Assignment 3

ISC 4232/5935

Due: October 21 at 4:00pm

Programming Language: any sensible language
Submit write-up, code, and any auxiliary files to bquaife@fsu.edu
Undergraduates: Submit questions 1 and 2
Graduates: Submit all questions

---

1. The goal of this question is to write a collection of explicit Runge-Kutta routines that can be used as a stand-alone library that we will call from a driver code. We will be using the four Runge-Kutta methods given by the Butcher tableaus

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 2/3 & 2/3 & 0 \\ \hline & 1/4 & 3/4 \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

All these methods can be found at *https://en.wikipedia.org/wiki/List_of_Runge-Kutta_methods*. Develop your library in the following manner.

   (a) Write 4 functions that take no input (give them each a unique name) and returns the appropriate `s` $\times$ `s` matrix `A` and the `s` $\times$ 1 vectors `c` and `b`.

   (b) Write a function that takes in as the inputs the initial condition `y0`, the time step size `dt`, a time horizon `T`, the number of stages `s`, a right-hand side of the IVP `f`, and, if available, a true solution `ytrue`. The input `f` should be a function that takes in a current state `y` and time `t` and returns the number `f(t,y)`. The input `ytrue` should be a function that takes in a set of time steps `t` and returns the vector `ytrue(t)`. The output of this function should be a collection of time steps `t` in `[0,T]`, the numerical solution `y` at these time steps, and the exact solution `yexact` at these time steps. If `ytrue` is not passed into the function, `yexact` should be an empty array.

   (c) In the first few lines of this code, set the number of time steps to `ceil(T/dt)`, allocate space for the time steps `t` and the numerical solution `y`, and allocate a vector of size `s` to store the stages $(k_1, k_2, \ldots, k_s)$. Then load the Butcher tableau from part (1a) corresponding to the input variable `s`.

   (d) The outermost loop should be a loop over the time variable. Within this outer loop, the `s` stages should be formed using an inner loop. Vectorize your code by using a dot product to compute the required term in the j^{th} stage. That is,

$$a_{j1}k_1 + a_{j2}k_2 + \cdots a_{j(j-1)}k_{j-1} = (a_{j1}, a_{j2}, \ldots, a_{j(j-1)}) \cdot (k_1, k_2, \ldots, k_{j-1}).$$

   Recall that a dot product is simply a row vector times a column vector. Therefore, there should be a loop over the index of the stage, but each stage should be computed without a loop.

(e) After the stages $k_1, k_2, \ldots, k_s$ are formed, compute the solution at the new time step Y_{n+1} using a dot product between the stages and the vector `b`.

(f) Store the current time in the vector of times `t` and the numerical solution in the vector of approximate solutions `y`.

(g) Outside of the time loop, if a function `ytrue` was passed as an input, evaluate the exact solution at the time steps `t` and store it in `yexact`. Otherwise, set `yexact` to be the empty array.

There should only be two loops nested within each other. The outer loop is used to move forward in time, and the inner loop is used to compute the stages.

2. Write a driver code that calls your explicit Runge-Kutta library. Use it to solve the initial value problems

$$\begin{aligned} y' &= 1 - y^2, \quad 0 < t \leq 2 \\ y(0) &= 0, \end{aligned}$$

which has the exact solution $y(t) = (e^{2t} - 1)/(e^{2t} + 1)$, and

$$\begin{aligned} y' &= -4t^3 y, \quad 0 < t \leq 1 \\ y(0) &= 1, \end{aligned}$$

which has the exact solution $y(t) = e^{-t^4}$.

For both problems, perform a convergence study for all four time Runge-Kutta methods. Plot the errors at the time horizon versus the time step size on a `loglog` scale and plot reference lines of slope 1, 2, 3, and 4. For these methods, the order of accuracy is equal to the number of stages. There should only be two plots—one for each IVP. Save and include these plots in your assignment (I shouldn't have to run your code for the plots to be formed).

3. (**Grad Students Only**) Use your explicit Runge-Kutta library to numerically compute and plot the stability regions of each of the four Runge-Kutta methods. Recall that this requires solving $y' = \lambda y$ which results in a amplification function Λ that satisifes $Y_{n+1} = \Lambda(\lambda \Delta t) Y_n$. Since λ and Δt always show us the product $\lambda \Delta t$, you can assume that $\Delta t = 1$ and vary λ over a meshgrid in the complex plane. Once the amplification factor is computed on this meshgrid, find all values with absolute value less than 1, and plot dots at these points. In Matlab, it would look something like

```
lambda = ...  % form mesh grid of complex numbers
Lambda = ...  % form amplification function

stab_reg = find(abs(Lambda) < 1)
% find points where Lambda is less than 1
plot(real(lambda(stab_reg)),imag(lambda(stab_reg)),'k.')
% plot stability region
```

4. (**Grad Students Only**) Diagonally implicit Runge-Kutta (DIRK) methods are implicit Runge-Kutta methods where the stage k_j depends on $k_1, k_2, \ldots, k_j$, but not on $k_{j+1}, k_{j+1}, \ldots, k_s$. Write another library that can used to apply the DIRK methods:

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} \frac{1}{2}+\frac{1}{2\sqrt{3}} & \frac{1}{2}+\frac{1}{2\sqrt{3}} & 0 \\ \frac{1}{2}-\frac{1}{2\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{2}+\frac{1}{2\sqrt{3}} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|ccc} \frac{1+\alpha}{2} & \frac{1+\alpha}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{\alpha}{2} & \frac{1+\alpha}{2} & 0 \\ \frac{1-\alpha}{2} & 1+\alpha & -(1+2\alpha) & \frac{1+\alpha}{2} \\ \hline & \frac{1}{6\alpha^2} & 1-\frac{1}{3\alpha^2} & \frac{1}{6\alpha^2} \end{array}$$

where

$$\alpha = \frac{2}{\sqrt{3}} \cos\left(\frac{\pi}{18}\right).$$

Design your library to solve the IVP

$$y' = f(y,t) = f_1(t)y(t) + f_2(t).$$

The functions $f_1(t)$ and $f_2(t)$, rather than $f(t,y)$, will be passed to your DIRK library.

Use your library to:

(a) Numerically compute the order of accuracy for the IVP

$$\begin{aligned} y'(t) &= 2ty(t) + t, \quad 0 < t \leq 1 \\ y(0) &= 0, \end{aligned}$$

which has the exact solution $y(t) = \frac{1}{2}(e^{t^2} - 1)$. Compare the relationship between the number of stages and the order with the same relationship for the explicit Runge-Kutta methods from Question 1.

(b) Numerically compute the stability regions of the DIRK methods. Be aware that there are small "islands" of stability that are disconnected from the main region of stability. Do any of the DIRK methods appear to be A-stable?