微分方程数值解 Project3 作业报告

褚朱钇恒　　3200104144

1　运行说明

本项目需要调用`jsoncpp`与`eigen3`库，故请在运行此项目前安装好这两个包。
您可以使用以下命令进行安装:

```
sudo apt-get install libeigen3-dev
sudo apt-get install libjsoncpp-dev
```

在`project`目录下使用`make`命令即可编译运行整个项目并得到实验报告。
如果您只需要渲染文档，可以使用`make report`命令

2　程序设计思路

设计了`IVPsolver`基类，用于求解微分方程，它派生了两个类

- LMM: 线性多步法
- RKM: 龙格库塔法

前者派生了三个类,分别实现了`AdamsBashforth`,`AdamsMoulton`,`BackDifferFormula`法;三者支持

后者派生了五个类,分别实现了`classicalRK`,`DormandPrinceRK`, `ESDIRK`,`FehlbergRK`,`GaussLegendreRK`法;其中`DormandPrinceRK_solver`实现了自适应步长。

此外，本程序还实现了 classFactory 方便调用各类求解器，由于 constuctor 属性是 GNU C 编译器的一个扩展，不是标准的 C 语言特性，为了更好的移植性，没有使用该参数。

3　数值求解结果

3.1　第一部分

对于 (11.199)$k = [0.004, 0.002, 0.001, 0.0005]$ 的结果（图为 k=0.0005 的结果）如下 (为了避免迭代无法控制精度导致死循环，ESDIRK 的步长为其他的 $\frac{1}{10}$

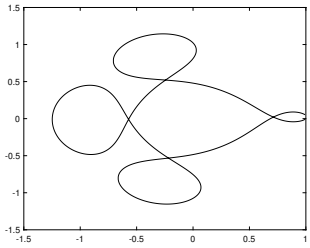

图 1: ABF

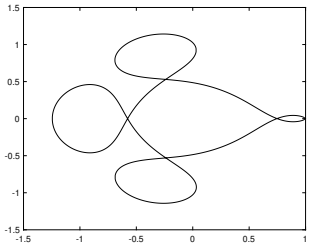

图 2: ADM

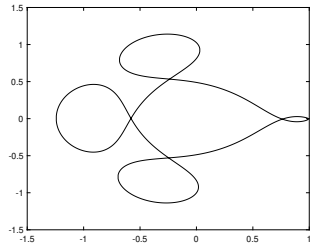

图 3: BDM

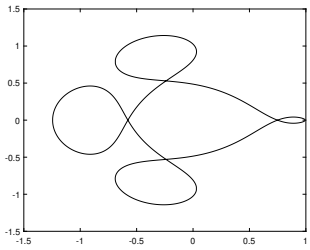

图 4: classicalRK

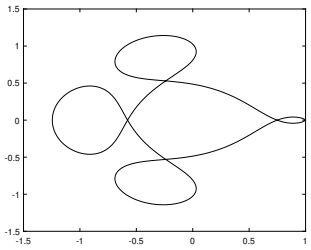

图 5: Dormand-Prince

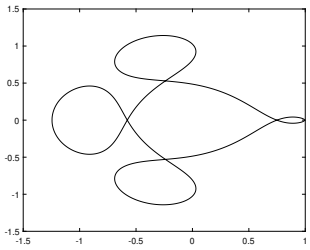

图 6: ESDIRK

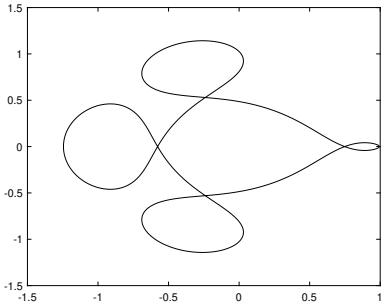

图 7: FehlbergRK

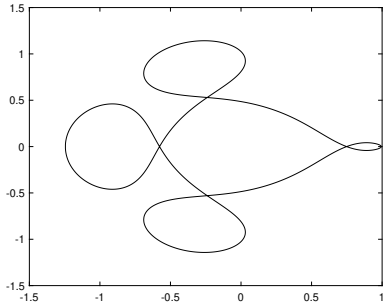

图 8: GaussLegendreRK

收敛速度和运算时间的详细信息在 data 文件夹中，由于测试组数过多，此处仅给出几个例子并画出图像。

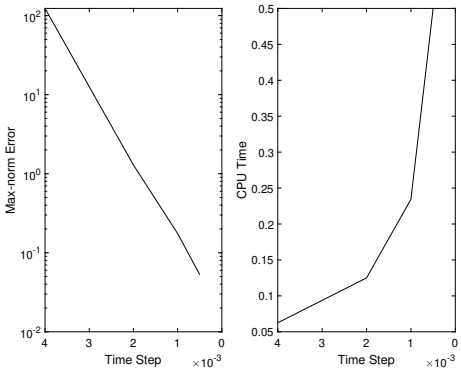

图 9: AdamsBashforth p=4

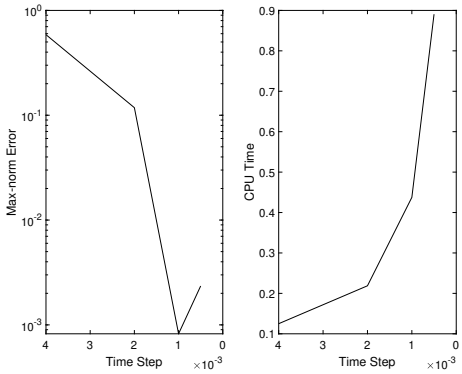

图 10: ADM p=5

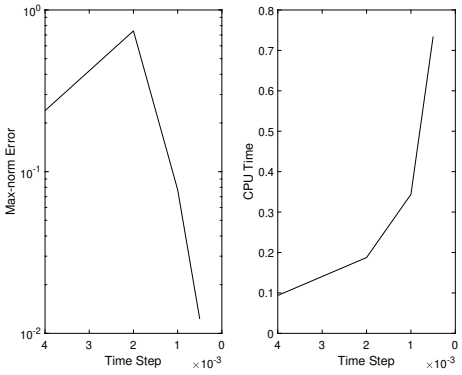

图 11: BDM p=4

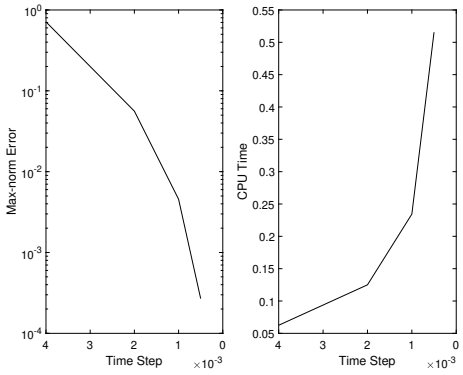

图 12: classicalRK

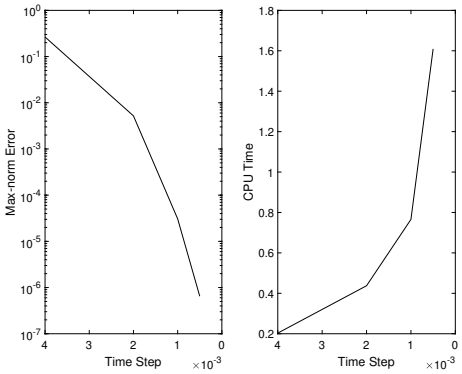

图 13: Dormand-Prince p=5

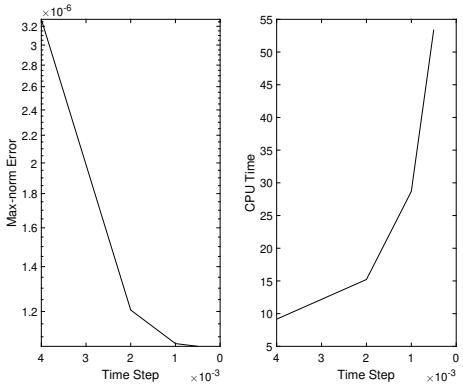

图 14: ESDIRK

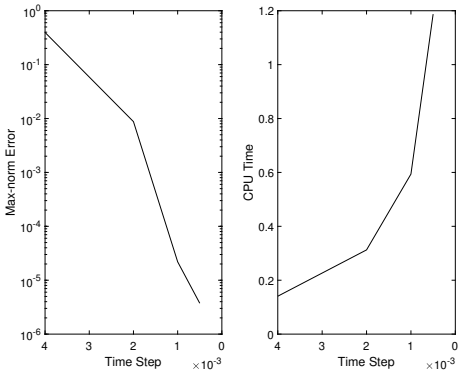

图 15: FehlbergRK p=5

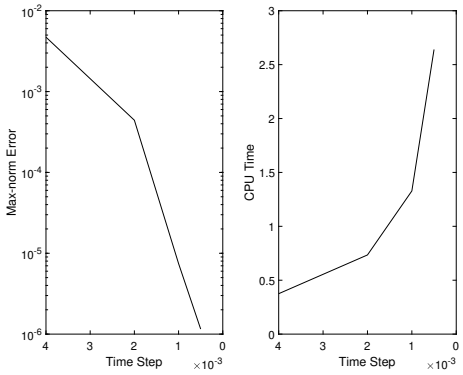

图 16: GaussLegendreRK p=3

对于 (11.200)$k = [0.004, 0.002, 0.001, 0.0005]$ 的结果（图为 k=0.0005 的结果）(为了避免迭代无法控制精度导致死循环，ESDIRK 的步长为其他的 $\frac{1}{10}$

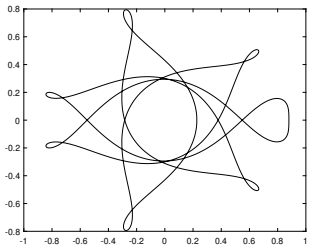

图 17: ABF

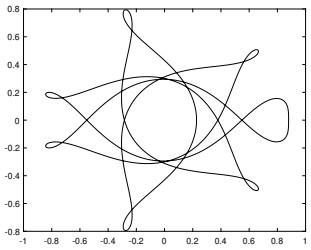

图 18: ADM

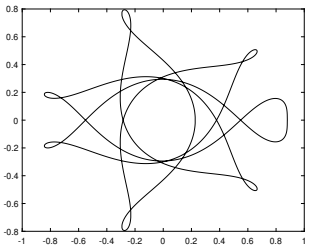

图 19: BDM

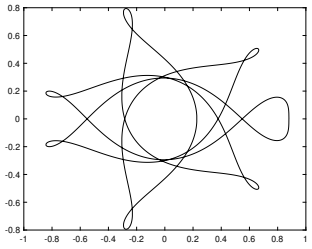

图 20: classicalRK

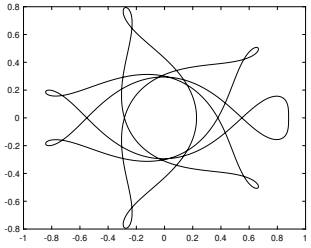

图 21: Dormand-Prince

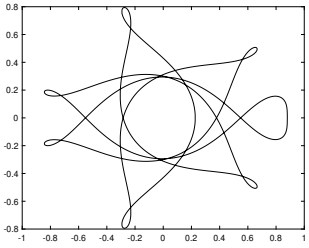

图 22: ESDIRK

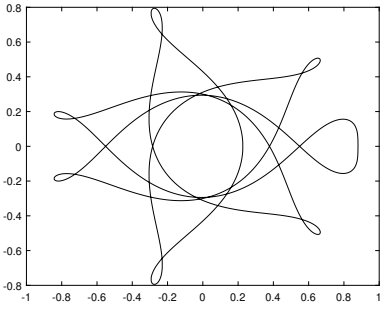

图 23: FehlbergRK

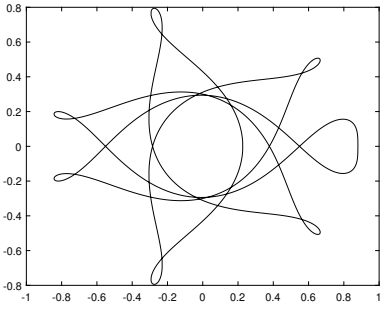

图 24: GaussLegendreRK

收敛速度和运算时间的详细信息在 data 文件夹中，由于测试组数过多，此处仅给出几个例子并画出图像。

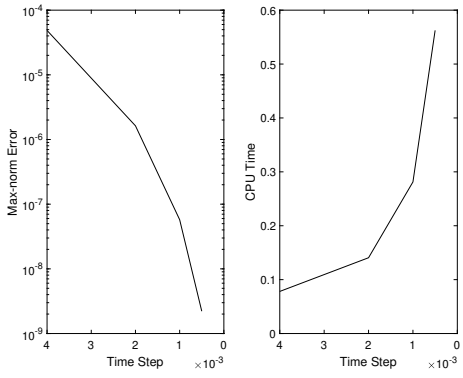

图 25: AdamsBashforth p=4

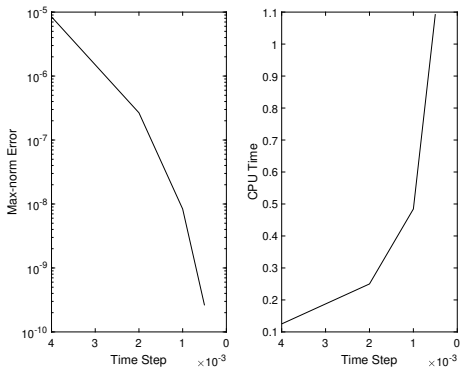

图 26: ADM p=5

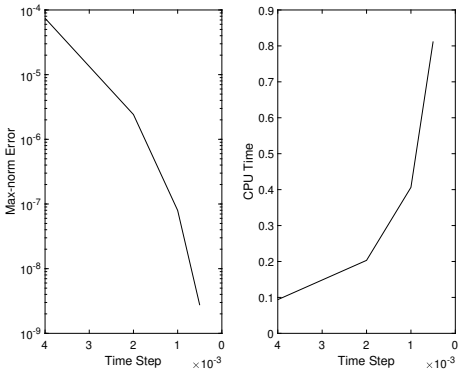

图 27: BDM p=4

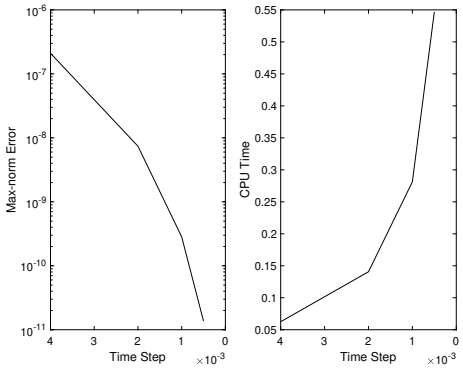

图 28: classicalRK

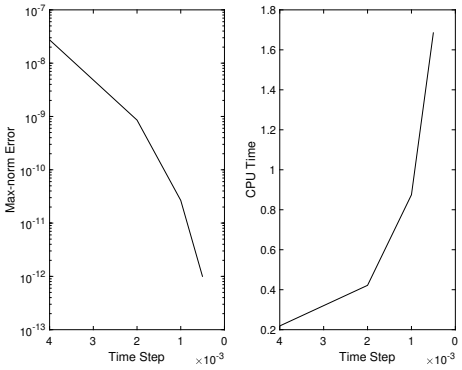

图 29: Dormand-Prince p=5

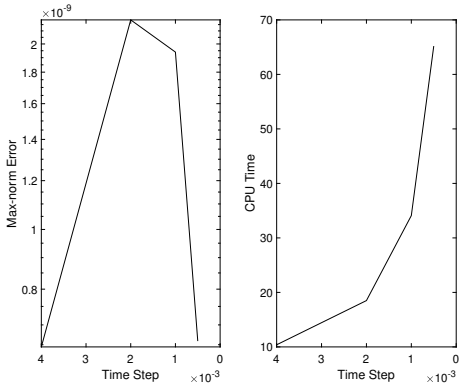

图 30: ESDIRK

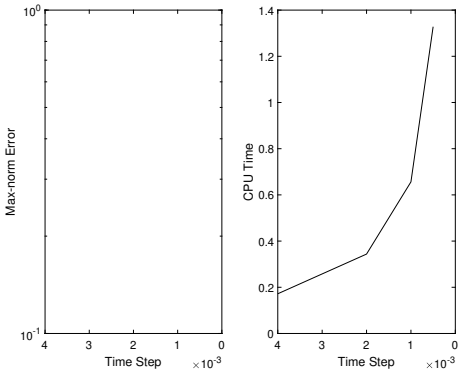

图 31: FehlbergRK p=5

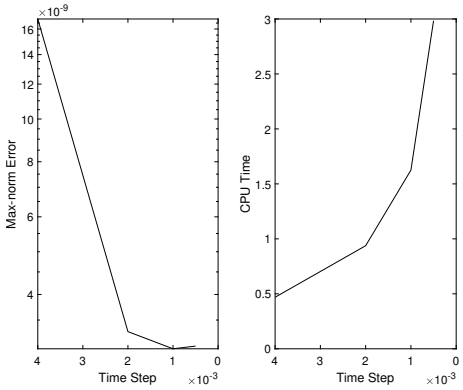

图 32: GaussLegendreRK p=3

计算得到 8 个方案的收敛率分别为:

- AdamsBashforth p=4: 4.7322
- ADM p=5: 4.7399
- BDM p=4: 4.9354
- classicalRK: 3.6432

- Dormand-Prince p=5: 5.0006
- ESDIRK: 0.8044
- FehlbergRK p=5: 4.7885
- GaussLegendreRK p=3: 4.6520

发现 ESDIRK 的收敛阶似乎于理论不符，查询数据发现由于为了避免迭代死循环，步长减小了 10 倍，而此时的误差已达到 10^{-6} 接近设置的 $eps = 10^{-9}$，小于其他方法的误差，所以可能是浮点误差导致的问题。

3.2 第二部分

测试 Euler 方法 24000 步，经典 RK 方法 6000 步，Dormand-Prince 方法使用自动控制步长 100 步，图像如下：

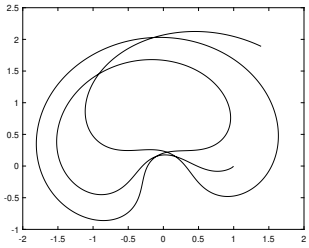

图 33: Euler

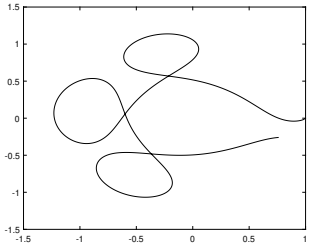

图 34: ClassicalRK

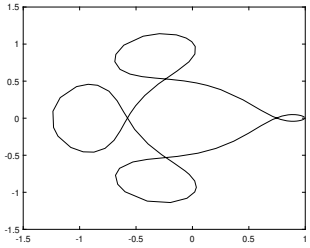

图 35: Dormand-Prince

经测试发现，步长从 k = 0.004 开始减小，Euler 方法最快，直至 k=0.0005 都不能达到精度要求；经典 RK 方法在 k = 0.0005 时达到精度要求，用时为 0.484375s；Dormand-Prince 方法在 k = 0.001 时达到精度要求，用时为 0.765625s。

所以如果仅考虑步长和精度的关系，经典 RK 方法胜利；如果仅考虑时间和精度的关系，Dormand-Prince 方法胜利，

具体数据可见 data 文件夹下的 AdamsBashforth1test1,classicalRK4test1,DormandPrinceRK5test1.