计算方法实验三

PB19030888张舒恒

2022 年 4 月 21 日

问题一

1.（Page186, Project10）用 Newton 迭代法求解非线性方程组

$$\begin{cases} f(x) = x^2 + y^2 - 1 = 0 \\ g(x) = x^3 - y = 0 \end{cases} \tag{1}$$

取 $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.6 \end{pmatrix}$，误差控制 $\max(|\Delta x_k|, |\Delta y_k|) \le 10^{-5}$.

输入：初始点 $(x_0, y_0) = (0.8, 0.6)$，精度控制 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).

算法过程

$$\begin{cases} \Delta x \dfrac{\partial f_1(x_0, y_0)}{\partial x} + \Delta y \dfrac{\partial f_1(x_0, y_0)}{\partial y} = -f_1(x_0, y_0) \\ \Delta x \dfrac{\partial f_2(x_0, y_0)}{\partial x} + \Delta y \dfrac{\partial f_2(x_0, y_0)}{\partial y} = -f_2(x_0, y_0) \end{cases} \tag{3.7}$$

如果

$$\det(J(x_0, y_0)) = \begin{vmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_1}{\partial y} \end{vmatrix}_{(x_0, y_0)} \neq 0$$

解出 $\Delta x, \Delta y$

$$w_1 = w_0 + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} x_0 + \Delta x \\ y_0 + \Delta y \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

再列出方程组

$$\begin{cases} \dfrac{\partial f(x_1, y_1)}{\partial x}(x - x_1) + \dfrac{\partial f(x_1, y_1)}{\partial y}(y - y_1) = -f(x_1, y_1) \\ \dfrac{\partial g(x_1, y_1)}{\partial x}(x - x_1) + \dfrac{\partial g(x_1, y_1)}{\partial y}(y - y_1) = -g(x_1, y_1) \end{cases}$$

解出

$$\Delta x = x - x_1, \quad \Delta y = y - y_1$$

$$w_2 = \begin{pmatrix} x_1 + \Delta x \\ y_1 + \Delta x \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

继续做下去, 每一次迭代都是解一个类似式 (3.7) 的方程组

$$J(x_k, y_k) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -f(x_k, y_k) \\ -g(x_k, y_k) \end{pmatrix}$$

$$\Delta x = x_{k+1} - x_k, \quad \Delta y = y_{k+1} - y_k$$

即

$$x_{k+1} = x_k + \Delta x, \quad y_{k+1} = y_k + \Delta y$$

直到 $\max(|\Delta x|, |\Delta y|) < \varepsilon$ 为止.

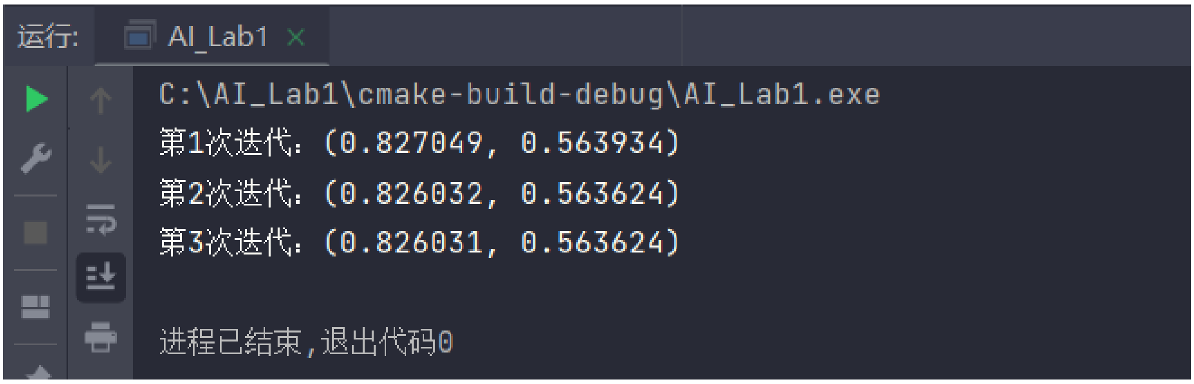
代码实现

计算delta_x，delta_y，将其加入x[i+1]，y[i+1]，输出当次迭代结果，若max(fabs(delta_x), fabs(delta_y)小于给定误差界限，则停止迭代，否则继续迭代。

```
while(true){
    delta_x = (-f(x[i], y[i]) - g(x[i], y[i]) * 2 * y[i]) / (2 * x[i] + 6.0 *
x[i] * x[i] * y[i]);
    delta_y = 3 * x[i] * x[i] * delta_x + g(x[i], y[i]);
    x[i+1] = x[i] + delta_x;
    y[i+1] = y[i] + delta_y;
    cout << "第" << i+1 << "次迭代: " << "(" << x[i+1] << ", " << y[i+1] << ")" <<
endl;
    i++;
    //cout << "max:" << max(delta_x, delta_y) << endl;
    if(max(fabs(delta_x), fabs(delta_y)) <= eps)
        break;
}
```

输出结果

在第3次迭代后满足精度要求，得出$x \approx 0.826031, y \approx 0.563624$



问题二

2.（Page187, Project21(1)）用二阶 Rouge-Kutta 公式求解常微分方程组初值问题

$$\begin{cases} y'(x) = f(x, y) \\ y(a) = y_0 \end{cases}, a \leq x \leq b \tag{2}$$

(1) 求解初值问题

$$\begin{cases} y'(x) = y\sin\pi x \\ y(0) = 1 \end{cases} \tag{3}$$

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，$k = 1, 2, ..., n$.

算法过程

Runge-Kutta 方法通常写成如下形式，

$$\begin{cases} y_{n+1} = y_n + h(c_1k_1 + c_2k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + ah, y_n + bhk_1) \end{cases} \tag{7.13}$$

若取 $c_1 = \frac{1}{2}$, $c_2 = \frac{1}{2}$, $a = 1$, $b = 1$, 得到式 (7.14) 的二阶 Runge-Kutta 公式：

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + hk_1) \end{cases} \tag{7.14}$$

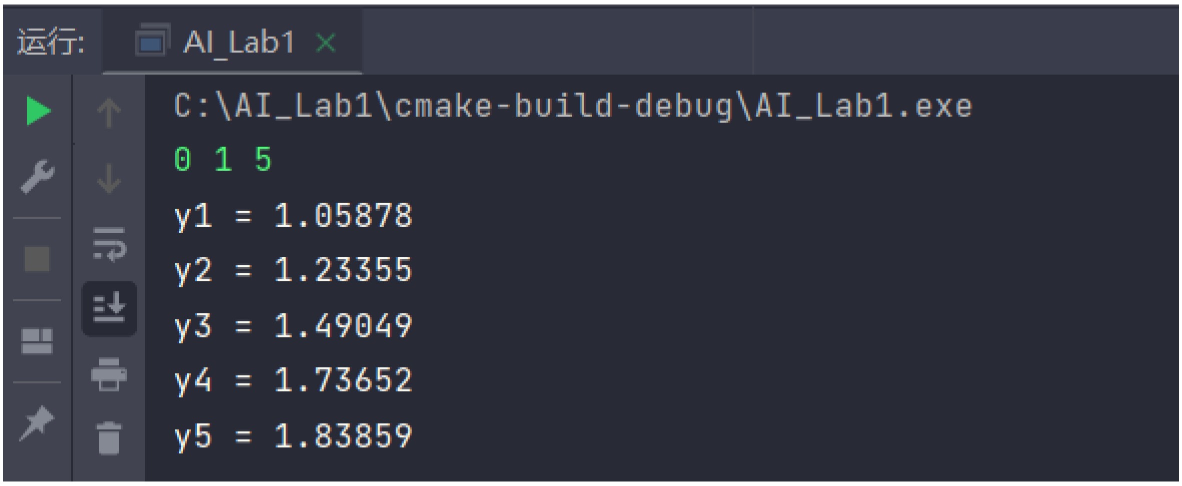
代码实现

计算k1，k2，y[i+1]，输出y[i+1]，若i < n则继续迭代，否则算法终止

```
for(int i = 0; i < n; i++){
    x[i] = i * h;
    k1 = f(x[i], y[i]);
    k2 = f(x[i] + h, y[i] + h * k1);
    y[i+1] = y[i] + h / 2.0 * (k1 + k2);
    cout << "y" << i+1 << " = " << y[i+1] << endl;
}
```

输出结果

这里取区间[0,1]，剖分点数5进行测试，求得y1 = 1.05878，y2 = 1.23355，y3 = 1.49049，y4 = 1.73652，y5 = 1.83859



问题三

3. （Page187, Project22）用改进的 Euler 公式求解常微分方程组初值问题计算公式：

$$\begin{pmatrix} \bar{y}_{n+1} \\ \bar{z}_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + h \begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + \frac{h}{2} \left[\begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} + \begin{pmatrix} f(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \\ g(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \end{pmatrix} \right] \tag{5}$$

输入：区间剖分点数 N，区间端点 a, b，定义函数

$$y'(x) = f(x, y, z), z'(x) = g(x, y, z) \tag{6}$$

输出：(y_k, z_k)，$k = 1, 2, ..., N$

利用上述方法，求解课本 Page156 例题 7.7：

$$\begin{cases} \frac{du}{dt} = 0.09u(1 - \frac{u}{20}) - 0.45uv \\ \frac{dv}{dt} = 0.06v(1 - \frac{v}{15}) - 0.001uv \\ u(0) = 1.6 \\ v(0) = 1.2 \end{cases} \tag{7}$$

算法过程

如上图改进的Euler公式所示

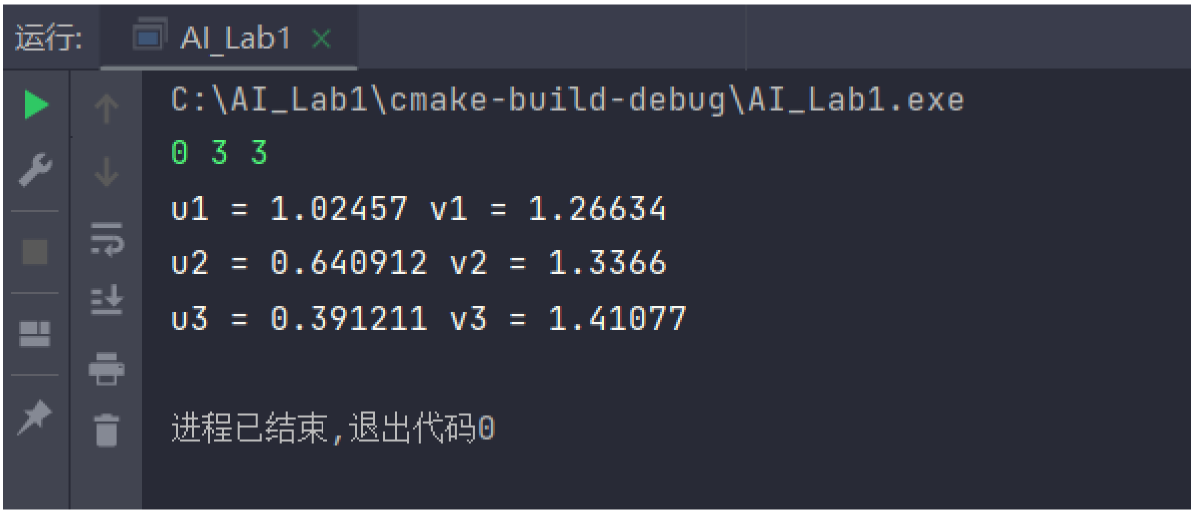
代码实现

依次计算u_[i+1], v_[i+1]，u[i+1]，v[i+1]，并将u[i+1]，v[i+1]输出

```
for(int i = 0; i < n; i++){
    u_[i+1] = u[i] + h * f(u[i], v[i]);
    v_[i+1] = v[i] + h * g(u[i], v[i]);
    u[i+1] = u[i] + h / 2.0 * (f(u[i], v[i]) + f(u_[i+1], v_[i+1]));
    v[i+1] = v[i] + h / 2.0 * (g(u[i], v[i]) + g(u_[i+1], v_[i+1]));
    cout << "u" << i+1 << " = " << u[i+1] << " " << "v" << i+1 << " = " <<
v[i+1] << endl;
}
```

输出结果

输入区间[0,3]，剖分点数3，计算得出u3 = 0.391211，v3 = 1.41077，即3年后这一对寄生虫数量分别为 0.391211和1.41077



实验总结

通过本次实验我基本掌握了Newton迭代法，Rouge-Kutta方法，改进的Euler公式。解决非线性方程组求解，常微分方程组初值问题。

参考资料

[1]数值计算方法与算法.第三版.张韵华,王新茂编

附录

问题一代码

```
#include <cmath>
#include "iostream"

using namespace std;

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

double g(double x, double y){
    return x * x * x - y;
}

int main(){

    double eps = 1E-5, x[100], y[100], delta_x, delta_y;
    x[0] = 0.8;
    y[0] = 0.6;
    int i = 0;
    while(true){
        delta_x = (-f(x[i], y[i]) - g(x[i], y[i]) * 2 * y[i]) / (2 * x[i] + 6.0
* x[i] * x[i] * y[i]);
        delta_y = 3 * x[i] * x[i] * delta_x + g(x[i], y[i]);
        x[i+1] = x[i] + delta_x;
        y[i+1] = y[i] + delta_y;
        cout << "第" << i+1 << "次迭代: " << "(" << x[i+1] << ", " << y[i+1] <<
")" << endl;
        i++;
        //cout << "max:" << max(delta_x, delta_y) << endl;
        if(max(fabs(delta_x), fabs(delta_y)) <= eps)
            break;
    }
    return 0;
}
```

问题二代码

```
#include <cmath>
#include "iostream"

using namespace std;
```

```
double f(double x, double y){

    return y * sin(numbers::pi * x);
}
int main(){
    int n;
    double a, b, y[100], x[100], k1, k2;
    y[0] = 1.0;
    cin >> a >> b >> n;
    double h = (b - a)/double(n);
    for(int i = 0; i < n; i++){
        x[i] = i * h;
        k1 = f(x[i], y[i]);
        k2 = f(x[i] + h, y[i] + h * k1);
        y[i+1] = y[i] + h / 2.0 * (k1 + k2);
        cout << "y" << i+1 << " = " << y[i+1] << endl;
    }
    return 0;
}
```

问题三代码

```
#include <cmath>
#include "iostream"

using namespace std;

double f(double u, double v){
    return 0.09 * u * (1 - u / 20.0) - 0.45 * u * v;
}

double g(double u, double v){
    return 0.06 * v * (1 - v / 15.0) - 0.001 * u * v;
}

int main(){
    double u[100], v[100], u_[100], v_[100], a, b;
    int n;
    u[0] = 1.6;
    v[0] = 1.2;
    cin >> a >> b >> n;
    double h = (b - a)/double(n);
    for(int i = 0; i < n; i++){
        u_[i+1] = u[i] + h * f(u[i], v[i]);
        v_[i+1] = v[i] + h * g(u[i], v[i]);
        u[i+1] = u[i] + h / 2.0 * (f(u[i], v[i]) + f(u_[i+1], v_[i+1]));
        v[i+1] = v[i] + h / 2.0 * (g(u[i], v[i]) + g(u_[i+1], v_[i+1]));
        cout << "u" << i+1 << " = " << u[i+1] << " " << "v" << i+1 << " = " <<
v[i+1] << endl;
    }

    return 0;
}
```