实验序号： 实验 4　　　　**实验项目：** 数值微积分与解微分方程

实验成绩： ________　　　　**教师签名：** ________

西 南 大 学 人 工 智 能 学 院
实 验 报 告

学年学期	2023 年秋季学期
课程名称	数值分析
姓　　名	洪浩钦
学　　号	222021335210144
学　　院	含弘学院
专　　业	智能科学与技术
班　　级	袁隆平班
任课教师	钟秀蓉

2023 年 **12** 月 **20** 日

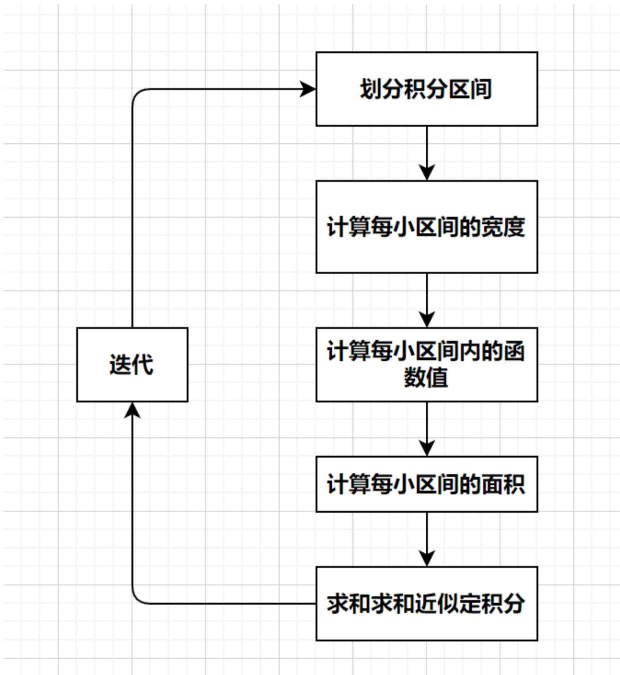
一、实验目的

1. **知识培养目标：** 致力于深入掌握数值微积分的理论基础，包括理解其核心原理和推导方法。同时，重点学习常微分方程的数值解法，确保能够理解和运用这些方法来解决具体问题。
2. **技能提升目标：** 通过实践操作，掌握使用 MATLAB 软件进行数值微积分计算的技巧。这不仅包括基本的操作方法，还包括如何有效地应用 MATLAB 解决常微分方程，以及如何利用这些工具进行复杂问题的数值分析。
3. **素质发展目标：** 培养能够将理论知识应用于实际的能力，特别是在分析和建模现实世界中的微积分问题和常微分方程时。强调理论与实践的结合，通过实际案例学习如何运用所学知识进行问题的求解、结果的验证，并在实际领域中找到相应的应用场景和解决方案。

二、算法原理概述

（一）梯形规则

采用梯形规则进行定积分的近似计算。将区域划分为等宽小区间,利用每个小区间与函数值的乘积求和近似整个区域下的定积分。



1.划分积分区间：

将求积分的整个区间[a,b]均匀地划分为 n 个小同宽子区间。

2.计算每小区间的宽度:

计算每个子区间的宽度 h 为(b-a)/n。

3.计算每小区间内的函数值:

在每个子区间内选取一点作为代表,如左米点或中心点,计算在该点的函数值。

4.计算每小区间的面积：

每个子区间近似看作一个梯形,其面积可看作函数值乘以宽度 h。

5.求和求和近似定积分：

将所有子区间的面积求和,作为定积分从 a 到 b 的近似值:

$$\int_{a}^{b} f(x)\,dx \approx h(f(x_1)/2 + f(x_2) + ... + f(x_{n-1}) + f(x_n)/2)$$

6.加以迭代：

可以重复上述过程，每次重复将区间细分数量 n 增加一倍，逼近真实定积分值。

7. 优化选点方法

如采用左端点、右端点或中心点不同会影响精度，选择精度高的点来代替区间内的函数值。

（二）常微分方程的数值解法

1. 初始化：设置初始条件和时间范围。
2. 选择求解器：基于方程的特性（如刚性或非刚性），选择合适的求解器。
3. 设置参数：如精度要求、步长、求解器选项等。
4. 求解循环：在指定的时间范围内，循环计算每一步的解。
5. 误差控制和步长调整：基于误差估计调整步长，以达到预定的精度。
6. 输出结果：得到在整个时间范围内的解。

三、软件开发环境及工具

MATLAB 2021b

四、实验内容

1、问题提出　　2、解决思路　　3、算法步骤

4、结果分析　　5、实验核心代码

数值微积分实验 1 用 MATLAB 指令计算下列积分

(4)

$$\int_0^{2\pi} exp(2x)sin^2(x)dx$$

1. 实验代码

（1） 自定义梯形规则函数

```
function result = my_integral(func, a, b)
    % 这个函数使用梯形规则计算在a到b之间的定积分
    % func是一个函数句柄，表示你要积分的函数
    % a和b是积分的下限和上限
    % n是分割的区间数

    n = 1000;
    h = (b - a) / n; % 计算每个小区间的宽度
    x = a:h:b; % 创建x的值
    y = func(x); % 计算每个x的函数值
    result = h * (0.5*y(1) + sum(y(2:end-1)) + 0.5*y(end)); % 使用梯形规则计算积分
end
```

（2）求解积分

```
% 实验1 用 MATLAB 指令计算下列积分
% （4）\int_{0}^{2\pi} exp(2x)sin^2(x)dx

% 定义函数
f = @(x) exp(2*x) .* (sin(x).^2);

% 计算积分
result = my_integral(f, 0, 2*pi);

% 显示结果
disp(result);
```

2. 实验结果

```
>> exp1
   3.5844e+04
```

数值微积分实验 6 计算积分

（1） $\int_{\infty}^{\infty} \frac{exp(-x^2)}{(1+x^2)} dx$

1. 实验代码

```
% 实验6 计算积分
% （1）\int_{\infty}^{\infty} exp(-x^2) / (1 + x^2) dx

% 定义函数
f = @(x) exp(-x.^2) ./ (1 + x.^2);

% 计算积分
result = integral(f, -inf, inf);

% 显示结果
disp(result);
```

2. 实验结果

```
>> exp2
    1.3433
```

装订线

装 订 线

数值微积分实验 6 计算积分

（4）

$$\iint\limits_{D}(1+x+y^2)dydx, D \text{ 为 } x^2+y^2 <= 2x$$

1. 实验代码

```
% 实验6 计算积分
% \iint\limits_{D}(1 + x + y^2)dydx, D 为 x^2 + y^2 <= 2x

% 定义函数
f = @(x, y) 1 + x + y.^2;

% 定义积分区域的边界
xmin = 0;
xmax = 2;
ymin = @(x) -sqrt(2*x - x.^2);
ymax = @(x) sqrt(2*x - x.^2);

% 计算积分
result = integral2(f, xmin, xmax, ymin, ymax);

% 显示结果
disp(result);
```

2. 实验结果

```
>> exp3
    7.0686
```

常微分方程的数值解法实验 1

用 ode45，ode23 和 ode113 解下列微分方程：

（2） x’ = 2x + 3y, y’ = 2x + y, x(0) = -2.7, y(0) = 2.8, 0 < t < 10, 作相平面图。

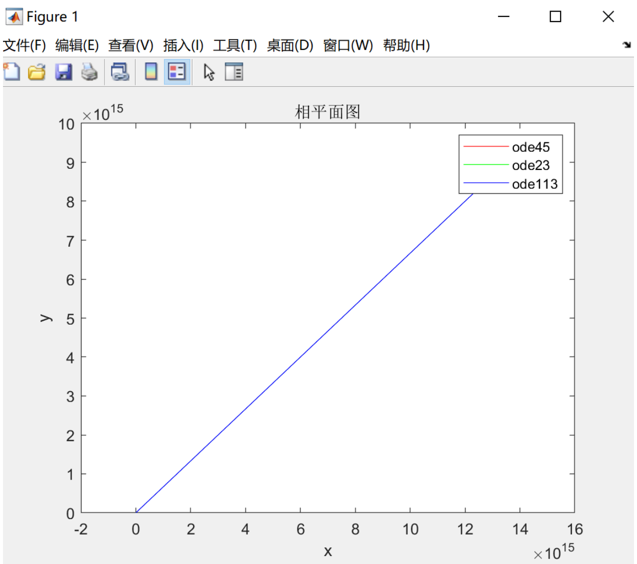
1. 实验代码

```
%  用ode45，ode23和ode113解下列微分方程：
% （2）x’= 2x + 3y, y’= 2x + y, x(0) = -2.7, y(0) = 2.8, 0 < t < 10,作相平面图。

% 定义微分方程作为匿名函数
myODE = @(t, y) [2*y(1) + 3*y(2); 2*y(1) + y(2)];

% 初始条件
```

```
y0 = [-2.7; 2.8];

% 时间范围
tspan = [0 10];

% 使用不同的求解器求解ODE
[t1, y1] = ode45(myODE, tspan, y0);
[t2, y2] = ode23(myODE, tspan, y0);
[t3, y3] = ode113(myODE, tspan, y0);

% 绘制相平面图
figure;
plot(y1(:,1), y1(:,2), 'r'); % ode45的结果
hold on;
plot(y2(:,1), y2(:,2), 'g'); % ode23的结果
hold on;
plot(y3(:,1), y3(:,2), 'b'); % ode113的结果
xlabel('x');
ylabel('y');
title('相平面图');
legend('ode45', 'ode23', 'ode113');
```

装 订 线

2. 实验结果



常微分方程的数值解法实验 3

分别用 ode45 和 ode15s 求解刚性方程组，并比较计算效率。

$$\begin{aligned} y_1' &= -1000.25y_1 + 999.75y_2 + 0.5, y_1(0) = 1, \\ y_2' &= 999.75y_1 - 1000.25y_2 + 0.5, y_2(0) = -1, \end{aligned} \quad 0 < x < 50$$

装 订 线

1. 实验代码

```
% 分别用ode45和ode15s求解刚性方程组，并比较计算效率。
% {y}'_1 & = -1000.25y_1 + 999.75y_2 +0.5, y_1(0) = 1,
% {y}'_2 & = 999.75y_1 - 1000.25y_2 + 0.5, y_2(0) = -1,
% 0 < x < 50

% 定义微分方程
stiffODE = @(t, y) [-1000.25*y(1) + 999.75*y(2) + 0.5; 999.75*y(1) - 1000.25*y(2) + 0.5];

% 初始条件
y0 = [1; -1];

% 时间范围
tspan = [0 50];

% 使用ode45求解微分方程
tic; % 开始计时
[t_ode45, y_ode45] = ode45(stiffODE, tspan, y0);
t_ode45_elapsed = toc; % 结束计时

% 使用ode15s求解微分方程
tic; % 开始计时
[t_ode15s, y_ode15s] = ode15s(stiffODE, tspan, y0);
t_ode15s_elapsed = toc; % 结束计时

% 输出计算时间
fprintf('ode45耗时: %f秒\n', t_ode45_elapsed);
fprintf('ode15s耗时: %f秒\n', t_ode15s_elapsed);
```

2. 实验结果

```
>> exp5
ode45 耗时：0.213671 秒
ode15s 耗时：0.145927 秒
```

常微分方程的数值解法实验 6

考虑用欧拉格式、隐式欧拉格式、中点欧拉格式、改进欧拉格式、4 阶经典龙格-库塔格式求解微分方程

$$y' = -50y,$$

取步长 h = 0.1，0.05，0.025，0.001，试验其稳定性。

装 订 线

1. 实验代码

（1）欧拉格式

```
function [x, y] = naeuler(dyfun, xspan, y0, h)
% 用途：欧拉法解常微分方程 y' = f(x, y), y(x0) = y0
% 格式：[x, y] = naeuler(dyfun, xspan, y0, h)。其中，dyfun为函数f(x, y),
%   xspan为求解区间[x0, xN], y0为初值y(x0), h为步长，x返回节点，y返回数值解
x = xspan(1):h:xspan(2);
y(1) = y0;
for n = 1: length(x) - 1
        y(n + 1) = y(n) + h * dyfun(x(n), y(n));
end
x = x'; y = y';
```

（2）隐式欧拉格式

```
function [x, y] = naeulerb(dyfun, xspan, y0, h)
% 用途：隐式欧拉法解常微分方程 y' = f(x, y), y(x0) = y0
% 格式：[x, y] = naImplicitEuler(dyfun, xspan, y0, h)。其中，
% dyfun为函数f(x, y), xspan为求解区间[x0, xN], y0为初值y(x0), h为步长，
% x返回节点，y返回数值解

x = xspan(1):h:xspan(2);
y = zeros(size(x));
y(1) = y0;

for n = 1:length(x) - 1
        % 使用匿名函数定义隐式方程
        implicitEq = @(ynext) ynext - y(n) - h * dyfun(x(n+1), ynext);

        % 使用牛顿法或其他根求解方法求解隐式方程
        y(n + 1) = fsolve(implicitEq, y(n)); % 初始猜测使用 y(n)
end

x = x'; y = y';
end
```

（3） 中点欧拉格式

```
function [x, y] = midpointeuler(dyfun, xspan, y0, h)
% 用途：中点欧拉法解常微分方程 y' = f(x, y), y(x0) = y0
% 格式：[x, y] = midpointEuler(dyfun, xspan, y0, h)。
% 其中，dyfun为函数f(x, y), xspan为求解区间[x0, xN], y0为初值y(x0), h为步长，
% x返回节点，y返回数值解
        x = xspan(1):h:xspan(2);
```

```
    y = zeros(1, length(x));
    y(1) = y0;
    for n = 1:length(x) - 1
        y_mid = y(n) + (h / 2) * dyfun(x(n), y(n));
        y(n + 1) = y(n) + h * dyfun(x(n) + h / 2, y_mid);
    end
    x = x'; y = y';
end
```

（4） 改进欧拉格式

```
function [x, y] = improvedeuler(dyfun, xspan, y0, h)
% 用途：改进欧拉法解常微分方程 y' = f(x, y), y(x0) = y0
% 格式：[x, y] = improvedEuler(dyfun, xspan, y0, h)。
% 其中，dyfun为函数f(x, y), xspan为求解区间[x0, xN], y0为初值y(x0), h为步长，
% x返回节点，y返回数值解
    x = xspan(1):h:xspan(2);
    y = zeros(1, length(x));
    y(1) = y0;
    for n = 1:length(x) - 1
        y_pred = y(n) + h * dyfun(x(n), y(n));
        y(n + 1) = y(n) + (h / 2) * (dyfun(x(n), y(n)) + dyfun(x(n + 1), y_pred));
    end
    x = x'; y = y';
end
```

（5）4 阶经典龙格-库塔格式

```
function [x, y] = rungeKutta4(dyfun, xspan, y0, h)
% 用途：4阶Runge-Kutta法解常微分方程 y' = f(x, y), y(x0) = y0
% 格式：[x, y] = rungeKutta4(dyfun, xspan, y0, h)。
% 其中，dyfun为函数f(x, y), xspan为求解区间[x0, xN], y0为初值y(x0), h为步长，
% x返回节点，y返回数值解
    x = xspan(1):h:xspan(2);
    y = zeros(1, length(x));
    y(1) = y0;
    for n = 1:length(x) - 1
        k1 = dyfun(x(n), y(n));
        k2 = dyfun(x(n) + h / 2, y(n) + h / 2 * k1);
        k3 = dyfun(x(n) + h / 2, y(n) + h / 2 * k2);
        k4 = dyfun(x(n) + h, y(n) + h * k3);
        y(n + 1) = y(n) + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
    end
    x = x'; y = y';
```

装 订 线

```
end
```

（6）常微分的数值求解

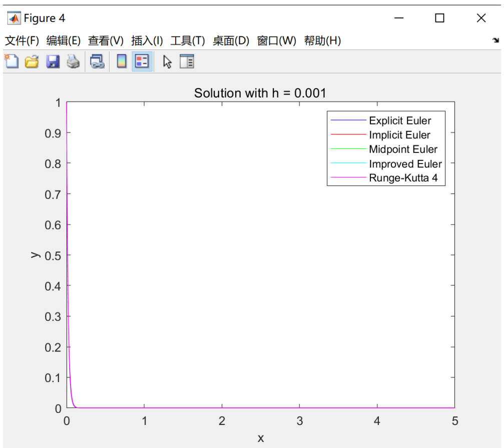
```
% 考虑用欧拉格式、隐式欧拉格式、中点欧拉格式、改进欧拉格式、4阶经典龙格-库塔格式求解微分方程
% y'= -50y,
% 取步长h = 0.1， 0.05， 0.025， 0.001， 试验其稳定性。

dyfun = @(x, y) -50*y;
xspan = [0, 5];
y0 = 1; % 初始条件

h_values = [0.1, 0.05, 0.025, 0.001];
for h = h_values
    % 调用不同的求解方法
    [x_explicit, y_explicit] = naeuler(dyfun, xspan, y0, h);
    [x_implicit, y_implicit] = naeulerb(dyfun, xspan, y0, h);
    [x_midpoint, y_midpoint] = midpointeuler(dyfun, xspan, y0, h);
    [x_improved, y_improved] = improvedeuler(dyfun, xspan, y0, h);
    [x_rungeKutta, y_rungeKutta] = rungeKutta4(dyfun, xspan, y0, h);

    % 绘图以比较结果
    figure;
    plot(x_explicit, y_explicit, 'b', x_implicit, y_implicit, 'r', ...
         x_midpoint, y_midpoint, 'g', x_improved, y_improved, 'c', ...
         x_rungeKutta, y_rungeKutta, 'm');
    legend('Explicit Euler', 'Implicit Euler', 'Midpoint Euler', ...
           'Improved Euler', 'Runge-Kutta 4');
    title(['Solution with h = ', num2str(h)]);
    xlabel('x');
    ylabel('y');
end
```

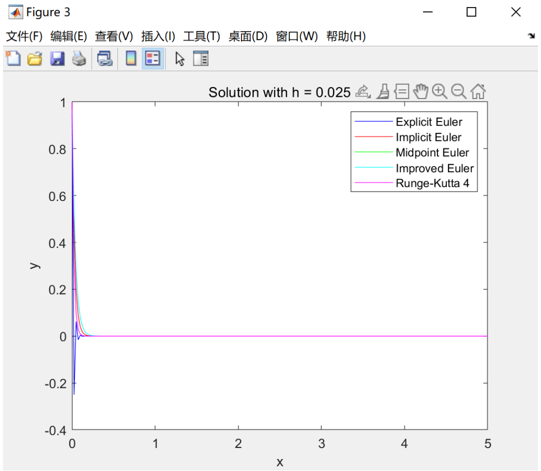
装

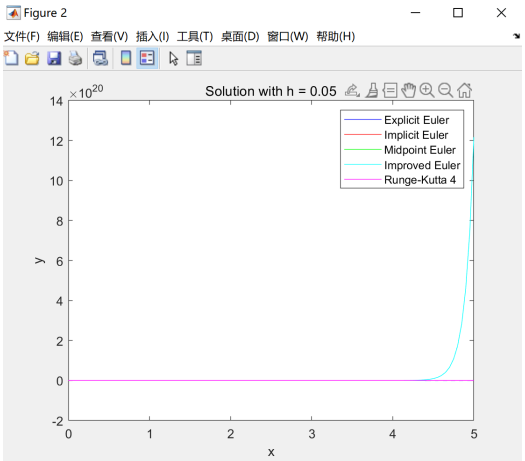
订

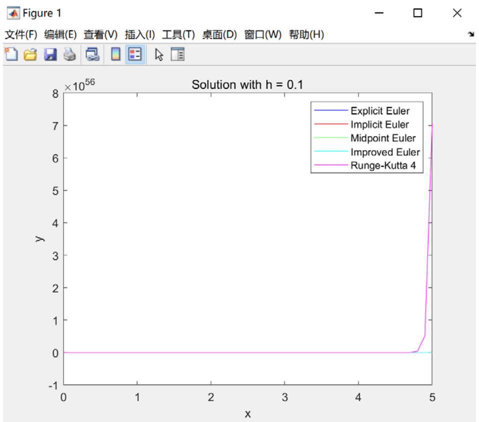
线

2. 实验结果









五、实验总结

在数值微积分部分，我探索了不同数值积分方法的原理和应用。通过实验，我理解了如何使用数值方法近似计算微积分问题，包括矩形法、梯形法和辛普森法等。这些方法的实现和比较加深了我对数值近似和误差分析的理解，使我认识到在复杂的实际问题中选择合适的数值积分方法的重要性。

在常微分方程的数值解法部分，我通过编程实践学习了欧拉法、改进欧拉法、中点欧拉法和 4 阶 Runge-Kutta 方法。这些方法在不同的步长下展示了不同的稳定性和精确度。特别是 4 阶 Runge-Kutta 方法，在大多数情况下表现出了高精度。实践中，我学会了如何根据特定问题的需求选择和应用最合适的数值方法。