数值分析上机报告

数学强基 DBQDSS
学号：********

2024 年 5 月 18 日

目录

1 观察高次插值多项式的 Runge 现象

1.1 题目

观察高次插值多项式的龙格现象.

给定函数 $f(x)=\frac{1}{1+25x^2}$ $(-1\leqslant x\leqslant 1)$. 取等距节点, 构造牛顿插值多项式 $N_5(x)$ 和 $N_{10}(x)$ 及三次样条插值函数 $S_{10}(x)$. 分别将三种插值多项式与 $f(x)$ 的曲线画在同一个坐标系上进行比较.

1.2 算法原理与程序框图

Newton 插值法 牛顿插值法是一种用于在给定一组数据点的情况下，通过一个多项式来逼近这些数据点的方法。它基于牛顿的差商公式。

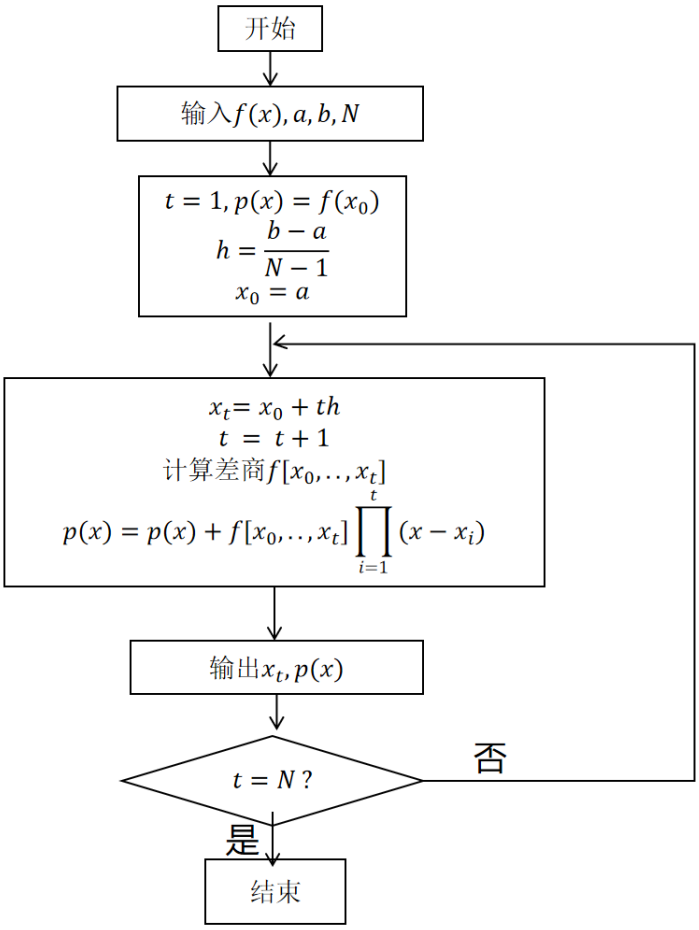
首先，我们假设有一组数据点 $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$，其中 x_i 是数据点的横坐标，y_i 是对应的纵坐标。牛顿插值法的原理是构造一个多项式，它通过这些数据点，并且可以经过这些点。这个多项式称为牛顿插值多项式。

牛顿插值多项式的一般形式为：

$$\begin{aligned}P(x)=f[x_0]+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)\\+\cdots+f[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})\end{aligned}$$

在这个表达式中，$f[x_0]$ 是数据点 (x_0,y_0) 的函数值，$f[x_0,x_1]$ 是数据点 (x_0,y_0) 和 (x_1,y_1) 的差商，$f[x_0,x_1,x_2]$ 是数据点 (x_0,y_0) (x_1,y_1) 和 (x_2,y_2) 的差商，依此类推。

牛顿插值法具有高度的灵活性和可扩展性，它可以使用任意多的数据点来逼近一个函数，并且可以在需要时动态地添加新的数据点。



三次样条插值法 三次样条插值法是一种用于在给定一组数据点的情况下，通过一条光滑曲线来逼近这些数据点的方法。它的原理是通过在相邻的数据点之间构造一组三次多项式，然后将这些多项式连接起来形成一条光滑的曲线。

首先，我们假设有一组数据点 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$，其中 x_i 是数据点的横坐标，y_i 是对应的纵坐标。

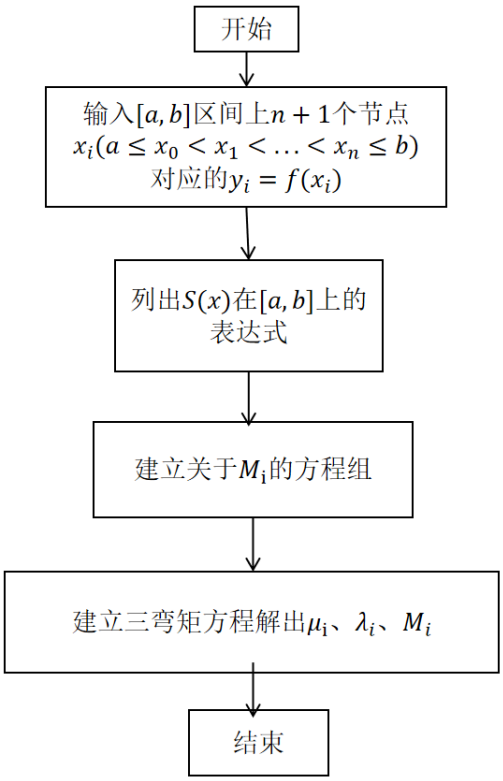
三次样条插值法的原理是将整个插值区间分割成一组小区间，并在每个小区间内构造一个三次多项式来逼近数据点。这组三次多项式称为样条函数。

要构造一个三次样条函数，首先需要满足两个条件：插值条件和光滑条

件。插值条件要求样条函数通过给定的数据点，即样条函数在每个数据点处的值与相应数据点的纵坐标值相等；光滑条件要求样条函数在相邻的数据点之间具有平滑的过渡，即样条函数在相邻数据点处的一阶导数相等。

为了满足这些条件，我们可以使用自然边界条件或固定边界条件来定义样条函数。自然边界条件要求样条函数的二阶导数在插值区间的两个端点处为零，而固定边界条件要求样条函数在插值区间的两个端点处具有固定的斜率。

一旦我们确定了边界条件，就可以使用三次多项式来逼近每个小区间内的数据点。通过解线性方程组，我们可以确定每个小区间内的三次多项式的系数，从而得到整个样条函数。



1.3 程序使用说明

Newton.py 文件得到 Newton 插值多项式

Newton_5.py 文件得到及其图象（图 1）

Newton_10.py 文件得到及其图象（图 2）
cubic.py 文件得到三次样条插值函数及其图象（图 3）
show.py 文件得到以上三个插值函数与原函数的图象（图 4）

1.4 运行结果展示

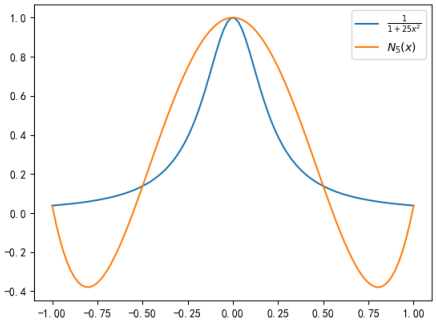


图 1: $N_5(x)$ 图象

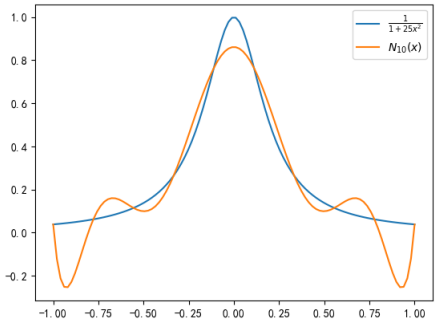


图 2: $N_{10}(x)$ 图象

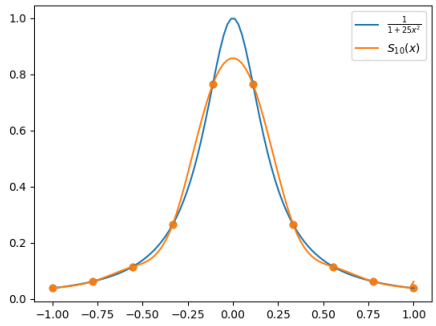


图 3: $S_{10}(x)$ 图象

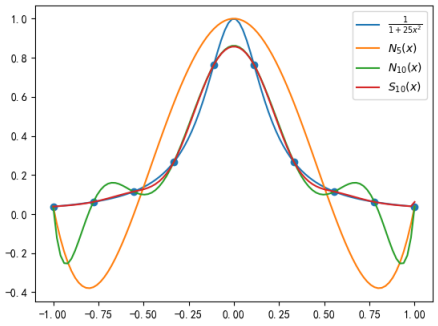


图 4: 三个插值图象与原函数图象

2 Romberg 积分法

2.1 题目

6.2 用龙贝格积分法计算习题 6.2 中的积分, 使计算结果尽可能准确.

6.2 取 $n=8$, 用复化梯形、复化辛普森求积公式计算下列积分, 保留 7 位小数计算:

(1) $\int_0^1 \frac{1}{1+x}\,\mathrm{d}x$; (2) $\int_0^1 \frac{\ln(1+x)}{1+x^2}\,\mathrm{d}x$;

(3) $\int_0^1 \frac{\ln(1+x)}{x}\,\mathrm{d}x$; (4) $\int_0^{\pi/2} \frac{\sin x}{x}\,\mathrm{d}x$.

2.2 算法原理与程序框图

Romberg 积分算法，也被称为龙贝格求积法或逐次分半加速算法，是一种用于求解数值积分的方法。它的基本原理是通过不断缩小积分区间，将区间划分为更小的子区间，并计算每个子区间的积分值，从而逼近更为准确的求积值。

具体来说，Romberg 积分算法的实现过程通常包括以下几个步骤：

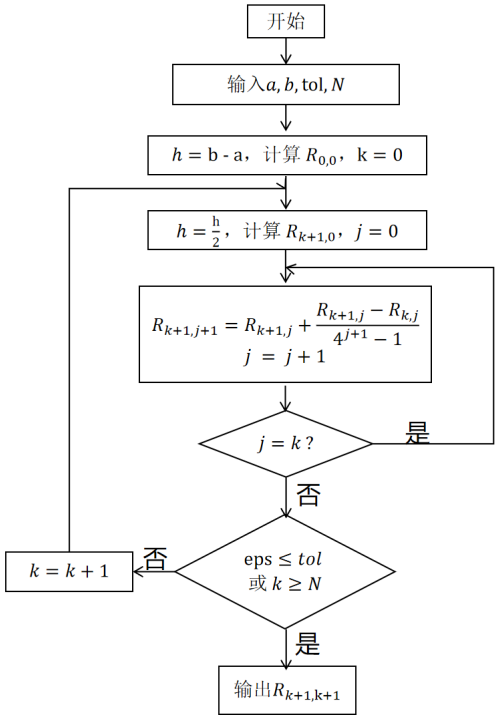
初始近似：首先，使用某种基础的数值积分方法（如梯形法、辛普森法等）对原始积分区间进行近似计算，得到一个初始的近似值。

区间分割：然后，将积分区间逐次分半，即将原始区间分成两个等长的子区间，并对每个子区间进行同样的数值积分计算。这样，可以得到一个更为精确的近似值。

迭代计算：接下来，继续将每个子区间逐次分半，并对每个更小的子区间进行数值积分计算。这样，每次迭代都会得到一个更为精确的近似值。同时，利用前一次迭代的结果，通过某种方式（如 Richardson 外推法）对当前的近似值进行修正，以提高其精度。

收敛判断：在迭代过程中，需要判断当前的近似值是否满足一定的精度要求。如果满足，则停止迭代并输出结果；否则，继续迭代计算。

Romberg 积分算法的优点在于其精度较高，且收敛速度较快。由于它采用了逐次分半的方法，因此每次迭代都可以利用前一次迭代的结果，从而提高了计算效率。此外，由于它利用了外推的思想方法，因此可以在不增加计算量的前提下提高误差的精度。



2.3 程序使用说明

Romberg.py 文件定义了含有 4 个参数的 romberg 函数（函数 f, 下限 a, 上限 b, 精度 eps）

proj1.py - proj4.py 文件分别定义四个函数 f(x) 并调用 romberg 函数进行积分的计算。其中，(1)(2) 不用做特殊处理，(3)(4) 需要补充定义 $f(0) = 1$。

2.4 运行结果展示

```
I[f] = 0.6931471806
```

图 5: 6.2(1) 运行结果

```
I[f] = 0.2721982613
```

图 6: 6.2(2) 运行结果

```
I[f] = 0.8224670334
```

图 7: 6.2(3) 运行结果

```
I[f] = 1.3707621682
```

图 8: 6.2(4) 运行结果

3 Runge-Kutta 法

3.1 题目

9.2 用标准 4 级 4 阶 R-K 法求解

$$\begin{cases} y''' = y'' + y' - y + 2x - 3\ , \\ y(0) = -1,\ y'(0) = 3,\ y''(0) = 2. \end{cases}$$

取步长 $h = 0.05$, 计算 $y(1)$ 的近似值, 并与解析解 $y(x) = x\mathrm{e}^x + 2x - 1$ 作比较.

3.2 算法原理与程序框图

对于一阶微分方程初值问题：

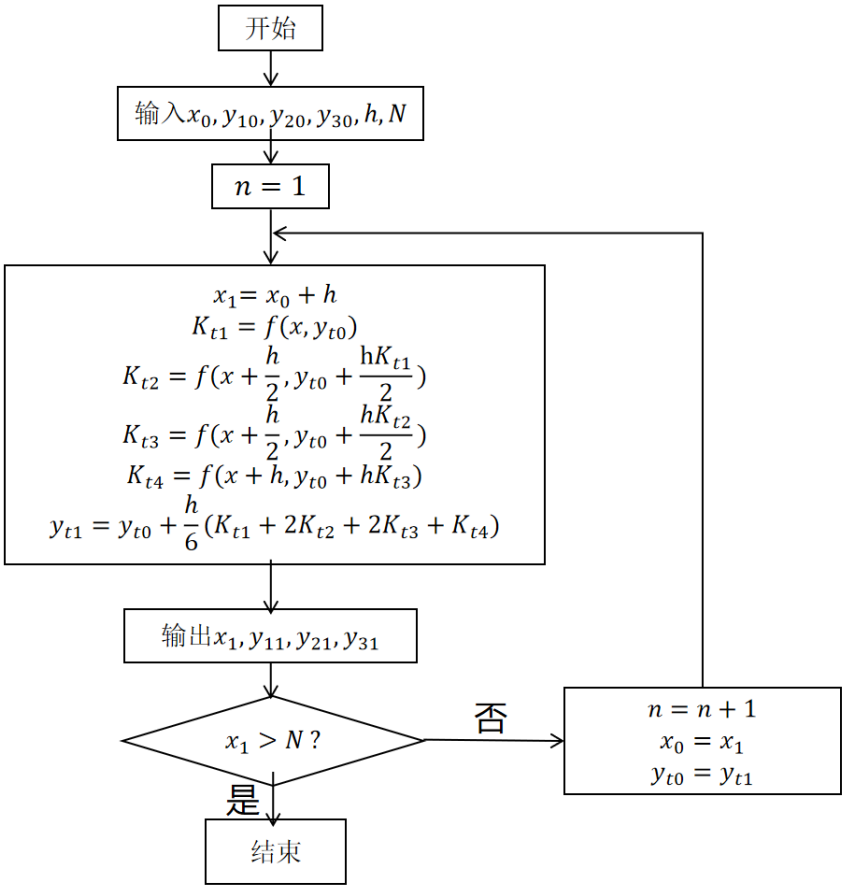
$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

四阶龙格库塔法 (Runge-Kutta) 求解算法为：

$$\begin{aligned} &已知x_0, f(x), h, y_0 = f(x_0) \\ &K_1 = f(x_k, y_k) \\ &K_2 = f(x_k + \frac{h}{2}, y_k + \frac{hK_1}{2}) \\ &K_3 = f(x_k + \frac{h}{2}, y_k + \frac{hK_2}{2}) \\ &K_4 = f(x_k + h, y_k + hK_3) \\ &y_{k+1} = y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned}$$

上式是关于 y_k 向 y_{k+1} 的递推形式，可以根据初始条件按照递推关系依次求出 $y_1, y_2, \ldots, y_N \ldots$，此离散序列即为微分方程的数值解。

微分方程的数值解法，本质是使用数值积分来实现 y' 向 y 的转换。四阶龙格库塔法通过对微分的四步分段逼近，在一个求解步长内能够逼近复杂的曲线，因此能够取得较高的计算精度，其截断误差为 $O(h^5)$。



3.3 程序使用说明

RK.py 文件得到的是数值解的图象，其中 y1,y2,y3 分别表示 y,y',y'' 在分点处的散点图（图 9）

A_s.py 文件得到的是解析解图象（图 10）

test.py 文件得到的是数值解散点图和解析解的图象（图 11）与二者差的绝对值对 10 取对数后的所得图象（图 12）

3.4 运行结果展示

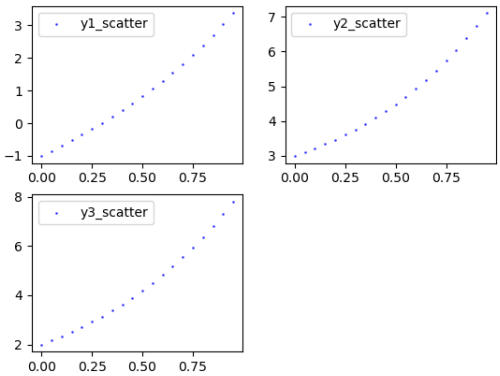


图 9: 数值解图象

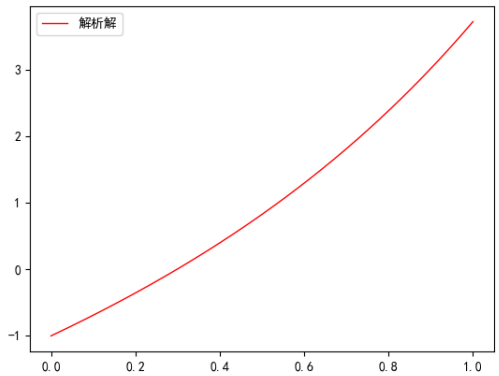


图 10: 解析解图象

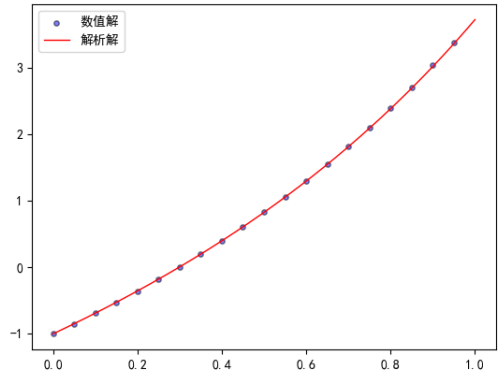


图 11: 数值解散点图和解析解的图象

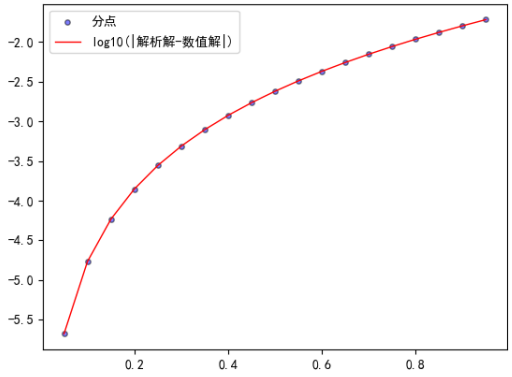


图 12: 二者差的绝对值对 10 取对数所得图象