Project: Numerical Solvers for System of Ordinary Differential Equations

Instructor: Seongjai Kim*

1. Introduction

In the project, we will study some numerical techniques for the numerical solution of systems of nonlinear ordinary differential equations (ODEs). In particular, we will consider **one-step methods** such as Euler method, Heun's method, Runge-Kutta method. Through the project, you will implement the algorithms for a given ODE system and compare the numerical solutions for different methods as well as for various step sizes. Even though MATLAB is a convenient computational language, we will try to utilize computational languages such as C++, C, and FORTRAN. In fact, we will experience the three languages in a program; the purpose of such a multi-language implementation is for you to be better prepared for both the computational languages and very heterogeneous simulation environments in industry. I know some of you may feel uncomfortable in the beginning; however, you will be okay soon. Do not hesitate for questions and suggestions.

2. The model and numerical methods

The first-order initial value problem (IVP) is

$$\begin{aligned} \frac{dy_i}{dx} &= f_i(x, y_1, y_2, \cdots, y_d), \\ y_i(x_0) &= y_{i0}, \end{aligned} \qquad i = 1, 2, \cdots, d, \tag{2.1}$$

for a prescribed initial values $\{y_{i0} : i = 1, 2, \cdots, d\}$. We assume that (2.1) admits a unique solution in a neighborhood of x_0. In fact, it is known for $d = 1$ that if f_1 and $\partial f_1/\partial y$ are continuous in a box $(a, b) \times (c, d)$ containing (x_0, y_0), then (2.1) has a unique solution in an interval I such that $I \subset (a, b)$ and $x_0 \in I$.

*Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506-0027 U.S.A. Email: skim@ms.uky.edu; Phone: (859) 257-8681; Office: 827 Patterson Office Tower.

In the following, we will present one-step methods for the case $d = 1$:

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(0) &= y_0. \end{aligned} \tag{2.2}$$

Let $x \in [0, T]$ and $h = T/N$, for a positive real T and an integer N. Let $x_n = nh$, $n = 0, 1, 2, \cdots, N$, and y_n be the approximate solution of y at x_n.

- **Euler Method (Error=$\mathcal{O}(h)$):**

$$y_{n+1} = y_n + hf(x_n, y_n).$$

- **Improved Euler (Heun's) Method (Error=$\mathcal{O}(h^2)$):**

$$\begin{aligned} K_1 &= h\, f(x_n, y_n) \\ K_2 &= h\, f(x_n + h, y_n + K_1) \\ y_{n+1} &= y_n + \frac{1}{2}(K_1 + K_2) \end{aligned}$$

- **Runge-Kutta Method (Error=$\mathcal{O}(h^4)$):**

$$\begin{aligned} K_1 &= h\, f(x_n, y_n) \\ K_2 &= h\, f(x_n + \frac{h}{2}, y_n + \frac{1}{2}K_1) \\ K_3 &= h\, f(x_n + \frac{h}{2}, y_n + \frac{1}{2}K_2) \\ K_4 &= h\, f(x_n + h, y_n + K_3) \\ y_{n+1} &= y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned}$$

Remark. The one-step methods can be utilized for the whole computation or for several starting steps of multi-step methods. So some people call them the *starting methods*. Vectorization of the methods for (2.1) is an easy task. Just use the same algorithm component by component!

3. The objects

For the project, I will provide routines including

- main program (in C++)
- I/O driver (in C)
- cstart and mkmk: generators of main/IO routines and Makefile
- Euler, Heun, and Runge-Kutta algorithms for (2.2) (in F77)

Your job is to

1. **Download the code, this document, and cstart+mkmk from** `www.ms.uky.edu/~skim/GRADE` **or** `www.modelcode.net`**:**
    - `ODESOL.CF.tar`
    - `ODESOL.ps`
    - `cstart+mkmk.v1.tar`
2. **Adjust the code for**

$$
\begin{aligned}
{y_1}' &= (x^2+2)(x+1)-(x+1)y_1-y_2-y_3\\
{y_2}' &= 3(y_1-1+e^{-x})^2\\
{y_3}' &= x+1-y_1-y_3
\end{aligned}
\tag{3.1}
$$

$$y_1(0)=y_2(0)=y_3(0)=0, \quad x\in[0,2].$$

3. **Compare the solutions for various step sizes and the three different methods.**
4. **We can get the analytic solution for (3.1). However, let's assume that we do not know the analytic solution. Under this assumption, how can you check if the numerical solution has the desired order of accuracy? This question is practically the main of the project.**

The algorithm is written using C++, C, and F77. Researchers in scientific computation know a language, but most of them don't know (or are not willing to know) another language. FORTRAN shows advantages in scientific computation. In particular, if you are dealing with complex-valued solutions, FORTRAN is the language. Also we know that C/C++ can easily incorporate other libraries, packages, and GUIs; they are so powerful! I hope you can get an idea for modern high-performance computation environments, after or during the project. I believe such an experience of multi-language computation will help your future research and job search.

Makefile: Multi-Language computation is not easy for some cases. However, I believe there is always a way to link objects constructed from different compilers. I assume the computer you are using is installed with C++, C, and F77. (You can find such a computer in campus. If it is not true for you, please contact me as soon as possible.) Once the compiling is finished, the next problem is linking. The makefile generator will produce `Makefile` to link all. If the linking is not successful, email me the result of the Unix command

```
echo $HOST $OSTYPE
```