计算方法实验报告

姓名：王木一

学号：200210231

院系：计算机科学与技术

专业：计算机类

班级：20 级计科 8 班

实验报告一·Lab5 高斯 Gauss 列主元消元法

第一部分：问题分析 *（描述并总结出实验题目）*

工程实践中常出现求线性方程组$Ax = b$的解。求解此问题有直接法和迭代法。

实验目的：

编写代码，使用直接法中的高斯 Gauss 消元法，并在消元时使用列选主元策略消元（未要求使用显式隐式相对），将复杂矩阵转化为三角阵，最后利用回代解出方程组的近似解（或确定该方程组是奇异的）。

输入：

n 方程组阶数

A 方程组系数矩阵

b 方程组等号右端向量

输出：

x 方程组的解（或奇异标志）

第二部分：数学原理

高斯 Gauss 消元法思想：

将复杂系数矩阵A进行消元，转化为上三角矩阵G。同时方程组右端向量由b转化为d。即利用消元将线性方程组$Ax = b$转化为等价的$Gx = d$形式。由于此时的系数矩阵为上三角阵，可利用回代（ 即从x_n起递推计算出x向量其他值），最后求出方程组的近似解。

消元过程：

第k步：（$k = 1,2, \dots, n - 1$）

若$a_{kk} = 0$，则此系数矩阵奇异

$$m_{ik} = \frac{a_{ik}}{a_{kk}} \qquad i = k + 1, \cdots, n$$

$a_{ij} = a_{ij} - a_{kj} m_{ik}$ 系数矩阵第 i 行=第 i 行-第 k 行×m_{ik}

$b_i = b_i - b_k m_{ik}$ 右端向量第 i 个=第 i 个-第 k 个×m_{ik}

$k = n, a_{nn} = 0$,则此系数矩阵奇异

回代过程：

$$x_n = \frac{b_n}{a_{nn}}$$

$$x_k = (b_k - \sum_{j=k+1}^{n} a_{kj} x_j)/a_{kk} \qquad k = n - 1, \cdots, 2, 1$$

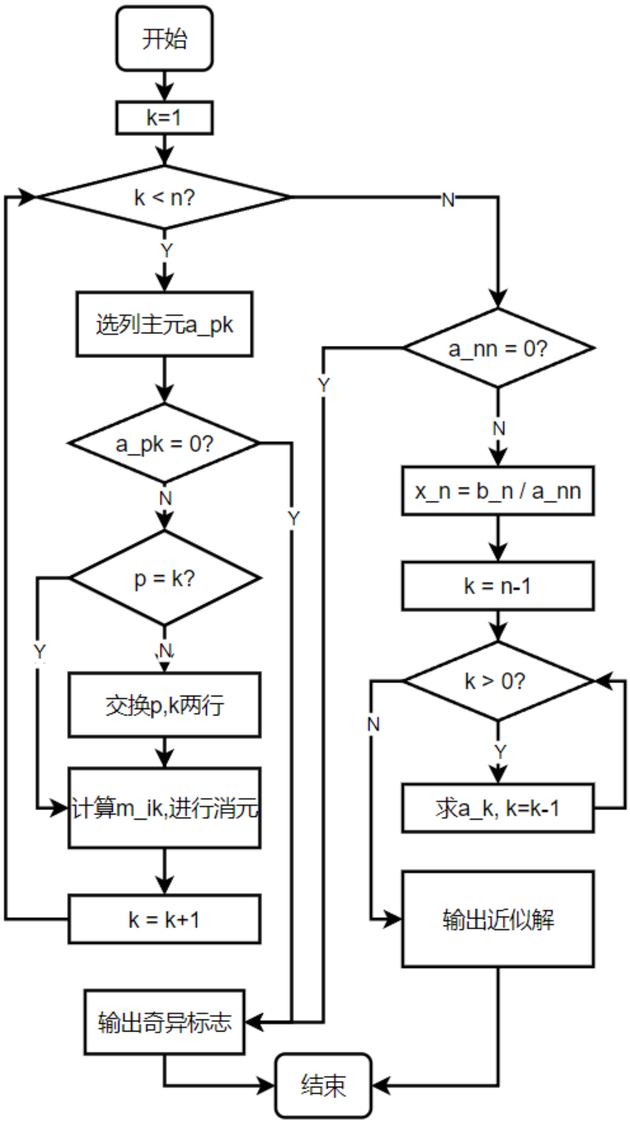
列选主元消元策略：

利用高斯消元将系数矩阵转化为上三角阵时，a_{kk}可能等于 0 或绝对值很小，不利于计算。而我们又知道：交换系数矩阵某两行不会改变解。故第 k 步消元之前，我们在第 k 列的第 k+1 到 n 行中选择绝对值最大的元素a_{pk}若$a_{pk} = 0$，则此系数矩阵奇异；若$p \neq k$,就交换矩阵第 p 行和第 k 行（右端向量

也要交换b_p，b_k）。再使用交换后的矩阵进行消元。

第三部分：程序设计流程

流程图



部分代码截图：

main.m* gauss.m

```
A11 = [0.4096 0.1234 0.3678 0.2943;     % @param A 系数矩阵
       0.2246 0.3872 0.4015 0.1129;
       0.3645 0.1920 0.3781 0.0643;
       0.1784 0.4002 0.2786 0.3927];
b11 = [1.1951;1.1262;0.9989;1.2499];    % @param b 右端向量
n11 = 4;                                % @param n 系数矩阵维数
disp("问题1（1）")
gauss(n11, A11, b11);
```

```
function [] = gauss(n, A, b)
x = zeros(n, 1);
for k = 1:n
    c = abs(A(k:end, k)); %取矩阵A的第k列的k到最后的数据，并将所有数据取绝对值
    [MAX, pm] = max(c);
    if MAX == 0          %列主元为0，奇异矩阵
        disp('Error!The matrix is singular!');
        return;
    end
    p = pm + k - 1;
    if p ~= k            %交换pk两行
        A([k,p], :) = A([p,k], :);
        b([k,p], :) = b([p,k], :);
    end
    for i = k+1:n        %进行消元
        m = A(i,k)/A(k,k);
        A(i,:) = A(i,:) - m*A(k,:);
        b(i) = b(i) - m*b(k);
    end
end
x(n) = b(n)/A(n,n);
for k = n-1:-1:1        %回代
    sum = 0;
    for j = k+1:n
        sum = sum + A(k,j)*x(j);
    end
    x(k) = (b(k) - sum)/A(k,k);
end
disp('ans = ');
disp(x);
end
```

第四部分：实验结果、结论与讨论

问题 1		x1	x2	x3	x4
1	近似解	1.000000	1.000000	1.000000	1.000000
	精确解	1.000000	1.000000	1.000000	1.000000
2	近似解	1.000000	1.000000	1.000000	1.000000
	精确解	1.000000	1.000000	1.000000	1.000000
3	近似解	1.000000	1.000000	1.000000	1.000000
	精确解	1.000000	1.000000	1.000000	1.000000
4	近似解	1.000000	1.000000	1.000000	1.000000
	精确解	1.000000	1.000000	1.000000	1.000000
问题 2					

1	近似解	0.953679	0.320957	1.078708	-0.090109
	精确解	0.953679	0.320957	1.078708	-0.090109
2	近似解	0.516177	0.415219	0.109966	1.036539
	精确解	0.516177	0.415219	0.109966	1.036539
3	近似解	1.000000	1.000000	1.000000	1.000000
	精确解	1.000000	1.000000	1.000000	1.000000
4	近似解	1.000000	1.000000	1.000000	
	精确解	1.000000	1.000000	1.000000	

注：解向量为转置后的结果

```
>> main
问题1（1）
ans =
    1.000000000000003
    1.000000000000002
    0.999999999999997
    0.999999999999999

问题1（2）
ans =
    1.000000000000118
    0.999999999999824
    0.999999999999901
    1.000000000000026

问题1（3）
ans =
    0.999999999999993
    1.000000000000069
    0.999999999999855
    1.000000000000085

问题1（4）
ans =
     1
     1
     1
     1
```

```
问题2（1）
ans =
    0.953679106901772
    0.320956845521104
    1.078708075793238
   -0.090108509539579

问题2（2）
ans =
    0.516177297958542
    0.415219472830135
    0.109966102867889
    1.036539223336201

问题2（3）
ans =
    1.000000000000000
    1.000000000000000
    1.000000000000000

问题2（4）
ans =
     1
     1
     1

fx >>
```

实验报告二·Lab4 牛顿 Newton 迭代法

第一部分：问题分析 *（描述并总结出实验题目）*

工程问题中有时需要求非线性方程$f(x)=0$的根x^*。求解此问题的数值方法有：二分法和迭代法。

本次实验目的：

利用牛顿迭代法求非线性方程$f(x)=0$的根x^*在指定迭代次数范围及精度下的近似值。

输入：

f 待求解方程

α 迭代初值

ε_1 迭代精度 1

ε_2 迭代精度 2

N 最大迭代次数

输出：

x' 解的近似值或失败标志

第二部分：数学原理

牛顿迭代法：

牛顿迭代法本质上是切线法，利用某点的切线将x逐步逼近方程的根。

迭代方程：

$$x_0 = \alpha$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

通过迭代方程可得到序列$\{x_i\}_{i=0}^{n}$,当随着$n\to\infty, x_i\to \mathrm{x}^*$说明迭代收敛.当$|x_{n+1}-x_n|<\varepsilon$,可将$x_{n+1}$视作根的近似值。令$\varphi(x)=x-\frac{f(x)}{f'(x)}$

收敛性与收敛速度（收敛阶）：

为判断是否收敛有以下定理：

【定理 1】若$\varphi(x)$在x^*的某个邻域内有一阶导数，且$|\varphi'(x)|<1$,则迭代过程在这个邻域内收敛。

【定理 2】设$f(x)$在$[a,b]$上二阶导数存在，且满足：

①$f(a)f(b)<0$

②$\forall x\in[a,b]\ f'(x)\neq 0$

③$\forall x\in[a,b]\ f"(x)$不变号

④初值$\forall x_0\in[a,b]\ f''(x_0)f(x_0)>0$

则迭代收敛于唯一值。

【定理 3】若$\varphi(x)$在x^*的某个邻域内有充分多阶连续导数：

迭代时p阶的 等价于 $\varphi^{(j)}(x)=0$，$\varphi^{(p)}(x)\neq 0 \quad j=1,2,\cdots,p-1$

那么，已知$\varphi'(x)=\frac{f(x)f''(x)}{[f'(x)]^2}$

若$f(x)\in C^2[a,b]$ $f(x^*)=0$ $f'(x^*)\neq 0$,则满足定理 1，迭代收敛，并且由定理 3 可知迭代是 2 阶的

而若$f(x)\in C^m[a,b]$ $f(x^*)=f'(x^*)=\cdots=f^{(m-1)}(x^*)=0$ $f^{(m)}(x^*)\neq 0$

则说明根是重根，当选取合适的初值，迭代也会收敛，同时迭代是 1 阶的。

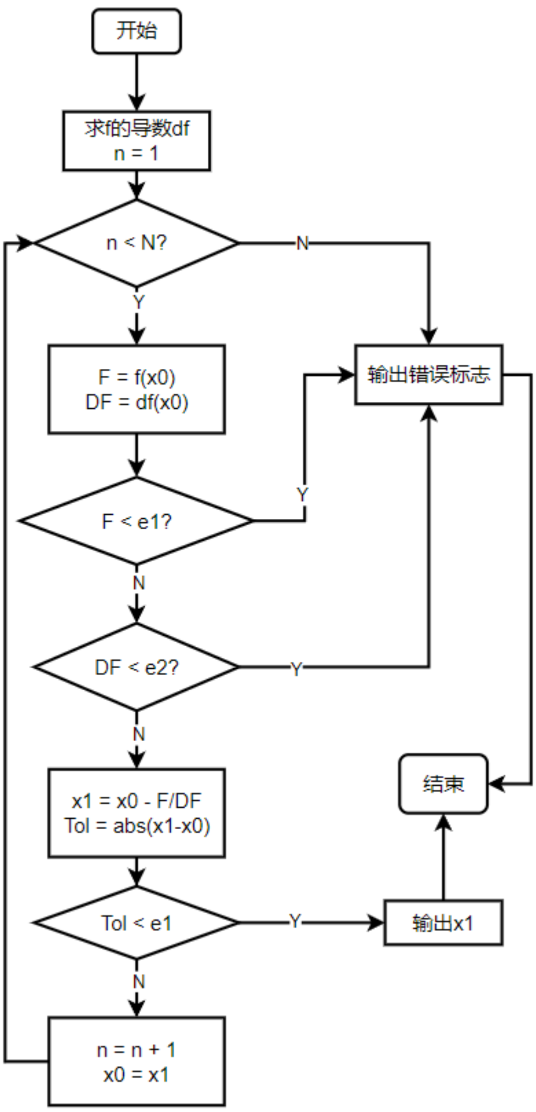
第三部分：程序设计流程

（To 老师/助教：当 matlab 中测试的代码过多，内存中存有多个变量，可能会导致 Lab4 代码编译不通过（因为其使用了符号变量 x）产生如图的报错。解决方法：命令行输入 clear 后，再调用 main，即可编译通过输出结果）

```
>> main
错误使用 sym/subsindex (line 825)
Invalid indexing or function definition. Indexing must follow MATLAB indexing. Function arguments must be symbolic variables, and function body must be sym expression.

出错 main (line 2)
f1(x) = cos(x) - x;                    % @param f 非线性方程
```

流程图：



部分代码截图：

main.m* | newton.m

```matlab
syms x;
f1(x) = cos(x) - x;                    % @param f 非线性方程
f2(x) = exp(-x) - sin(x);
f3(x) = x- exp(-x);
f4(x) = x^2 - 2*x*exp(-x) + exp(-2*x);
e1 = 10^(-6);                          % @param e1 精度1
e2 = 10^(-4);                          % @param e2 精度2
N1 = 10;                               % @param N 最大迭代次数
N2 = 20;
x01 = pi/4;                            % @param x0 初值
x02 = 0.6;
x03 = 0.5;
x04 = 0.5675;

disp("问题一（1）");
newton(f1, x01, e1, e2, N1);
```

main.m | newton.m

```matlab
function []= newton(f, x0, e1, e2, N)
n = 1;
df = diff(f);
while n <= N
    F = f(x0);
    DF = df(x0);
    if abs(F) < e1
        fprintf("ans = %.6f\n",x0);
        fprintf("iteration %d times\n",n);
        return;
    elseif abs(DF) < e2
        disp('Error!');
        return;
    else
        x1 = x0 - (F/DF);
        Tol = abs(x1 - x0);
        if Tol < e1
            fprintf("ans = %.6f\n",x1);
            fprintf("iteration %d times\n",n);
            return;
        end
        n = n + 1;
        x0 = x1;
    end
end
disp('Error!');
end
```

第四部分：实验结果、结论与讨论

实验结果：

问题	1.1	1.2	2.1	2.2
近似解	0.739085	0.588533	0.567143	0.567143

```
>> main
问题一（1）
ans = 0.739085
iteration 3 times
问题一（2）
ans = 0.588533
iteration 3 times
问题二（1）
ans = 0.567143
iteration 3 times
问题二（2）
ans = 0.567143
iteration 3 times
```

思考题：

1. 对实验 1 确定初值的原则是什么？实际计算中应如何解决？

答：

确定初值原则：

选取初值时应尽可能靠近根，才更大概率收敛且迭代次数更少

解决方法：

可以事先尝试多个点计算其函数值，选择较小的；或电脑做出图像，根据图像选择。也有简化方法：对于方程$f(x)=0$,若初值x_0满足

$$f''(x_0)\neq 0,\quad |f'(x_0)|^2>|\frac{f(x_0)f''(x_0)}{2}|$$

则迭代在大多数情况下收敛。

（来源：https://blog.csdn.net/SanyHo/article/details/106365358）

2.. 对实验 2 如何解释在计算中出现的现象？试加以说明

答：

现象：

当计算问题 2（2）时，程序超时并报错

说明：

通过调试可知，解决问题 2（2）时完成每次迭代的时间越来越长。仔细观察原函数和代码可知原因：

①问题 2（2）的根是二重根，并且是上一题的平方，由定理 3 可知其迭代是 1 阶的，迭代耗时长。

②代码中比较精度的代码，如 if abs(F) < e1。不等式两边类型不同，当迭代次数增加符号变量变得越来越复杂。（解决：强制类型转换 if double(abs(F) < e1)

解决：改进迭代格式为：

$$x_0=\alpha$$

$$x_{n+1}=x_n-2\frac{f(x_n)}{f'(x_n)}$$

此时$\varphi'(x)=0$，由定理 3 可知迭代恢复 2 阶，迭代加快。

实验报告三·Lab2 龙贝格 Romberg 积分法

第一部分：问题分析 *（描述并总结出实验题目）*

工程问题中会遇到求解定积分，但有时由于 Newton-Leibnitz 方法难找到原函数，或原函数复杂，或只知道一系列的离散点。这时就要使用数值积分方法求定积分，包括矩形公式、梯形公式、复化梯形公式、复化 Simpson 公式、复化 Cotes 公式以及 Romerg 积分法。

实验目的：

使用 Romberg 积分法求解定积分，并输出 T 数表。

输入：

f　待积函数
a　积分下限
b　积分上限
ϵ　积分精度

输出：

I'　定积分近似值
T　T 数表

第二部分：数学原理

龙贝格积分法：

已知利用复化梯形公式，通过改变步长h,使其逼近 0 得到一系列复化梯形公式，并且其值也逐渐逼近真实积分值。

由 Euler-Maclaurin 求和公式可知积分公式与复化梯形公式及其余项的关系，即$\int_a^b f(x)dx = T(0) = T(h) + \sum_{j=1}^{\infty} a_j h^{2j}$

那么我们可以使用 Richardson 外推法，将逼近的速度加快.一般的，对于起始的$T_{0,0}$我们取最简单的梯形公式，即$n = 1, h = b - a$.可得龙贝格积分递推公式：

$$T_{0,0} = \frac{b-a}{2}\big(f(a) + f(b)\big)$$

$$T_{0,i} = \frac{1}{2}[\,T_{0,i-1} + \frac{b-a}{2^{i-1}}\sum_{j=1}^{2^{i-1}} f(\,a + (2j-1)\frac{b-a}{2^i})]$$

$$T_{m,k} = \frac{4^m T_{m-1,k+1} - T_{m-1,k}}{4^m - 1}$$

当$|T_{m+1,0} - T_{m,0}| < \varepsilon$时，可将$T_{m+1,0}$视作定积分的近似值。

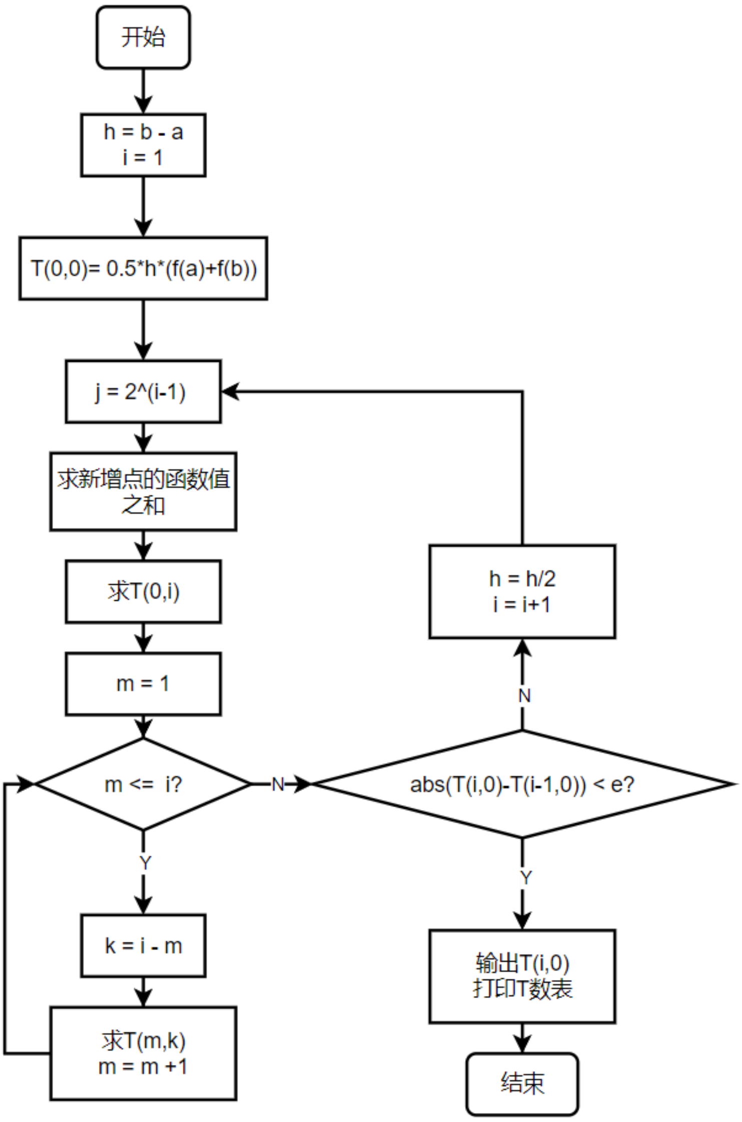
通过计算可以发现：

$m = 0$时，得到复化梯形公式
$m = 1$时，得到复化 Simpson 公式
$m = 2$时，得到复化 Cotes 公式
$m = 3$时，得到复化 Romberg 公式

第三部分：程序设计流程

与指导书中提供的算法有所不同的是：用矩阵存储 T 数表，最后输出矩阵

流程图：



部分代码截图：

```
main.m × Romberg.m × +
disp("Q1.1");
f = @(x) x^2*exp(x);     % @param f 待积函数
a = 0;                   % @param a 积分下限
b = 1;                   % @param b 积分上限
e = 10^(-6);             % @param e 精度
Romberg(f, a, b, e);
```

main.m | Romberg.m

```
function []=Romberg(f, a, b, e)
h = b - a;
i = 1;
T=[];
T(1, 1)= 0.5*h*(f(a)+f(b));
while true
    j = 2^(i-1);
    sum = 0;
    for k = 1:j
        sum = sum + f(a+(k-1/2)*h);
    end
    T(1, i+1) = 0.5*( T(1, i-1+1) + h*sum );
    for m = 1:i
        k = i - m;
        T(m+1, k+1) = ( 4^m * T(m-1+1, k+1+1) -T(m-1+1, k+1))/(4^m -1);
    end
    if abs(T(i+1, 0+1)-T(i-1+1, 0+1)) < e
        fprintf("近似值:%.12f\n", T(i+1, 0+1));
        disp("T数表:");
        disp(T);
        return;
    end
    h = h/2;
    i = i+1;
end
return;
end
```

第四部分：实验结果、结论与讨论

实验结果

问题1. 1	近似值	0.7182818					
	T数表	1.3591409	0.8856606	0.7605963	0.7288902	0.7209358	
		0.7278338	0.7189082	0.7183215	0.7182843		
		0.7183132	0.7182823	0.7182818			
		0.7182819	0.7182818				
		0.7182818					

问题1. 2	近似值	10.9501703					
	T数表	5.1218264	9.2797629	10.5205543	10.8420435	10.9230939	10.9433984
		10.6657417	10.9341514	10.9492065	10.9501107	10.9501666	
		10.9520454	10.9502102	10.9501710	10.9501703		
		10.9501811	10.9501704	10.9501703			
		10.9501703	10.9501703				
		10.9501703					

问题1. 3	近似值	3.1415927					
	T数表	3.00000000	3.10000000	3.13117647	3.13898849	3.14094161	3.14142989
		3.13333333	3.14156863	3.14159250	3.14159265	3.14159265	
		3.14211765	3.14159409	3.14159266	3.14159265		
		3.14158578	3.14159264	3.14159265			
		3.14159267	3.14159265				
		3.14159265					

问题1. 4	近似值	0.6931472					
	T数表	0.75000000	0.70833333	0.69702381	0.69412185	0.69339120	
		0.69444444	0.69325397	0.69315453	0.69314765		
		0.69317460	0.69314790	0.69314719			
		0.69314748	0.69314748	0.69314718			
		0.69314718					

思考题：

2. 在实验 1 中二分次数和精度的关系如何？

由教材 pp.198

(2) 可以证明：当 $f(x) \in C^{2m+2}[a,b]$ 时，$T_{m,k}$ 的余项为

$$E_{m,k}(f) = \int_a^b f(x)\mathrm{d}x - T_{m,k}$$

$$= -\frac{B_{2m+2}}{2^{(m+1)(m+2k)} \cdot (2m)!}(b-a)^{2m+3} f^{(2m+2)}(\zeta), \qquad (4.3.21)$$

其中，B_{2m+2} 是只与 m 有关而与 k 无关的常数，且 $\zeta \in (a,b)$.

即 T 数表第 m 行的求积公式$T_{m,k}$有$2m+1$阶代数精度。若要使精度越好，递推的深度也就要越深 m 越大，那么所需的基础即第 0 行元素越多，即复化梯形公式的二分次数越多。

实验报告四·Lab3 四阶龙格库塔 Runge-Kutta 方法

第一部分：问题分析 *（描述并总结出实验题目）*

实际工程实践中常遇到求解带初值的微分方程问题，有时更只要求在某点上的函数值。求解这一类问题有解析解法和数值解法，而数值解法包括单步法和多步法。

实验目的：

使用四级龙格库塔方法求解微分方程初值问题，并将求出的近似值与真正值比较，分析龙格库塔方法中步长对结果的影响。

输入：

f 微分方程（m 文件中用的是 df）

a 区间下界

b 区间上界

α 原函数在 a 点的初值

N 离散点数目（实际有 N+1 个点，控制步长）

y 微分方程准确解（实际中未知，但此次实验传入中用于分析 RK 方法的特点）（m 文件中用的是 f）

输出：

$\{y_i\}$ 一系列近似值

$\{t_i\}$ 一系列真实值

第二部分：数学原理

四阶龙格库塔法

龙格库塔方法是微分方程初值问题的数值解法，通过离散化方法，将微分方程转化为差分方程。通过构造$y(x_n)$的近似值y_n的递推格式，求解出各节点上的近似值。

其本质是利用 Taylor 展开的 r 级 Taylor 级数法，并且是单步法。但由于难以计算$f(x,y)$高阶导数和小区间中的$f(x,y)$值无法直接求得，故使用间接 Taylor 展开式，同时利用已知$f(x,y)$的线性组合来代替f',同时使用待定系数法使得 RK 方法有较高的阶数（精度）。

四阶龙格库塔法为 4 级 RK 方法，且其具有四阶精度。

四阶龙格库塔法递推公式：

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$K_1 = f(x_n, y_n)$$

$$K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right)$$

$$K_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2\right)$$

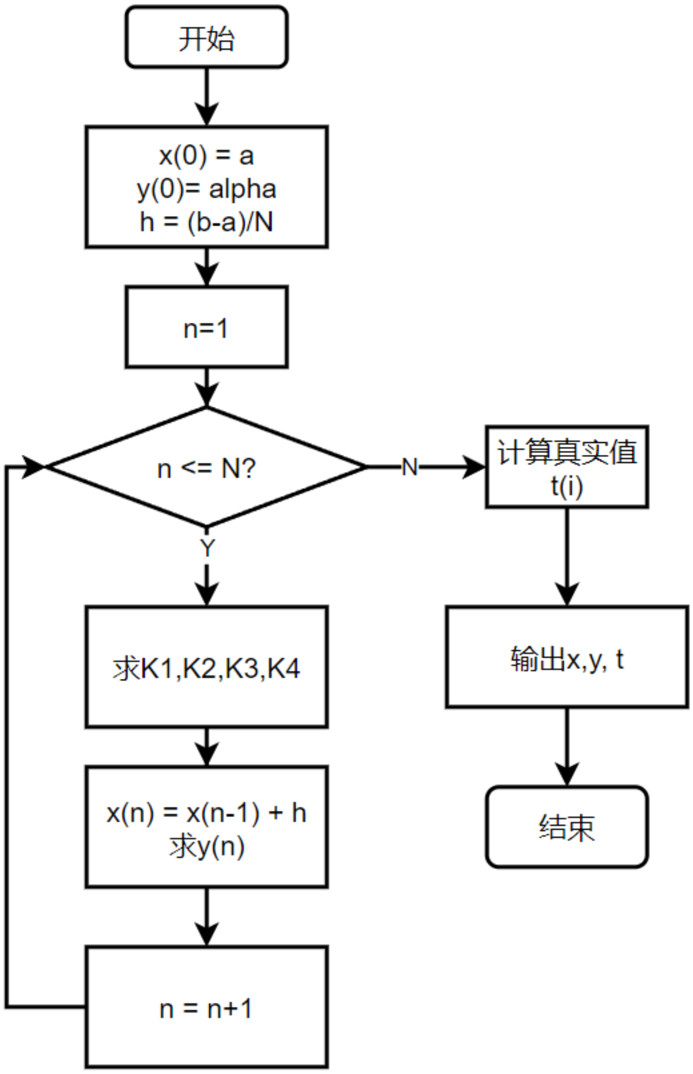
$$K_4 = f(x_n + h, y_n + hK_3)$$

$$h = \frac{b-a}{N}$$

第三部分：程序设计流程

与指导书中提供的算法有所不同的是：用矩阵存储x_i, y_i, t_i，最后输出矩阵

流程图：



部分代码截图：

RungeKutta.m | main.m

```
disp("Q1.1");
df = @(x,y)x+y;        %@param df 微分方程
f = @(x) -x-1;         %@param f  精确解
a = 0;                 %@param a 区间下限
b = 1;                 %@param b 区间上限
alpha = -1;            %@param alpha 初值
for N = [5, 10, 20]
    disp("N="+N);
    RungeKutta(df, a, b, alpha, N, f);
end
```

```matlab
function [] = RungeKutta(df, a, b, alpha, N, f)
x = [];
y = [];
true = [];        %@param true 真实值
x(1) = a;
y(1) = alpha;
h = (b-a)/N;
for n=1:N
    K1 = df(x(n-1+1), y(n-1+1));
    K2 = df((x(n-1+1)+h/2),  (y(n-1+1)+h*K1/2));
    K3 = df((x(n-1+1)+h/2),  (y(n-1+1)+h*K2/2));
    K4 = df((x(n-1+1)+h),  (y(n-1+1)+h*K3));
    x(n+1) = x(n) + h;
    y(n+1) = y(n) + h*(K1+2*K2+2*K3+K4)/6;
end
for i = 1:N+1
     true(i) = f(x(i));
end
disp("x=");
disp(x);
disp("近似值：");
disp(y);
disp("精确值：");
disp(true);
return;
end
```

第四部分：实验结果、结论与讨论

问题一

实验结果（加色块是为了比较同一自变量下，N 不同近似值的变化）

问题1. 1	N								
	5	x	0.000000	0.200000	0.400000	0.600000	0.800000	1.000000	
		近似值	-1.000000	-1.200000	-1.400000	-1.600000	-1.800000	-2.000000	
		真实值	-1.000000	-1.200000	-1.400000	-1.600000	-1.800000	-2.000000	
	10	x	0.000000	0.100000	0.200000	0.300000	0.400000	0.500000	
		近似值	-1.000000	-1.100000	-1.200000	-1.300000	-1.400000	-1.500000	
		真实值	-1.000000	-1.100000	-1.200000	-1.300000	-1.400000	-1.500000	
		x	0.600000	0.700000	0.800000	0.900000	1.000000		
		近似值	-1.600000	-1.700000	-1.800000	-1.900000	-2.000000		
		真实值	-1.600000	-1.700000	-1.800000	-1.900000	-2.000000		
	20	x	0.000000	0.050000	0.100000	0.150000	0.200000	0.250000	0.300000
		近似值	-1.000000	-1.050000	-1.100000	-1.150000	-1.200000	-1.250000	-1.300000
		真实值	-1.000000	-1.050000	-1.100000	-1.150000	-1.200000	-1.250000	-1.300000
		x	0.350000	0.400000	0.450000	0.500000	0.550000	0.600000	0.650000
		近似值	-1.350000	-1.400000	-1.450000	-1.500000	-1.550000	-1.600000	-1.650000
		真实值	-1.350000	-1.400000	-1.450000	-1.500000	-1.550000	-1.600000	-1.650000
		x	0.700000	0.750000	0.800000	0.850000	0.900000	0.950000	1.000000
		近似值	-1.700000	-1.750000	-1.800000	-1.850000	-1.900000	-1.950000	-2.000000
		真实值	-1.700000	-1.750000	-1.800000	-1.850000	-1.900000	-1.950000	-2.000000

问题1.2	N								
	5	x	0.000000	0.200000	0.400000	0.600000	0.800000	1.000000	
		近似值	1.00000000	0.83333904	0.71429213	0.62500589	0.55556069	0.50000441	
		真实值	1.00000000	0.83333333	0.71428571	0.62500000	0.55555556	0.50000000	
		偏差	0.00000000	-0.00000570	-0.00000642	-0.00000589	-0.00000513	-0.00000441	
	10	x	0.000000	0.100000	0.200000	0.300000	0.400000	0.500000	
		近似值	1.00000000	0.90909119	0.83333373	0.76923121	0.71428615	0.66666709	
		真实值	1.00000000	0.90909091	0.83333333	0.76923077	0.71428571	0.66666667	
		偏差	0.00000000	-0.00000028	-0.00000040	-0.00000044	-0.00000044	-0.00000042	
		x	0.600000	0.700000	0.800000	0.900000	1.000000		
		近似值	0.62500040	0.58823567	0.55555590	0.52631611	0.50000030		
		真实值	0.62500000	0.58823529	0.55555556	0.52631579	0.50000000		
		偏差	-0.00000040	-0.00000037	-0.00000035	-0.00000032	-0.00000030		
	20	x	0.000000	0.050000	0.100000	0.150000	0.200000	0.250000	0.300000
		近似值	1.00000000	0.95238096	0.90909093	0.86956524	0.83333336	0.80000003	0.76923080
		真实值	1.00000000	0.95238095	0.90909091	0.86956522	0.83333333	0.80000000	0.76923077
		偏差	0.00000000	-0.00000001	-0.00000002	-0.00000002	-0.00000003	-0.00000003	-0.00000003
		x	0.350000	0.400000	0.450000	0.500000	0.550000	0.600000	0.650000
		近似值	0.74074077	0.71428574	0.68965520	0.66666669	0.64516132	0.62500003	0.60606063
		真实值	0.74074074	0.71428571	0.68965517	0.66666667	0.64516129	0.62500000	0.60606061
		偏差	-0.00000003	-0.00000003	-0.00000003	-0.00000003	-0.00000003	-0.00000003	-0.00000002
		x	0.700000	0.750000	0.800000	0.850000	0.900000	0.950000	1.000000
		近似值	0.58823532	0.57142859	0.55555558	0.54054056	0.52631581	0.51282053	0.50000002
		真实值	0.58823529	0.57142857	0.55555556	0.54054054	0.52631579	0.51282051	0.50000000
		偏差	-0.00000002	-0.00000002	-0.00000002	-0.00000002	-0.00000002	-0.00000002	-0.00000002

1. 对实验 1，数值解和解析解相同吗？为什么？试加以说明。

答：

由上表可知，问题 1.1 数值解和解析解相同；问题 1.2 数值解和解析解不同。

原因：

因为问题 1.1 的精确解是线性方程，在任意点处的导数都为-1，RK 方法估计的导数值就是真实的导数值；而问题 1.2 的精确解不是线性方程，会存在误差。同时我们使用的 RK 方法是 4 阶方法，其对 4 阶（含）以下的多项式都精确成立，故问题 1.1 数值解和解析解相同；问题 1.2 数值解和解析解不同。

问题二

实验结果：

问题2.1	N								
	5	x	1.000000	1.400000	1.800000	2.200000	2.600000	3.000000	
		近似值	0.00000000	2.61394279	10.77631317	30.49165420	72.58559861	156.22519828	
		真实值	0.00000000	2.62035955	10.79362466	30.52458129	72.63928396	156.30529585	
		偏差	0.00000000	0.00641676	0.01731149	0.03292708	0.05368535	0.08009758	
	10	x	1.00000000	1.20000000	1.40000000	1.60000000	1.80000000	2.00000000	
		近似值	0.00000000	0.86637911	2.61974052	5.71989528	10.79201760	18.68085236	
		真实值	0.00000000	0.86664254	2.62035955	5.72096153	10.79362466	18.68309708	
		偏差	0.00000000	0.00026342	0.00061903	0.00106625	0.00160706	0.00224472	
		x	2.20000000	2.40000000	2.60000000	2.80000000	3.00000000		
		近似值	30.52159814	47.83236583	72.63450354	107.60885199	156.29825744		
		真实值	30.52458129	47.83619262	72.63928396	107.61470115	156.30529585		
		偏差	0.00298315	0.00382679	0.00478042	0.00584916	0.00703841		
	20	x	1.00000000	1.10000000	1.20000000	1.30000000	1.40000000	1.50000000	1.60000000
		近似值	0.00000000	0.34591029	0.86662169	1.60718135	2.62031131	3.96760190	5.72087932
		真实值	0.00000000	0.34591988	0.86664254	1.60721508	2.62035955	3.96766629	5.72096153
		偏差	0.00000000	0.00000959	0.00002084	0.00003373	0.00004825	0.00006440	0.00008220
		x	1.70000000	1.80000000	1.90000000	2.00000000	2.10000000	2.20000000	2.30000000
		近似值	7.96377179	10.79350178	14.32293573	18.68292657	24.02498942	30.52435589	38.38345866
		真实值	7.96387348	10.79362466	14.32308154	18.68309708	24.02518645	30.52458129	38.38371431
		偏差	0.00010169	0.00012288	0.00014581	0.00017051	0.00019703	0.00022540	0.00025565
		x	2.40000000	2.50000000	2.60000000	2.70000000	2.80000000	2.90000000	3.00000000
		近似值	47.83590478	59.15100383	72.63892578	88.65657333	107.61426439	129.98333312	156.30477188
		真实值	47.83619262	59.15132583	72.63928396	88.65696974	107.61470115	129.98381238	156.30529585
		偏差	0.000288	0.000322	0.000358	0.000396	0.000437	0.000479	0.000524

问题2.2	N							
	5	x	1.000000	1.400000	1.800000	2.200000	2.600000	3.000000
		近似值	-2.00000000	-1.55398900	-1.38361729	-1.29340153	-1.23754016	-1.19954796
		真实值	-2.00000000	-1.55555556	-1.38461538	-1.29411765	-1.23809524	-1.20000000
		偏差	0.000000000	-0.001566557	-0.000998095	-0.000716120	-0.000555080	-0.000452042
	10	x	1.00000000	1.20000000	1.40000000	1.60000000	1.80000000	2.00000000
		近似值	-2.00000000	-1.71424518	-1.55552288	-1.45451975	-1.38459451	-1.33331586
		真实值	-2.00000000	-1.71428571	-1.55555556	-1.45454545	-1.38461538	-1.33333333
		偏差	0.000000000	-0.000040534	-0.000032671	-0.000025705	-0.000020878	-0.000017477
		x	2.20000000	2.40000000	2.60000000	2.80000000	3.00000000	
		近似值	-1.29410266	-1.26314480	-1.23808362	-1.21738087	-1.19999054	
		真实值	-1.29411765	-1.26315789	-1.23809524	-1.21739130	-1.20000000	
		偏差	-0.000014986	-0.000013096	-0.000011617	-0.000010431	-0.000009460	
	20	x	1.00000000	1.10000000	1.20000000	1.30000000	1.40000000	1.50000000	1.60000000
		近似值	-2.00000000	-1.83333283	-1.71428517	-1.62499950	-1.55555511	-1.49999961	-1.45454510
		真实值	-2.00000000	-1.83333333	-1.71428571	-1.62500000	-1.55555556	-1.50000000	-1.45454545
		偏差	0.000000000	-0.000000504	-0.000000544	-0.000000500	-0.000000445	-0.000000394	-0.000000352
		x	1.70000000	1.80000000	1.90000000	2.00000000	2.10000000	2.20000000	2.30000000
		近似值	-1.41666635	-1.38461510	-1.35714260	-1.33333309	-1.31249978	-1.29411744	-1.27777759
		真实值	-1.41666667	-1.38461538	-1.35714286	-1.33333333	-1.31250000	-1.29411765	-1.27777778
		偏差	-0.000000316	-0.000000286	-0.000000261	-0.000000240	-0.000000222	-0.000000206	-0.000000192
		x	2.40000000	2.50000000	2.60000000	2.70000000	2.80000000	2.90000000	3.00000000
		近似值	-1.26315771	-1.24999983	-1.23809508	-1.22727258	-1.21739116	-1.20833320	-1.19999987
		真实值	-1.26315789	-1.25000000	-1.23809524	-1.22727273	-1.21739130	-1.20833333	-1.20000000
		偏差	-0.000000180	-0.000000169	-0.000000160	-0.000000151	-0.000000143	-0.000000136	-0.000000130

2. 对实验 2， N 越大越精确吗？试加以说明。

答：

有实验二数据可知：N 越大越精确。

原因：四阶 RK 方法有 4 阶精度，即其局部截断误差$T(n) = O(h^{4+1})$.当 N 越大，步长越短即 h 越小，误差则越小。同时 RK 方法是显式单步法。

【定理】：若单步法有 p 阶精度，且增量函数φ关于满足 Lipchitz 条件，且初值是精确的，则整体截断误差$y(x_n) - y_n = O(h^p)$。

四阶 RK 方法满足此定理，且 p=4，故此方法收敛，N 越大，h 越小，越精确。

问题三

问题3.1	N							
	5	x	0.000000	0.200000	0.400000	0.600000	0.800000	1.000000	
		近似值	0.33333333	1.76000000	8.81333333	43.68000000	217.29333333	1084.32000000	
		真实值	0.33333333	0.04610521	0.16011182	0.36000205	0.64000004	1.00000000	
		偏差	0.000000000	-1.713894787	-8.653221512	-43.319997952	-216.653333296	-1083.319999999	
	10	x	0.000000	0.100000	0.200000	0.300000	0.400000	0.500000	
		近似值	0.33333333	0.12277778	0.07925926	0.10475309	0.16658436	0.25386145	
		真实值	0.33333333	0.05511176	0.04610521	0.09082625	0.16011182	0.25001513	
		偏差	0.000000000	-0.067666017	-0.033154046	-0.013926836	-0.006472541	-0.003846321	
		x	0.600000000	0.700000000	0.800000000	0.900000000	1.000000000		
		近似值	0.36295382	0.49265127	0.64255042	0.81251681	1.00250560		
		真实值	0.36000205	0.49000028	0.64000004	0.81000001	1.00000000		
		偏差	-0.002951770	-0.002650995	-0.002550387	-0.002516803	-0.002505602		
	20	x	0.000000000	0.050000000	0.100000000	0.150000000	0.200000000	0.250000000	0.300000000
		近似值	0.33333333	0.12755208	0.05694661	0.04015706	0.04667348	0.06505464	0.09101007
		真实值	0.33333333	0.12512648	0.05511176	0.03909569	0.04610521	0.06474598	0.09082625
		偏差	0.000000000	-0.002425603	-0.001834854	-0.001061374	-0.000568269	-0.000308657	-0.000183822
		x	0.350000000	0.400000000	0.450000000	0.500000000	0.550000000	0.600000000	0.650000000
		近似值	0.12293086	0.16021366	0.20263220	0.25010166	0.30259021	0.36008591	0.42258430
		真实值	0.12280396	0.16011182	0.20254114	0.25001513	0.30250557	0.36000205	0.42250075
		偏差	-0.000126900	-0.000101835	-0.000091068	-0.000086527	-0.000084639	-0.000083862	-0.000083546
		x	0.700000000	0.750000000	0.800000000	0.850000000	0.900000000	0.950000000	1.000000000
		近似值	0.49008370	0.56258347	0.64008338	0.72258335	0.81008334	0.90258334	1.00008333
		真实值	0.49000028	0.56250010	0.64000004	0.72250001	0.81000001	0.90250000	1.00000000
		偏差	-0.000083419	-0.000083367	-0.000083347	-0.000083339	-0.000083335	-0.000083334	-0.000083334

问题3.2

N								
5	x	0.000000	0.200000	0.400000	0.600000	0.800000	1.000000	
	近似值	1.00000000	5.19733811	25.37617070	125.48681526	625.31209552	3123.79515095	
	真实值	1.00000000	0.21698497	0.38975380	0.56464862	0.71735620	0.84147099	
	偏差	0.000000000	-4.980353137	-24.986416899	-124.922166644	-624.594739314	-3122.953679960	
10	x	0.000000	0.100000	0.200000	0.300000	0.400000	0.500000	
	近似值	1.00000000	0.43313900	0.30966047	0.33232467	0.40141397	0.48307434	
	真实值	1.00000000	0.23516870	0.21698497	0.29799896	0.38975380	0.47947094	
	偏差	0.000000000	-0.197970297	-0.092675498	-0.034325708	-0.011660166	-0.003603403	
	x	0.600000000	0.700000000	0.800000000	0.900000000	1.000000000		
	近似值	0.56543528	0.64398900	0.71672235	0.78249915	0.84052572		
	真实值	0.56464862	0.64421852	0.71735620	0.78332692	0.84147099		
	偏差	-0.000786662	0.000229514	0.000633856	0.000827774	0.000945266		
20	x	0.000000000	0.050000000	0.100000000	0.150000000	0.200000000	0.250000000	0.300000000
	近似值	1.00000000	0.42497852	0.24045622	0.20216844	0.21843866	0.25481165	0.29829102
	真实值	1.00000000	0.41785861	0.23516870	0.19922520	0.21698497	0.25414191	0.29799896
	偏差	0.000000000	-0.007119908	-0.005287522	-0.002943238	-0.001453694	-0.000669745	-0.000292063
	x	0.350000000	0.400000000	0.450000000	0.500000000	0.550000000	0.600000000	0.650000000
	近似值	0.34392855	0.38979534	0.43509617	0.47946262	0.52268809	0.56462864	0.60516599
	真实值	0.34380969	0.38975380	0.43508894	0.47947094	0.52270393	0.56464862	0.60518867
	偏差	-0.000118862	-0.000041531	-0.000007229	0.000008316	0.000015843	0.000019979	0.000022680
	x	0.700000000	0.750000000	0.800000000	0.850000000	0.900000000	0.950000000	1.000000000
	近似值	0.64419376	0.68161253	0.71732804	0.75125076	0.78329581	0.81338305	0.84143727
	真实值	0.64421852	0.68163907	0.71735620	0.75128045	0.78332692	0.81341551	0.84147099
	偏差	0.000024756	0.000026540	0.000028166	0.000029683	0.000031112	0.000032457	0.000033718

问题3.3

N								
5	x	0.000000	0.200000	0.400000	0.600000	0.800000	1.000000	
	近似值	0.00000000	0.29864621	0.92721987	2.83547734	10.71088533	47.94144638	
	真实值	0.00000000	0.24265527	0.58094390	1.02884567	1.59650534	2.28735529	
	偏差	0.000000000	-0.055990944	-0.346275969	-1.806631673	-9.114379990	-45.654091094	
10	x	0.000000	0.100000	0.200000	0.300000	0.400000	0.500000	
	近似值	0.00000000	0.11205511	0.24511651	0.40177810	0.58409696	0.79382205	
	真实值	0.00000000	0.11033299	0.24265527	0.39891055	0.58094390	0.79043908	
	偏差	0.000000000	-0.001722120	-0.002461246	-0.002867543	-0.003153056	-0.003382970	
	x	0.600000	0.700000	0.800000	0.900000	1.000000		
	近似值	1.03241831	1.30101499	1.60032101	1.93052103	2.29115692		
	真实值	1.02884567	1.29729511	1.59650534	1.92667330	2.28735529		
	偏差	-0.003572639	-0.003719876	-0.003815671	-0.003847730	-0.003801636		
20	x	0.000000	0.050000	0.100000	0.150000	0.200000	0.250000	0.300000
	近似值	0.00000000	0.05259504	0.11040899	0.17370939	0.24274900	0.31777169	0.39901355
	真实值	0.00000000	0.05254166	0.11033299	0.17362234	0.24265527	0.31767297	0.39891055
	偏差	0.000000000	-0.000053384	-0.000075998	-0.000087051	-0.000093732	-0.000098720	-0.000102999
	x	0.350000	0.400000	0.450000	0.500000	0.550000	0.600000	0.650000
	近似值	0.48670207	0.58105449	0.68227577	0.79055629	0.90606933	1.02896834	1.15938414
	真实值	0.48659515	0.58094390	0.68216175	0.79043908	0.90594922	1.02884567	1.15925927
	偏差	-0.000106919	-0.000110588	-0.000114025	-0.000117210	-0.000120108	-0.000122678	-0.000124872
	x	0.700000	0.750000	0.800000	0.850000	0.900000	0.950000	1.000000
	近似值	1.29742175	1.44315720	1.59663402	1.75785967	1.92680163	2.10338342	2.28748035
	真实值	1.29729511	1.44302927	1.59650534	1.75773083	1.92667330	2.10325633	2.28735529
	偏差	-0.000126641	-0.000127930	-0.000128682	-0.000128836	-0.000128330	-0.000127096	-0.000125063

3. 对实验 3， N 较小会出现什么现象？试加以说明

答：

由实验三数据可知：N 较小时，越往后递推所得近似值误差越大。

原因：

由于四阶 RK 方法是显式单步法，其稳定性与步长h有关。已知递推方程① $y_{n+1}=y_n+h\varphi(x_n,y_n,h)$，而当存在一个扰动即$\widetilde{y_n}=y_n+\delta_n$时，有②$\widetilde{y_{n+1}}=\widetilde{y_n}+h\varphi(x_n,\widetilde{y_n},h)$。②-①可得$\frac{\delta_{n+1}}{\delta_n}=1+h\frac{\partial\varphi}{\partial y}$,为保证稳定性，就需要取合适的$h$使得$\left|\frac{\delta_{n+1}}{\delta_n}\right|<1$.对于四阶 RK 方法，当 N 较小，h 较大时，不满足此条件。就会使误差逐步放大，变得不稳定。