P346 Project Report

Adaptive Step Size Runge-Kutta Method

P K Samay

2011102

(Dated: November 17, 2022)

The classic Runge-Kutta method (RK4) is a very useful tool in finding numerical solutions to ordinary differential equations accurately, while having a reasonable computational expense. This method was proposed by Carl Runge and Wilhelm Kutta. However, the method's efficiency is heavily dependent upon the step size it uses. In this project, a version of the RK4 method that can appropriately change its step size, given a specific tolerance value. Using the adaptive step size runge-kutta method, I have attempted the predator prey problem and visualized different scenarios.

I. INTRODUCTION

The RK4 method (multiple-step 4th order RungeKutta) is found to have no flexibility while it performs computations. The step size, *h*, once chosen, remains constant during the execution of code for a given problem. There are issues in obtaining accurate solutions when the step size is either too large, or too small.

We know from our coursework in P346 that choosing smaller step sizes and higher order methods are two possible routes to reducing the size of the error we get in our solution. However, this is not always the case (say, in an N-body problem). This issue can sometimes be solved by being able to dynamically change the step size for our numerical differential equation solver, when given a fixed amount of accuracy for our solution. We will now discuss such a method that allows us to do the same, which is known as the Runge-Kutta-Fehlberg method.

II. WORKING PRINCIPLE AND DERIVATION

The method that is described in Fehlberg's 1969 paper is novel due to the reason that it is an 'embedded' method from the RK family of methods. In embedded methods, identical function evaluations are used together to create methods of multiple orders and similar error constants. Fehlberg's algorithm (also known as the RKF45 method) is a 4th order method, but it has a 5th order error estimator. With the performance of an extra calculation, the solution's error can be estimated and regulated accordingly by using the higher-order embedded method that allows for a step size to be determined dynamically.

In general, $(p-1)^{th}$ order adaptive methods require a p^{th} order step in order to compute the local truncation error for a step in question. Since these two steps occur in a single, interwoven step for a loop, a lower computational cost is entailed that what would've been entailed for a method of a higher order.

During the execution of the integration, the step size is changed in such a way so as to keep the estimated error below a pre-determined threshold. If the error is higher than permissible, the step is repeated with a lower step size. if the error is small, the step size is increased to reduce the computational cost. This results in the maintenance of an optimal step size, which saves time spent in computation. Hence, additional effort is not needed to find an appropriate step size for a given problem. The higher order step is computed as,

$$y_{n+1} = y_n + h\Sigma_{i=1}^{s} b_i k_i \tag{1}$$

We also know that the lower order step is computed as,

$$y'_{n+1} = y_n + h\Sigma_{i=1}^{s} b'_i k_i \tag{2}$$

Here, k_i are the same as for the higher order method. Hence, we get the error between the two for the (n+1)th step as,

$$e_{n+1} = y_{n+1} - y'_{n+1} = h\Sigma_{i=1}^{s}(b'_i - b_i)k_i \tag{3}$$

And here, *e* is the higher order estimator. For this family of adaptive methods, it is seen that the general Butcher tableau is given as,

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\cdot & \cdot & \cdot & & & \\
\cdot & \cdot & & \cdot & & \\
\cdot & \cdot & & & \cdot & \\
c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s \\
 & b'_1 & b'_2 & \dots & b'_{s-1} & b'_s
\end{array}
$$

Hence, we can write out the Butcher tableau for RKF45 as,

$$
\begin{array}{c|ccccccc}
0 & 0 & & & & & & \\
\frac{1}{5} & \frac{1}{5} & 0 & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & 0 & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{-212}{729} & 0 & & \\
1 & -\frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & 0 & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

These coefficients are applied in the code to find the increments and the summations of the weights for the increments, which is then used to compute the appropriate adjustment for the step size.

The truncation error (E), which is the 5th order estimator, is calculated using the coefficients obtained in the embedded pair (the lower two rows, or weights) as,

$$E = |\Sigma_{i=1}^{6}(b_i^{'} - b_i)k_i| \tag{4}$$

From the truncation error, we compute the new step size as,

$$h_{new} = kh\left(\frac{\epsilon}{E}\right)^{\frac{1}{4}} \tag{5}$$

Here, the k factor can be between 0.8 and 1.0, depending upon the amount of fluidity required in changing the step size. If $E > \epsilon$, then we replace h with h_{new} and repeat the step. But if $E \leqslant \epsilon$, then the step is completed successfully and we replace h with h_{new} for the following step.

III. PROBLEM

Now, we will have a look at the efficiency and that our adaptive step algorithm is able to provide the scenario of Predator-Prey system. We invoke the case of a predator-prey model where the rate of population change for two species (let's say, X and Y) with a prey-predator relationship is dictated by the following two differential equations,

$$\frac{dx}{dt} = ax - \alpha xy \tag{6}$$

$$\frac{dy}{dt} = -cy + \gamma xy \tag{7}$$

Here, x is the population of species X (the prey species), and y is the population of species Y (the predator species). a is known as the prey birth rate and c is known as the predator death rate, respectively. While α and γ can be called as the 'interaction coefficients' for the two species.

From observation, it can be seen that the two differential equations cannot be separated from each other, i.e., they are coupled differential equations. As per the literature, this system of equations behaves in an oscillatory manner, but without a fixed period. We qualitatively examine the behaviour of this period in our analysis.

IV. RESULTS

The birth rate (a) and death rate (c) various cases. It was seen that the maximum populations of the prey and predator species increased dramatically with each period, as opposed to a much slower increase. It is difficult to tell if this is a more accurate or a less accurate solution that the one provided in Juarlin's paper, since no analytical solution exists for this system of equations. However, we can clearly see how the step size changes multiple times during a single period to accommodate for both the prey and predator solution curves. In the 1st case seen below, t was initialised at t = 0 and terminated at t = 500, taking an execution time of about a second.

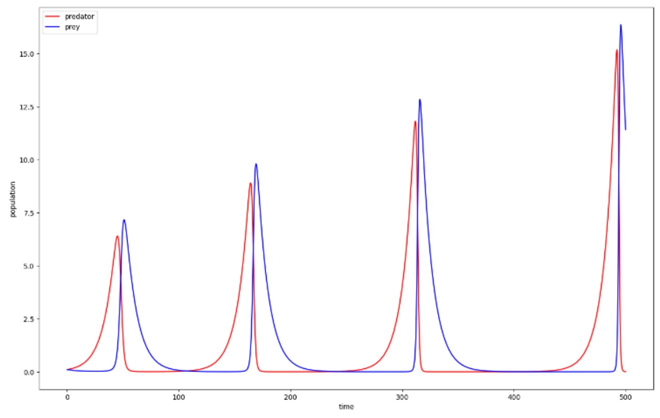

Figure 1 : a = 0.1 and c = 0.1

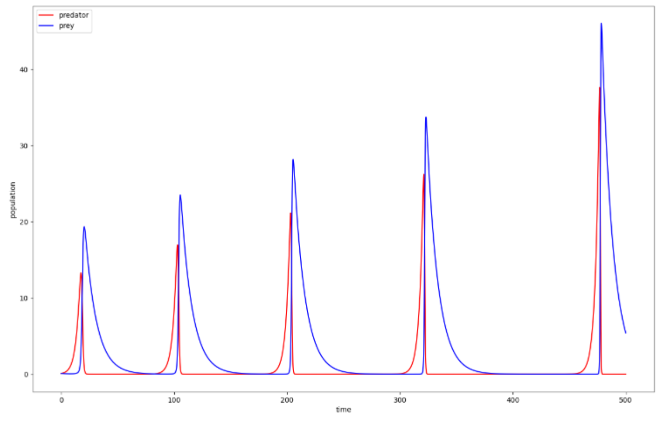

Figure 2 : a = 0.3 and c = 0.1

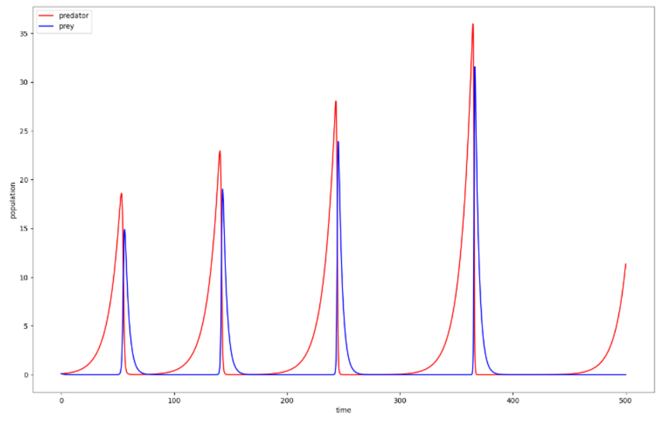

Figure 3 : a = 0.1 and c = 0.3

In the 1st case we set the birth rate of prey and the death rate of predators the same (a = c = 0.1). And we observe a similar rate change in population over time but in the contrast manner. We observe that the scarcity of any of them makes the other one rare with a time lag and just in the next time period the 1st one repopulates which is followed by the other species with the similar time lag.

Next (Figure 2), we set the prey to reproduce rapidly (a = 0.3). We can see that this results in a much smaller period for the oscillation of the population curves, as the prey species is produced much quicker, the predator is also eliminated quicker. But to be noted that the population of both prey and predator reached more value than the 1st case.

Finally (Figure 3), we look at a system where the predators die rapidly ($c = 0.3$) while the prey species are born at the same rate as the first case ($a = 0.1$). We see that a large amount of prey animals is needed before the predator population rises significantly. Here we can see that although the death rate in predators is high still the population amplitude was larger than that of the 1st case but smaller than the 2nd one.

But I must admit that I'm not convinced with the plots. As a broad picture it seems correct. But if we look closely we see that in the regions between the population peaks when there is no predator why the population of the prey dropped and the other way too when the prey were not available how does the predator were surviving. Maybe natural limitation describe it.

It is ultimately difficult to comment on the accuracy of these results, as not much literature exists on this particular application of the RKF45 algorithm.

After observing all the application, we hence come to the conclusion that the RungeKutta-Fehlberg modification to the original RK4 algorithm offers some valuable computational discounts in some cases, while resulting in only a marginally higher processing time per step. It is a wise idea to compare the performance for such numerical algorithms before using them on much larger data sets, to check the suitability for a given algorithm to a particular problem.

V. REFERENCES

[1] NISER P346 Lecture Notes: Numerical Integration Slides: https://classroom.google.com/u/0/c/ NTM3OTY3MDcwMjQz

[2] E. Fehlberg, "Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems", NASA Technical Reports, TR R-315(1969).

[3] E. Juarlin, "Solution of Simple Prey-Predator Model by Runge-Kutta Method", J. Phys.: Conf. Ser **1341** 062024(2019).