计算方法Project3

> PB19051035周佳豪

问题描述

1. 使用 Newton 迭代法求解非线性方程组
$$\begin{cases} f(x) = x^2 + y^2 - 1 = 0 \\ g(x) = x^3 - y = 0 \end{cases}$$
   取 $(x_0, y_0) = (0.8, 0.6)$,误差控制 $max\{|\delta_x|, |\delta_y|\} <= 10^{-5}$

   > 输入：初始点(x_0, y_0),精度控制值 e ，定义函数 $f(x), g(x)$
   >
   > 输出：迭代次数 k ,第 k 步的迭代解 (x_k, y_k)

2. 使用二阶 Rouge-Kutta 公式求解常微分方程初值问题
$$\begin{cases} y^{'}(x) = ysin\pi x \\ y(0) = 1 \end{cases}$$

   > 输入：区间剖分点数 n ,区间端点 a, b，定义函数 $y^{'}(x) = f(x, y)$
   >
   > 输出：$y_k, k = 1, 2, \ldots, n$

3. 用改进的 Euler 公式求解常微分方程初值问题
$$\begin{pmatrix} \overline{y}_{n+1} \\ \overline{z}_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + h \begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix}$$
$$\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + \frac{h}{2} \left[\begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} + \begin{pmatrix} f(x_{n+1}, \overline{y}_{n+1}, \overline{z}_{n+1}) \\ g(x_{n+1}, \overline{y}_{n+1}, \overline{z}_{n+1}) \end{pmatrix} \right]$$

   > 输入：区间剖分点数 N ,区间端点 a, b，定义函数：
   > $$\begin{cases} u^{'}(t) = 0.09u(1 - \frac{u}{20}) - 0.45uv \\ v^{'}(t) = 0.06v(1 - \frac{v}{15}) - 0.001uv \\ u(0) = 1.6 \\ v(0) = 1.2 \end{cases}$$
   > 输出：(y_k, z_k)，$k = 1, 2, \ldots, N$

算法设计

1. 先根据线性方程组 $\begin{cases} \delta_x \frac{\partial f(x_k, y_k)}{\partial x} + \delta_y \frac{\partial f(x_k, y_k)}{\partial y} = -f(x_k, y_k) \\ \delta_x \frac{\partial g(x_k, y_k)}{\partial x} + \delta_y \frac{\partial g(x_k, y_k)}{\partial y} = -g(x_k, y_k) \end{cases}$ 求得 δ_x, δ_y，再根据公式 $x_{k+1} = x_k + \delta_x, y_{k+1} = y_k + \delta_y$ 即可求得 x_{k+1}, y_{k+1} ，重复这个过程直至 $max\{|\delta_x|, |\delta_y|\} < e$ ，其中 e 为精度控制值

   核心代码为：

```
while(1){
		k++;
		double val_f1_x = f1_x(x,y);
		double val_f1_y = f1_y(x,y);
		double val_f2_x = f2_x(x,y);
		double val_f2_y = f2_y(x,y);
```

```
            double val_f1 = f1(x,y);
            double val_f2 = f2(x,y);
            double delta_x = (val_f2/val_f2_y-
val_f1/val_f1_y)/(val_f1_x/val_f1_y-val_f2_x/val_f2_y);//根据公式计算delta_x
            double delta_y = -(val_f1-val_f1_x*delta_x)/val_f1_y;
            x += delta_x;
            y += delta_y;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
            if(fabs(delta_x)<error&&fabs(delta_y)<error)
                break;
        }
```

2. 根据公式 $\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + hk_1) \end{cases}$ 或 $\begin{cases} y_{n+1} = y_n + hk_2 \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \end{cases}$ 即可求得 y_k，$k = 1, 2, \dots, n$

   核心代码为：

```
//二阶RK公式
void Runge_Kutta1(){
        double h=(b-a)/n;
        int k=0;
        double x=this->x_0,y=this->y_0;
        while(k<n)
        {
            k++;
            double k1=_y(x,y);
            double k2 = _y(x+h,y+k1*h);
            y=y+h/2*(k1+k2);
            x=x+h;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
        }
    }
//中点公式
void Runge_Kutta2(){
        double h=(b-a)/n;
        int k=0;
        double x=this->x_0,y=this->y_0;
        while(k<n)
        {
            k++;
            double k1=_y(x,y);
            double k2 = _y(x+h/2,y+k1*h/2);
            y=y+h*k2;
            x=x+h;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
        }
    }
```

3. 套用题目中的公式即可

   核心代码为：

```
void Euler(){
        double h=(b-a)/N;
        double x=a;
```

```
        double y=1.6;
        double z=1.2;
        for(int k=1;k<=N;k++)
        {
            double temp_y=y+h*_y(x,y,z);
            double temp_z=z+h*_z(x,y,z);
            y=y+h/2*(_y(x,y,z)+_y(x+h,temp_y,temp_z));
            z=z+h/2*(_z(x,y,z)+_z(x+h,temp_y,temp_z));
            x=x+h;
            cout<<"x="<<x<<" y="<<y<<" z="<<z<<endl;
        }
    }
```

实验结果

1.

```
k=1 x=0.827049 y=0.636066
k=2 x=0.827263 y=0.566705
k=3 x=0.826035 y=0.560044
k=4 x=0.826035 y=0.563629
k=5 x=0.826031 y=0.563615
k=6 x=0.826031 y=0.563624
```

2. 输入 $n = 20, a = 0, b = 2$

```
//二阶RK公式
k=1 x=0.1 y=1.01545
k=2 x=0.2 y=1.06191
k=3 x=0.3 y=1.13859
k=4 x=0.4 y=1.24318
k=5 x=0.5 y=1.37036
k=6 x=0.6 y=1.51056
k=7 x=0.7 y=1.64931
k=8 x=0.8 y=1.76842
k=9 x=0.9 y=1.84932
k=10 x=1 y=1.87789
k=11 x=1.1 y=1.84888
k=12 x=1.2 y=1.76765
k=13 x=1.3 y=1.6484
k=14 x=1.4 y=1.50968
k=15 x=1.5 y=1.36958
k=16 x=1.6 y=1.24249
k=17 x=1.7 y=1.13793
k=18 x=1.8 y=1.06116
k=19 x=1.9 y=1.01454
k=20 x=2 y=0.998865

//中点公式
k=1 x=0.1 y=1.01564
k=2 x=0.2 y=1.06247
k=3 x=0.3 y=1.1398
k=4 x=0.4 y=1.24547
k=5 x=0.5 y=1.37433
k=6 x=0.6 y=1.51686
k=7 x=0.7 y=1.65844
k=8 x=0.8 y=1.78045
k=9 x=0.9 y=1.86366
```

```
k=10 x=1 y=1.89326
k=11 x=1.1 y=1.86364
k=12 x=1.2 y=1.78034
k=13 x=1.3 y=1.65815
k=14 x=1.4 y=1.51639
k=15 x=1.5 y=1.37374
k=16 x=1.6 y=1.24484
k=17 x=1.7 y=1.1392
k=18 x=1.8 y=1.0619
k=19 x=1.9 y=1.01511
k=20 x=2 y=0.999475
```

3. 输入 $N = 20, a = 0, b = 2$

```
t=0.1 u=1.52832 v=1.20646
t=0.2 u=1.45948 v=1.21295
t=0.3 u=1.39337 v=1.21949
t=0.4 u=1.32991 v=1.22606
t=0.5 u=1.26899 v=1.23268
t=0.6 u=1.21053 v=1.23934
t=0.7 u=1.15445 v=1.24603
t=0.8 u=1.10067 v=1.25277
t=0.9 u=1.04909 v=1.25954
t=1 u=0.999652 v=1.26635
t=1.1 u=0.95227 v=1.27321
t=1.2 u=0.906872 v=1.2801
t=1.3 u=0.863388 v=1.28703
t=1.4 u=0.821748 v=1.294
t=1.5 u=0.781885 v=1.30101
t=1.6 u=0.743733 v=1.30806
t=1.7 u=0.70723 v=1.31515
t=1.8 u=0.672315 v=1.32228
t=1.9 u=0.638928 v=1.32944
t=2 u=0.607012 v=1.33665
```

结果分析

- Newton 迭代法是一种不错的求解非线性方程组的解法，运行时间较快
- R-K公式的局部截断误差为 $O(h^3)$，且其具体取值与 c_1, c_2 等常数的具体取值有关
- 改进的 $Euler$ 公式同样可以有效地求解常微分方程组的初值问题

附录

问题1

```
//问题1
#include<iostream>
#include<cmath>
using namespace std;

const double error = 1e-5;

double f(double x,double y) {
    return x*x + y*y - 1;
}
```

```
double f_x(double x,double y) {
    return 2*x;
}
double f_y(double x,double y) {
    return 2*y;
}

double g(double x,double y) {
    return x*x*x -y;
}
double g_x(double x,double y) {
    return 3*x*x;
}
double g_y(double x,double y) {
    return -1;
}


class solution{
private:
    double (*f1)(double,double);
    double (*f2)(double,double);
    double (*f1_x)(double,double);
    double (*f1_y)(double,double);
    double (*f2_x)(double,double);
    double (*f2_y)(double,double);
public:
    solution(){
        this->f1 = f;
        this->f2 = g;
        this->f1_x = f_x;
        this->f1_y = f_y;
        this->f2_x = g_x;
        this->f2_y = g_y;
    }
    solution(double (*f1)(double,double),double (*f2)(double,double),double
(*f1_x)(double,double),double (*f1_y)(double,double),double (*f2_x)
(double,double),double (*f2_y)
(double,double)):f1(f1),f2(f2),f1_x(f1_x),f1_y(f1_y),f2_x(f2_x),f2_y(f2_y){};
    void Newton(double x_0,double y_0,double e){
        double x=x_0;
        double y=y_0;
        int k=0;
        while(1){
            k++;
            double val_f1_x = f1_x(x,y);
            double val_f1_y = f1_y(x,y);
            double val_f2_x = f2_x(x,y);
            double val_f2_y = f2_y(x,y);
            double val_f1 = f1(x,y);
            double val_f2 = f2(x,y);
            double delta_x = (val_f2/val_f2_y-
val_f1/val_f1_y)/(val_f1_x/val_f1_y-val_f2_x/val_f2_y);
            double delta_y = -(val_f1-val_f1_x*delta_x)/val_f1_y;
            x += delta_x;
            y += delta_y;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
            if(fabs(delta_x)<error&&fabs(delta_y)<error)
```

```
                break;
            }
        }
};

int main(){
    solution s1;
    s1.Newton(0.8,0.6,error);
    return 0;
}
```

问题2

```
//问题2
#include<iostream>
#include<cmath>

using namespace std;
const double pi=3.14159265;

double f(double x,double y){
    return y*sin(pi*x);
}

class Solution{
private:
    double a,b;
    int n;
    double x_0,y_0;
    double (*_y)(double ,double );
public:
    Solution(int n,double a,double b,double x_0,double y_0,double(*f)
(double,double))
    {
        this->n = n;
        this->a = a;
        this->b = b;
        this->x_0 = x_0;
        this->y_0 = y_0;
        this->_y = f;
    }
    void Runge_Kutta2(){
        double h=(b-a)/n;
        int k=0;
        double x=this->x_0,y=this->y_0;
        while(k<n)
        {
            k++;
            double k1=_y(x,y);
            double k2 = _y(x+h/2,y+k1*h/2);
            y=y+h*k2;
            x=x+h;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
        }
    }
    void Runge_Kutta1(){
        double h=(b-a)/n;
```

```
        int k=0;
        double x=this->x_0,y=this->y_0;
        while(k<n)
        {
            k++;
            double k1=_y(x,y);
            double k2 = _y(x+h,y+k1*h);
            y=y+h/2*(k1+k2);
            x=x+h;
            cout<<"k="<<k<<" x="<<x<<" y="<<y<<endl;
        }
    }

};

int main(){
    Solution s(20,0,2,0,1,f);
    s.Runge_Kutta1();//测试二阶RK公式
    //s.Runge_Kutta2();//测试中点公式
    return 0;
}
```

问题3

```
//问题3
#include <iostream>

using namespace std;

double f(double x, double y,double z)
{
    return 0.09*y*(1-y/20)-0.45*y*z;
}
double g(double x, double y, double z)
{
    return 0.06*z*(1-z/15)-0.001*z*y;
}

class Solution
{
private:
    int N;
    double a, b;
    double (*_y)(double, double, double);
    double (*_z)(double, double, double);
public:
    Solution(int N,double a,double b,double(*f)(double,double,double),double(*g)
(double,double,double))
    {
        this->N = N;
        this->a = a;
        this->b = b;
        this->_y = f;
        this->_z = g;
    }
    void Euler(){
        double h=(b-a)/N;
```

```
        double x=a;
        double y=1.6;
        double z=1.2;
        for(int k=1;k<=N;k++)
        {
            double temp_y=y+h*_y(x,y,z);
            double temp_z=z+h*_z(x,y,z);
            y=y+h/2*(_y(x,y,z)+_y(x+h,temp_y,temp_z));
            z=z+h/2*(_z(x,y,z)+_z(x+h,temp_y,temp_z));
            x=x+h;
            cout<<"t="<<x<<" u="<<y<<" v="<<z<<endl;
        }
    }
};

int main()
{
    Solution s(20,0,2,f,g);
    s.Euler();
    return 0;
}
```