

长 沙 理 工 大 学

数学与统计学院

实 验 报 告

(第二版)

**课 程 名 称**

**任 课 教 师**

**实验项目名称**

**实 验 类 型**

**实 验 日 期**

**班 级**

**学 号**

**姓 名**

**实验报告填写说明**

**（实验项目名称、实验项目类型必须与实验教学大纲保持一致）**

**1．实验环境**：

详细书写实验使用的硬件和软件环境。

**2．实验目的**：

根据实验教学大纲，写出实验的要求和目的。

**3．实验原理：**

简要说明本实验项目所涉及的理论知识。

**4．实验内容**：

这是实验报告极其重要的容。对于验证性验，要写清楚操作方法，需要经过哪几个步骤来实现其操作。对于设计性和综合性实验，还应写出设计思路和设计方法。对于创新性实验，还应注明其创新点。

**5．实验结论：**

根据实验过程中得到的结果，做出结论。

**6．实验总结：**

本次实验的收获、体会和建议。

**7．指导教师评语及成绩：**

指导教师依据学生的实际报告内容，给出本次实验报告的评价和成绩。

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1 实验环境**  硬件：Core i7-8750H  软件：Matlab 2020a  **2 实验目的**  掌握求解常微分方程的欧拉法，Runge-Kutta方法（二阶的预估校正法，经典的四阶R-K）  **3实验原理**  欧拉法，预估校正法，经典的四阶龙格库塔方法的格式  具体公式原理如下：  **欧拉法**  将区间进行N等分：，步长  .并将其写成等价的积分形式，再对其右端积分用矩形公式计算，后在右端取，舍去余项。则得 作为的近似值，以此类推， ，为欧拉法计算公式  **预估校正法**  先用欧拉法求得一个近步的近似值，称之为预测值，预测值的精度可能很差，再用梯形公式校正一次，将其称为改进的欧拉公式    **四阶龙格-库塔方法**  当用点处的斜率近似值K1与右端点处的斜率K2的算术平均值作为平均斜率K\*的近似值，那么就会得到二阶精度的改进拉格朗日中值定理,以此类推，如果在区间内多预估几个点上的斜率值K1、K2、……Km，并用他们的加权平均数作为平均斜率K\*的近似值，显然能构造出具有很高精度的高阶计算公式。经数学推导、求解，可以得出四阶龙格－库塔公式    **4实验内容**  **a)用欧拉法，预估校正法，经典的四阶龙格库塔法求解下列ODE问题：**  例题：在区间[0,1]上以h=0.1用欧拉法，预估校正法，经典的四阶龙格库塔法求解微分方程，初值y(0)= 1;其精确解为 ，且计算结果与精确解进行比较，对三个算法的收敛性的进行分析比较。  **b)用欧拉法，预估校正法，经典的四阶龙格库塔法求解初值问题**  初值y(0)=1,比较三种算法的精度；并计算结果与精确解比较。  考虑在区间[0,1]上分别取步长h=0.1;0.05时进行计算，对三个算法的收敛性进行比较。  **5实验结论**  **b)各方法结果如下：（从左到右每列依次代表x,精确值，欧拉法结果，预估校正法结果，四阶龙格库塔法结果）**  步长为0.1时：  0 1.0000 1.0000 1.0000 1.0000  0.1000 0.9094 0.9000 0.9095 0.9094  0.2000 0.8351 0.8190 0.8354 0.8351  0.3000 0.7742 0.7535 0.7745 0.7742  0.4000 0.7239 0.7004 0.7243 0.7239  0.5000 0.6823 0.6572 0.6827 0.6823  0.6000 0.6476 0.6218 0.6480 0.6476  0.7000 0.6182 0.5925 0.6186 0.6182  0.8000 0.5931 0.5680 0.5935 0.5931  0.9000 0.5712 0.5472 0.5716 0.5712  1.0000 0.5518 0.5291 0.5521 0.5518  可见，欧拉法一般有一位有效数字，预估校正法前期有两位，但多数为一位有效数字，四阶龙格库塔有四位有效数字，以下将位数增多，为：  0 1.000000000000000 1.000000000000000  0.100000000000000 0.909361605126139 0.909361752330248  0.200000000000000 0.835105368139542 0.835105621133404  0.300000000000000 0.774155040612395 0.774155366476807  0.400000000000000 0.723945649718491 0.723946022538553  0.500000000000000 0.682346992176713 0.682347391787578  0.600000000000000 0.647597730590951 0.647598141525220  0.700000000000000 0.618248703220305 0.618249113827023  0.800000000000000 0.593114232634733 0.593114634344162  0.900000000000000 0.571230371935542 0.571230758647548  1.000000000000000 0.551819161757164 0.551819529334232  可见四阶龙格库塔有五到六位的有效数字  各方法对比图如下：    步长为0.05时：  0 1.0000 1.0000 1.0000 1.0000  0.0500 0.9524 0.9500 0.9524 0.9524  0.1000 0.9094 0.9049 0.9094 0.9094  0.1500 0.8704 0.8642 0.8704 0.8704  0.2000 0.8351 0.8274 0.8352 0.8351  0.2500 0.8031 0.7942 0.8032 0.8031  0.3000 0.7742 0.7642 0.7742 0.7742  0.3500 0.7479 0.7371 0.7479 0.7479  0.4000 0.7239 0.7126 0.7240 0.7239  0.4500 0.7022 0.6904 0.7023 0.7022  0.5000 0.6823 0.6702 0.6824 0.6823  0.5500 0.6642 0.6519 0.6643 0.6642  0.6000 0.6476 0.6351 0.6477 0.6476  0.6500 0.6323 0.6199 0.6324 0.6323  0.7000 0.6182 0.6058 0.6183 0.6182  0.7500 0.6052 0.5929 0.6053 0.6052  0.8000 0.5931 0.5810 0.5932 0.5931  0.8500 0.5818 0.5699 0.5819 0.5818  0.9000 0.5712 0.5596 0.5713 0.5712  0.9500 0.5613 0.5499 0.5613 0.5613  1.0000 0.5518 0.5408 0.5519 0.5518  可见，欧拉法一般有一位有效数字，预估校正法前期有四位，但多数为三位有效数字，四阶龙格库塔有四位有效数字，以下将位数增多，为：  0 1.000000000000000 1.000000000000000  0.050000000000000 0.952418461281340 0.952418465967355  0.100000000000000 0.909361605126139 0.909361613814992  0.150000000000000 0.870390941159840 0.870390953240715  0.200000000000000 0.835105368139542 0.835105383067443  0.250000000000000 0.803138307542386 0.803138324832131  0.300000000000000 0.774155040612395 0.774155059833101  0.350000000000000 0.747850235213985 0.747850255984008  0.400000000000000 0.723945649718491 0.723945671700794  0.450000000000000 0.702188001973478 0.702188024871379  0.500000000000000 0.682346992176713 0.682347015729994  0.550000000000000 0.664213469200535 0.664213493181885  0.600000000000000 0.647597730590951 0.647597754802708  0.650000000000000 0.632327947101781 0.632327971372959  0.700000000000000 0.618248703220305 0.618248727403883  0.750000000000000 0.605219645699425 0.605219669669864  0.800000000000000 0.593114232634732 0.593114256285721  0.850000000000000 0.581818576115204 0.581818599357603  0.900000000000000 0.571230371935542 0.571230394695511  0.950000000000000 0.561257910288345 0.561257932505646  1.000000000000000 0.551819161757163 0.551819183383615  可见四阶龙格库塔有七位的有效数字  各方法对比图如下：  2  **6实验总结（收获、体会和建议）**  根据结果可以得出，龙格库塔公式在精度上比较其他两个方法都有显著的提升，可以在较少的步长内得到更为精确地值，截断误差的精度阶数高于其余两个方法  **7指导教师评语及成绩** | | | | | |
| **评 语** | **评语等级** | | | | |
| **优** | **良** | **中** | **及格** | **不及格** |
| **实验方案设计的合理程度** |  |  |  |  |  |
| **实验结论的记录情况** |  |  |  |  |  |
| **实验总结情况** |  |  |  |  |  |
| **实验报告是否按时完成,书写是否规范（文字叙述，层次结构）** |  |  |  |  |  |
| **成 绩：**    **指导教师签名：**    **批阅日期：** | | | | | |

**附录1：源 程 序**

|  |
| --- |
| **main.m**  clc;clear  % format long  % x0 = 0;  % y0 = 1;  % xn = 1;  % h = 0.1;  % fun = @(x,y) -y+x+1;  % fun\_i = @(x) x+exp(-x);  % x0 = input('x0 = ');  % y0 = input('y0 = ');  % xn = input('xn = ');  % h = input('h = ');  % fun = input('fun = ');  % fun\_i = input('fun\_i = ');  x0 = 0;  y0 = 1;  xn = 1;  h = 0.05;  fun = @(x,y) x\*exp(-x)-y;  fun\_i = @(x) 1/2\*(x^2+2)\*exp(-x);  [x,y\_I] = I(fun\_i,x0,y0,xn,h); % 精确值  [y\_E] = Euler(fun,x0,y0,xn,h); % 欧拉法  [y\_P] = Predictor(fun,x0,y0,xn,h); % 预估校正法  [y\_R] = Runge(fun,x0,y0,xn,h); %龙格库塔法  T = [x',y\_I',y\_E',y\_P',y\_R']  % format long  % T\_2 = [x',y\_I',y\_R']  % format short  plot(x,y\_I,'k', x,y\_E,'b-.',x,y\_P,'r:',x,y\_R,'g--')  title('各方法对比')  legend('精确值','欧拉法','预估校证法','龙格库塔法')  xlabel('x')  ylabel('y\_I and y\_E and y\_P and y\_R')  **I.m**  function [x,y\_I] = I(fun\_i,x0,y0,xn,h)  % 精确解，此处为写成通用形式  n = (xn-x0)/h;  x = zeros(1,n+1);  y\_I = zeros(1,n+1);  y\_I(1) = y0;  for k = 1:n  x(k+1) = x(k)+h;  y\_I(k+1) = fun\_i(x(k+1));  end  **Euler.m**  function [y\_E] = Euler(fun,x0,y0,xn,h)  % 欧拉法 右端函数fun，初值x0,y0,端点xn,步长h  n = (xn-x0)/h; % 区间的个数  x = zeros(1,n+1);  y\_E = zeros(1,n+1);  x(1) = x0;  y\_E(1) = y0;    for i = 1:n  x(i+1) = x(i)+h;  y\_E(i+1) = y\_E(i)+h\*fun(x(i),y\_E(i));  end  **Predictor.m**  function [y\_P] = Predictor(fun,x0,y0,xn,h)  % 预估校正法 右端函数fun，初值x0,y0,端点xn,步长h  n = (xn-x0)/h; % 区间的个数  x = zeros(1,n+1);  y\_P = zeros(1,n+1);  x(1) = x0;  y\_P(1) = y0;  for i = 1:n  x(i+1) = x(i)+h;  y\_p = y\_P(i)+h\*fun(x(i),y\_P(i));  y\_c = y\_P(i)+h\*fun(x(i+1),y\_p);  y\_P(i+1) = 1/2\*(y\_p+y\_c);  end  **Runge.m**  function [y\_R] = Runge(fun,x0,y0,xn,h)  % 龙格库塔法 右端函数fun，初值x0,y0,端点xn,步长h  n = (xn-x0)/h; % 区间的个数  x = zeros(1,n+1);  y\_R = zeros(1,n+1);  x(1) = x0;  y\_R(1) = y0;    for i = 1:n  x(i+1) = x(i)+h;  k\_1 = fun(x(i),y\_R(i));  k\_2 = fun(x(i)+h/2,y\_R(i)+h/2\*k\_1);  k\_3 = fun(x(i)+h/2,y\_R(i)+h/2\*k\_2);  k\_4 = fun(x(i)+h,y\_R(i)+h\*k\_3);  y\_R(i+1) = y\_R(i)+h/6\*(k\_1+2\*k\_2+2\*k\_3+k\_4);  end |