# 1 Numerical Solutions to ODEs–More Runge-Kutta

In our exploration of numerical solutions to ODEs, we've yet to attempt to assess how good the approximate solution is. We begin to do so now.

(1) Translate the adjustable step size algorithm just introduced in the lecture to `MatLab` code.

(2) Using an adaptive step method that doubles (or halves) the initial step size works, but it is rigid in that only doubling (or halving) of the steps is allowed. A modification of Runge-Kutta due to Fehlberg uses the relative error to determine the amount the step size should change. The details of how this is done have been presented in the lecture. Without worrying about the specific forms of the functions $f_i$ or the constants $b_i, c_i$, write a step by step procedure in the table below for solving ODEs using the the Fehlberg adaptive method just discussed. In the right column, write snippets of code that might be used to carry out each step.

| Step | Code |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

(3) Download the file `MyRK4_Adapt_Fragment.m` from the Teams page. The file as it stands is not complete and will not work. However, the section that determines whether or not the current step size is adequate is complete. *Map* the code you started to develop in question (2) on to this code.

(4) Make the code usable and apply it to the ODE

$$\frac{dy}{dt} = \exp(t)\sin(y).$$

Use a starting condition of $t_o = 0, y_o = 3$ and an initial step size of $h = 1$. Have the code plot the solution and print $h$ to the screen whenever it is changed.

(5) At this point you have a functioning ODE solver that is adaptive, stable, and can handle many types of ODEs encountered in physics. There are some changes that will make the code more usable. These changes include

- Making the initial conditions inputs
- Making the output of the function arrays
- Making epsilon a 'global' variable
- Separating the algorithm from any specific function
- Stopping the program when a problem is encountered by checking if $h_{new}$ is too large or too small. This indicates that the nature of the solution is not being captured by the algorithm.

Download the file `modifications.m` from Team which contains some of these changes. Modify your existing code using this file, to implement these changes.

---

Many (or most?) problems in physics involve multiple dependent variables. For example Newton's second law,

$$\frac{d\boldsymbol{p}}{dt} = m\frac{d\boldsymbol{v}}{dt}$$

is a actually three equations. Thus one is no longer dealing with an ordinary differential equation, but with a *system* of ordinary differential equations. In general, each equation is coupled to the other meaning that the dependent variables appear in all the equations.

We can write the dependent variables as a *vector* such as

$$\vec{\boldsymbol{S}} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

where $x$ and $y$ are the dependent variables and become the components of the vector, $\vec{\boldsymbol{S}}$. Similarly the differential function is,

$$\vec{\boldsymbol{F}} = \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix} \tag{2}$$

where $f(x,y)$ and $g(x,y)$ are the differential functions. The *system* can now be written as

$$\frac{d\vec{\boldsymbol{S}}}{dt} = \vec{\boldsymbol{F}}(x,y). \tag{3}$$

It is important to keep in mind that Eq.(3) is really a system of equations. That is, Eq.(3) is really,

$$\begin{aligned} \frac{dx}{dt} &= f(x,y) \\ \frac{dy}{dt} &= g(x,y) \end{aligned}$$

While we have shown only two dependent variables, this can obviously be extended to any number of variables.

One advantage of expressing a system of ODEs as done in Eq. (3) is that the system now 'looks' like a single equation and we can deal with it as we did a single ODE so long as we remember that the variable $\vec{\boldsymbol{S}}$ is a vector and so has components. More importantly for us, is that the way to solve systems of ODEs numerically becomes a relatively minor modification to the algorithm we've already developed.

(6) Enumerate the changes to your exisiting code that you need to make in order to handle *systems* of ODEs.

(7) Implement these changes to your code