实验二 常微分方程初值问题数值解法(4课时)

一、实验目的

1.掌握 Euler 法和改进的 Euler 法公式的用法。
2.进一步加深对微分方程数值解的理解。

二、实验要求

1.编写欧拉法程序。
2.编写改进的欧拉法程序，学会用改进的欧拉公式来求解常微分方程初值问题。

三、实验原理

（一）常微分方程初值问题

$$\begin{cases} y' = f(x,y), a \le x \le b \\ y(a) = y_0 \end{cases} \qquad (\text{式 } 1)$$

的数值解法，这也是科学与工程计算经常遇到的问题。
（二）欧拉法

求初值问题(式 1)的一种最简单方法是将节点x_n的导数$y'(x_n)$用差商$\dfrac{y(x_n+h)-y(x_n)}{h}$代替，于是(式 1)的方程可近似写成

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y)(x_n)), n = 0,1,\cdots \qquad (\text{式 } 2)$$

从x_0出发$y(a)=y(x_0)=y_0$，由(式 2)求得$y(x_1) \approx y_0 + hf(x_0,y_0) = y_1$再将$y_1 \approx y(x_1)$代入(式 2)右端，得到$y(x_2)$的近似$y_2 = y_1 + hf(x_1,y_1)$，一般写成

$$y_{n+1} = y_n + hf(x_n, y_n), n = 0,1,\cdots$$

称为解初值问题的 Euler 法。

（三）改进欧拉法

先用 Euler 法计算出y_{n+1}的近似$\bar{y}_{n+1}$，将隐式梯形公式改为

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \dfrac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})] \end{cases}$$

称为改进 Euler 法，它实际上是显式方法。即

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))]$$

（四）例子

求初值问题 $\begin{cases} y' = y - \dfrac{2x}{y}, \quad 0 \le x \le 1 \\ y(0) = 1 \end{cases}$ 的数值解，取步长 $h = 0.1$ 。（精确解为

$y(x) = (1+2x)^{1/2}$ ）

解：（1）利用欧拉法 $\begin{cases} y_{i+1} = 1.1y_i - 0.2x_i / y_i \\ y_0 = 1 \quad , i = 0,1,2\cdots,9 \end{cases}$

（2）利用改进欧拉法 $\begin{cases} y_{i+1} = y_i + 0.05(K_1 + K_2) \\ K_1 = y_i - 2x_i / y_i \\ K_2 = y_i + 0.1K_1 - \dfrac{2(x_i + 0.1)}{y_i + 0.1K_1} \\ y_0 = 1 \quad , i = 0,1,2\cdots,9 \end{cases}$

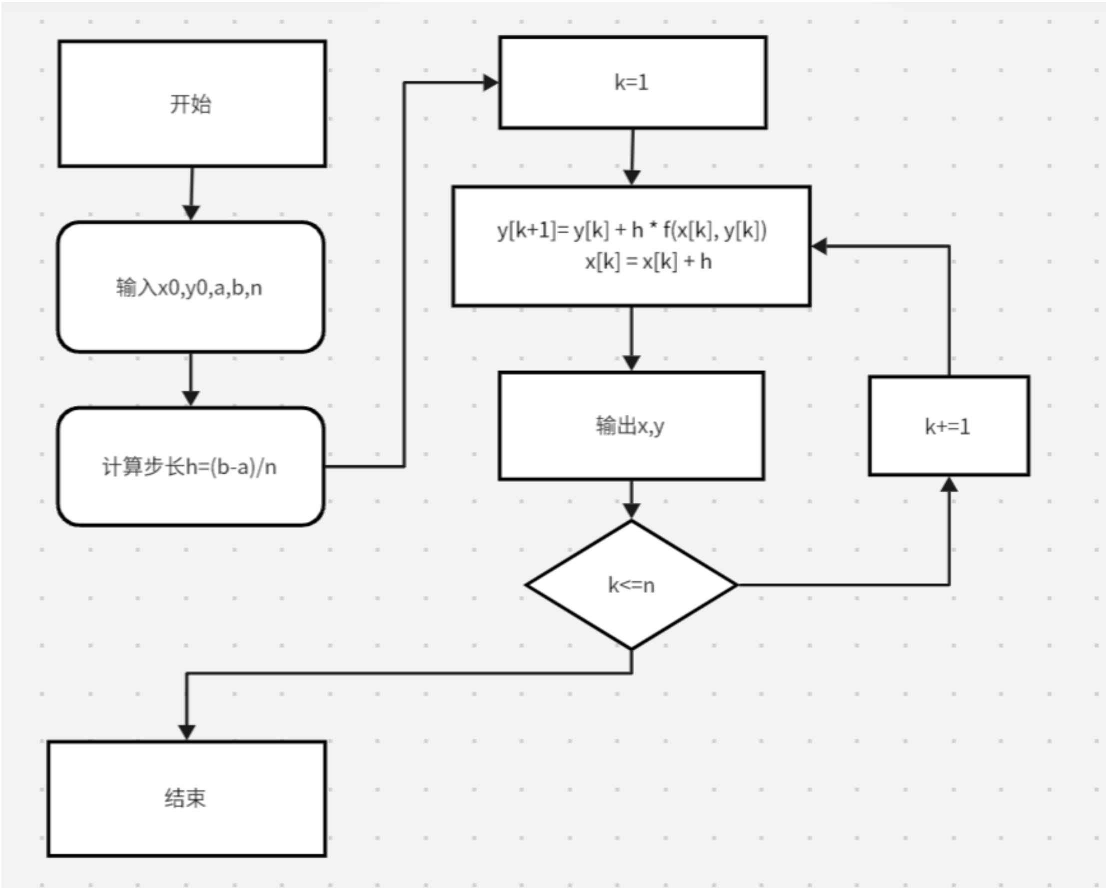
计算结果如下：

i	xi	Euler 方法 yi	改进 Euler 法 yi	精确解 y(xi)
0	0	1	1	1
1	0.1	1.1	1.	1.
2	0.2	1.	1.	1.
3	0.3	1.	1.	1.
4	0.4	1.	1.	1.
5	0.5	1.	1.	1.
6	0.6	1.	1.	1.
7	0.7	1.	1.	1.
8	0.8	1.	1.	1.
9	0.9	1.	1.	1.
10	1	1.	1.	1.

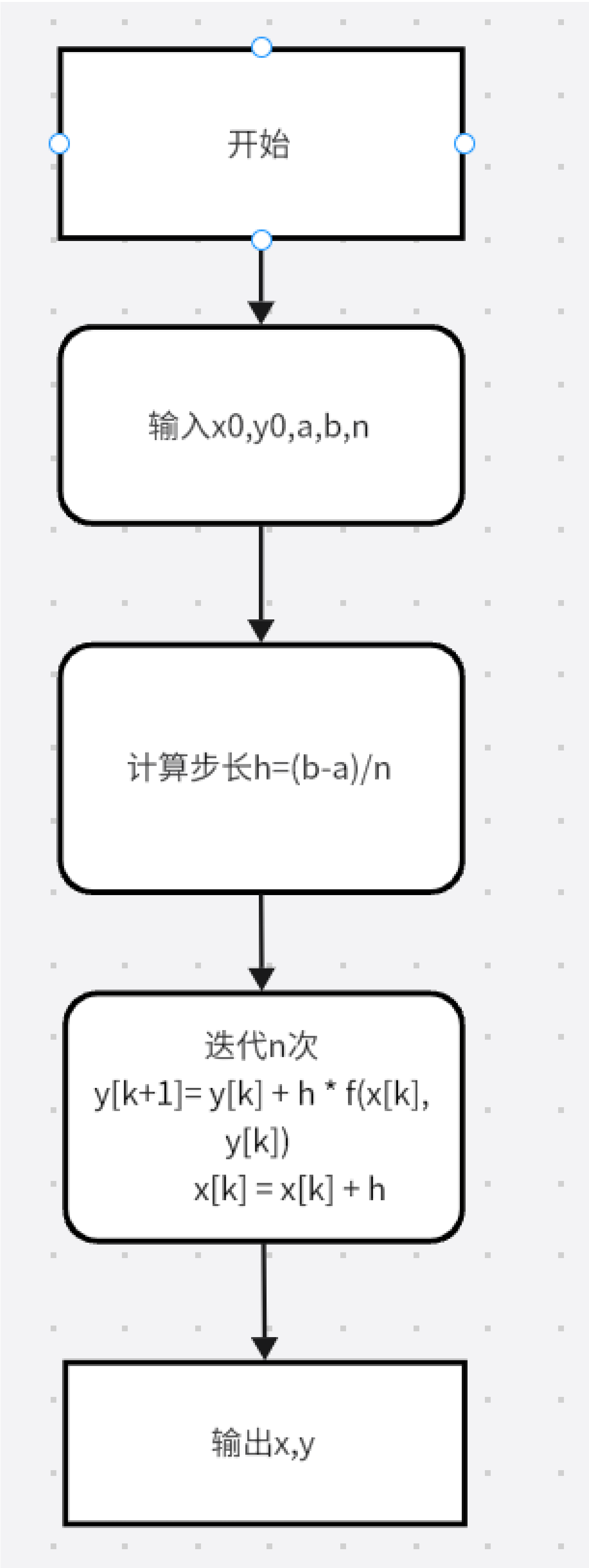
四、实验内容

（一）算法流程图

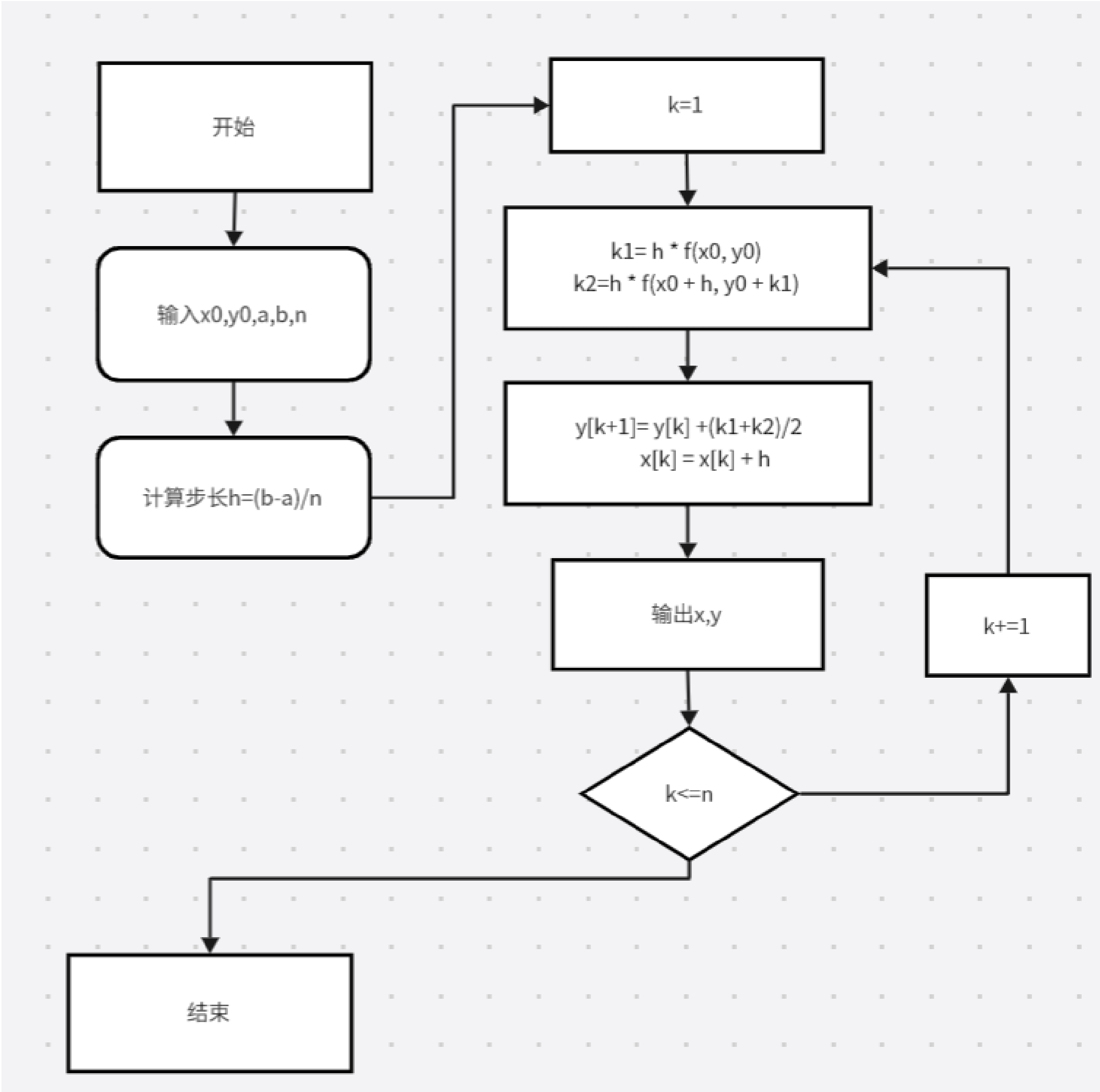
1. 请根据欧拉公式，画出其算法流程图



2. 请根据改进欧拉公式，画出其算法流程图。



2. 请根据改进欧拉公式，画出其算法流程图。



（二）编程作业

编写 Euler 法和改进的 Euler 法程序。求微分方程 $\begin{cases} \dfrac{dy}{dx} = -y, & x \in [0,1] \\ y(0) = 1 \end{cases}$ 在区间 10 等份的近似解。

1. Euler 法程序

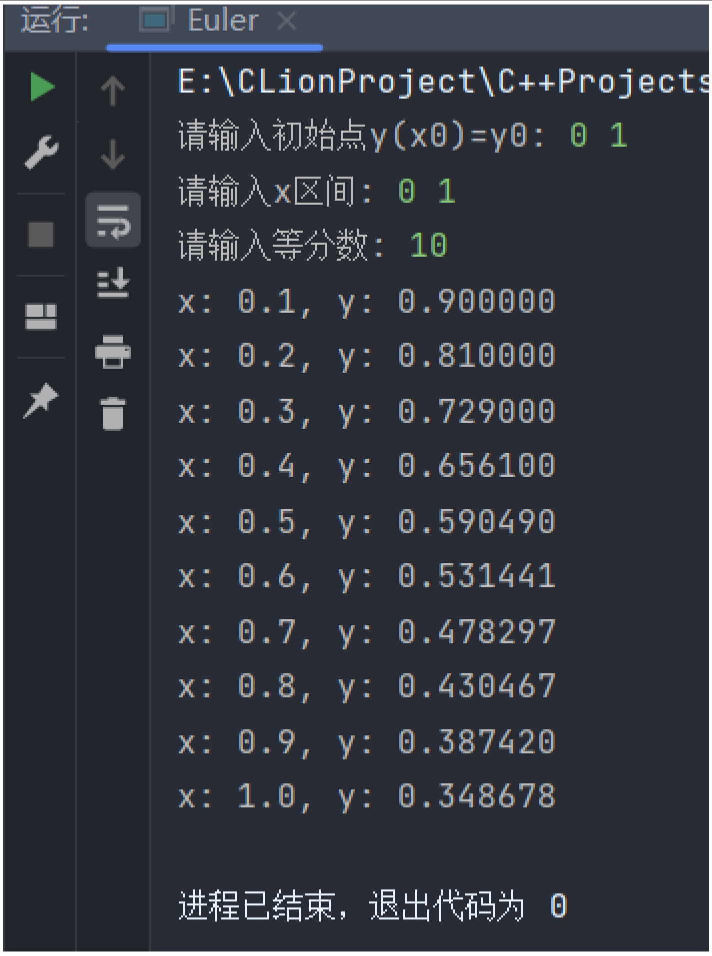
```
#include <bits/stdc++.h>

using namespace std;

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

int main() {
```

```
    double h, x0, y0, a, b, n;
    cout << "请输入初始点 y(x0)=y0: ";
    cin >> x0 >> y0;
    cout << "请输入 x 区间: ";
    cin >> a >> b;
    cout << "请输入等分数: ";
    cin >> n;
    h = (b - a) / n;
    for (int i = 0; i < n; i++) {
        y0 = y0 + h * f(x0, y0); // 迭代计算下一个点的 y 值
        x0 = x0 + h; // 更新 x 值
        printf("x: %.1f, y: %.6f\n", x0, y0); // 输出结果
    }
    return 0;
}
```



2. 改进的 Euler 法程序

```
#include <bits/stdc++.h>

using namespace std;

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

```cpp
    return -y;
}

int main() {
    double h, x0, y0, a, b, n;
    cout << "请输入初始点 y(x0)=y0: ";
    cin >> x0 >> y0;
    cout << "请输入 x 区间: ";
    cin >> a >> b;
    cout << "请输入等分数: ";
    cin >> n;
    h = (b - a) / n;
    for (int i = 0; i < n; i++) {
        double k1 = h * f(x0, y0); // 计算 k1
        double k2 = h * f(x0 + h, y0 + k1); // 计算 k2
        y0 = y0 + (k1 + k2) / 2; // 更新 y 值
      x0 = x0 + h; // 更新 x 值
      printf("x: %.1f, y: %.6f\n", x0, y0); // 输出结果
   }
    return 0;
}
```

在编写改进的 Euler 法程序时，有关输入输出部分，可参照以下屏幕上应出现内容。

please input a,b,h and a0

0 1 0.1 1

x=0., y=1.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=0., y=0.

x=1., y=0.

（三）选做题

使用梯形公式编写程序，解决上述例子问题。

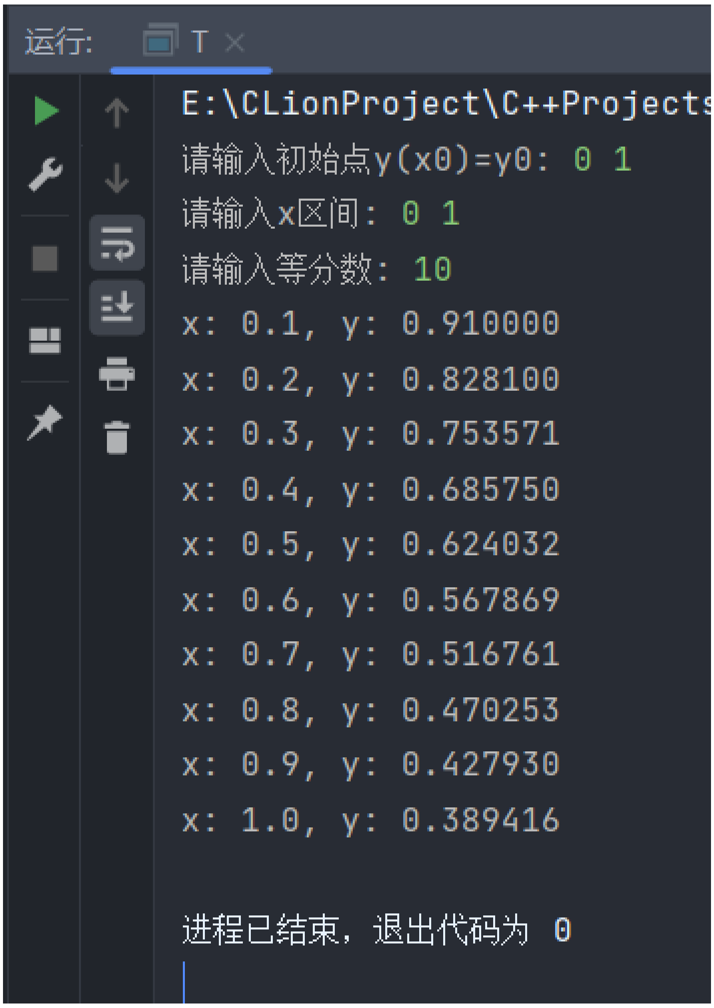
```
#include <bits/stdc++.h>

using namespace std;

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

int main() {
    double h, x0, y0, a, b, n;
    cout << "请输入初始点 y(x0)=y0: ";
    cin >> x0 >> y0;
    cout << "请输入 x 区间: ";
    cin >> a >> b;
    cout << "请输入等分数: ";
    cin >> n;
    h = (b - a) / n;

    for (int i = 0; i < n; i++) {
        double k1 = h * f(x0 - h, y0);
        double k2 = h * f(x0, y0 + k1);
        y0 = y0 + k2;
        x0 = x0 + h;
        printf("x: %.1f, y: %.6f\n", x0, y0); // 输出结果
    }
    return 0;
}
```


运行:
T
E:\CLionProject\C++Project
请输入初始点y(x0)=y0: 0 1
请输入x区间: 0 1
请输入等分数: 10
x: 0.1, y: 0.910000
x: 0.2, y: 0.828100
x: 0.3, y: 0.753571
x: 0.4, y: 0.685750
x: 0.5, y: 0.624032
x: 0.6, y: 0.567869
x: 0.7, y: 0.516761
x: 0.8, y: 0.470253
x: 0.9, y: 0.427930
x: 1.0, y: 0.389416
进程已结束，退出代码为 0