来了方便，缺点在于利用高斯消元法解方程不安全，由给定非线性方程组的特殊性质，没考虑主元变换。