Phys 410 Homwork 2

Jack Parkinson

November 4, 2022

Introduction

Solutions to systems differential equations is an important problem in computational physics. There are many equations that cannot be evaluated analytically thus requiring that computational methods exist to solve these problems. There are multiple methods to solve systems of ordinary differential equations, but one of the most common is the 'Classic Runge-Kutta method'. This is the method that will be explained and implemented in this paper to solve ordinary differential system of equations.

Mathematical Background

There are Three main aspects to the Runge-Kutta Method:

1) Defining System of Differential Equations:
In order to use the Runge-Kutta method first we have to define the initival value problem:

$$\frac{dy}{dt} = f(t, y);$$

2) Defining Initial Conditions:
In order to be able to use the Classic Runge-Kutta method we need to define intial conditions of the form

$$t(0) = t_0; \quad y(t_0) = y0$$

We also want to specify the difference in time from t_0 that we wish approximate the solution, or the Δt from

$$y(t_0 + \Delta t)$$

.
3) Applying the Runge-Kutta Method:
For this report we are just interested in the fourth-order classic Runge-Kutta method which has form:

$$y(t_0 + \Delta t) = y_0 + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (1)$$

where

$$k_1 = f(t0, y_0),$$
$$k_2 = f(t_0 + \frac{\Delta t}{2}, y_n + \Delta t\frac{k_1}{2}),$$
$$k_3 = f(t_0 + \frac{\Delta t}{2}, y_n + h\frac{k_2}{2}),$$
$$k_4 = f(t_0 + \Delta t, y_n + hk_3),$$

Now we can implement the classic Runge-Kutta method for different system of Equations as we will see in the solutions to three problems:

Problem 1 - Single Fourth Order Runge-Kutta step

In this problem we wished to take a single fourth order Runge-Kutta step. The implementation uses inputs: fcn, t0, dt, y0 whre fcn is the system of ODEs and the other paramters are as we defined in the mathematical background ($t0 = t_0; y0 = y_0; dt = \Delta t$). From here we can use the exact implementation of (1).

To test the accuracy of our implementation we can consider the case of a simple harmonic oscillator:

$$\frac{dx^2}{dt^2} = -x(t)$$

which we can decompose into the system of equations:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -x_1$$

where $x(0) = 0$, $\frac{dx}{dt}(0) = 1, \Delta t = 0.1$
And if we use Runge Kutta then $x_1(t)$ estimates $x(t)$ and $x_2(t)$ estimates $\frac{dx}{dt}$
The solution to this system which we is:

$$x(t) = sin(t)$$

This yields a computed fourth Order Runge-Kutta step of

rk4step$(0 + 0.1) = 0.0998$

Which yields a local per step error $x(0 + 0.1) - \sin(0.1) = 8.33 \times 10^{-8}$

which we can see is on the order $O(t^5)$: $\quad 8.33 \times 10^{-8} < 10^{-5}$

(To reproduce these results run trk4step)

Because we have verified the local per step error is $O(t^5)$ we can assume our Runge-Kutta fourth order step is working as expected and we can proceed on:

Problem 2 - Complete fourth Order Runge-Kutta

Now that we have a function to compute a single runge-kutta fourth order step, we can loop this over a range of time steps to find the solution of a system of ODEs over time.

This function will take in three parameters (fcn - a function for right hand side of ODES, tspan- vector of output times, and y0- vector of initial values)

We can try this method on the simple harmonic oscillator with the same initial conditions as in problem 1 except with a vector of times from 0 to 3π uniformly spaced out in nt points

where

$$nt = 2^l + 1$$

where l is defined to be the 'levels' corresponding to many estimate solution values we want between 0 and 3π

In order to explore the relationship between levels and the errors of the solution we can graph the difference between the solution estimates relative to the lowest error: $l = 5$:

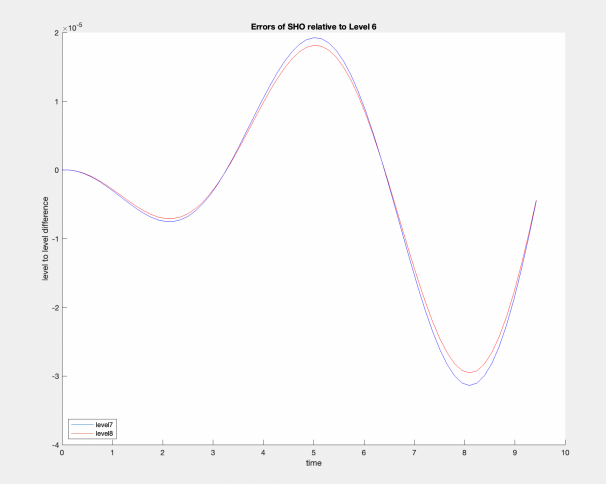


Figure 1: Error comparison relative to Level 5

As we can see in the above figure there is a clearly defined relation in the error between levels. From this relation we can see that the error starts out relatively

small then grows as time increases. A takeaway from this is that if we want to use lower levels we should restrict ourselvs to values closer to the initial conditions. This makes sense because all 3 sets of levels have an accurate t_0 but time increases the differnece from the solution becomes more apparent.

(to reproduce these results run trk4_sho)

Besides the Simple Harmonic Oscillator another System of ODE that we can solve is the Van der Pol Oscillator which has form:

$$\frac{d^2x}{dt^2} + a(x^2 - 1)\frac{dx}{dt} + x = 0$$

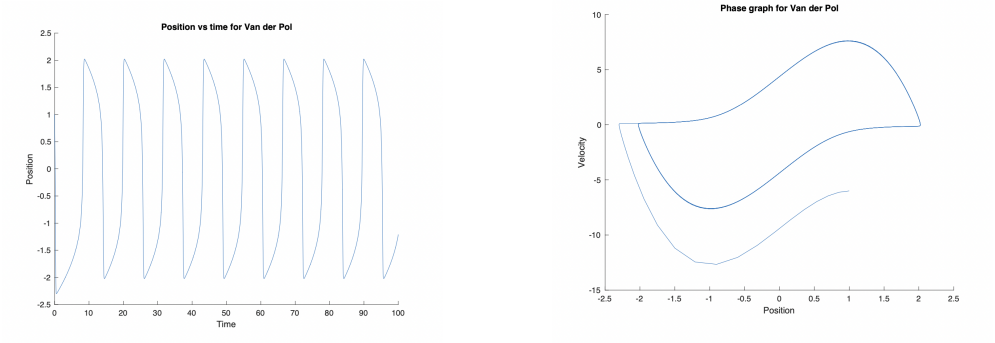
Which we can decompose into form:

$$\frac{dx_1}{dt} = x_2; \quad \frac{dx_2}{dt} = \alpha(1 - x_1^2)x_2 - x_1$$

Implementing this into the Complete fourth-order Runge-Kutta with the initial Conditions:

$$\alpha = 5; x(0) = 1; \frac{dx}{dt}(0) = -6$$

And with an evenly spaced time grid from 0 to 100 with step size $\frac{100}{2^{12}+1}$
We get:



(a) Van der Pol: Position (b) Van der Pol: Phase

Figure 2: Position and Phase graphs of Van Der Pol

These results show the graphical solution to the unforced Van Der Pol system of ODE. We can tell from these that the Position of the Van Der Pol graph is periodic in time and the graph also illustrates the phase over time.

(to reproduce run trk4_vdp)

Problem 3 - Adaptive Runge-Kutta

Now we wish to implement an adaptive Classic Runge Kutta that changes the step size depending on the given error between steps:

$$y_C(t_0 + \Delta t) - y_F(t_0 + \Delta t) = \frac{15}{16} e_C$$

where y_C denotes a 'Coarse' Runge-Kutta step, or the same Runge-Kutta step as in Problem 1, and y_F denotes a 'Fine Runge-Kutta Step which is the same as the second of two coarse steps with $\Delta t_F = \Delta t_C/2$. Also, e_C is the Local error, so we want to relativize it with y_F. Putting that all together we get:

$$e = \frac{16}{15 y_F}(y_C - y_F)$$

we then want to compare this relative error with an input parameter: rel tol, and if the error is greater than the relative tolerance we halve dt. And if $\frac{dt}{2} < 10^{-4}$ we do not decrease dt anymore. Finally, if we have reached the next element in the time array we store the final y_C value we got and reset dt.

Implementing this with the same Initial Conditions for the Simple Harmonic Oscillator seen earlier for 4 different relative tolerances, and graphing the error with the solution of the Simple Harmonic Oscillator ($x(t) = sin(t)$) we get the following error graph:

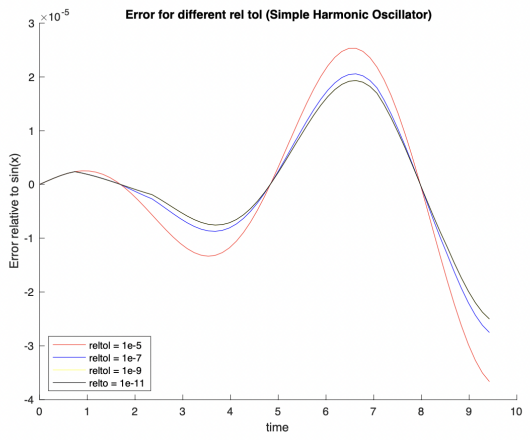
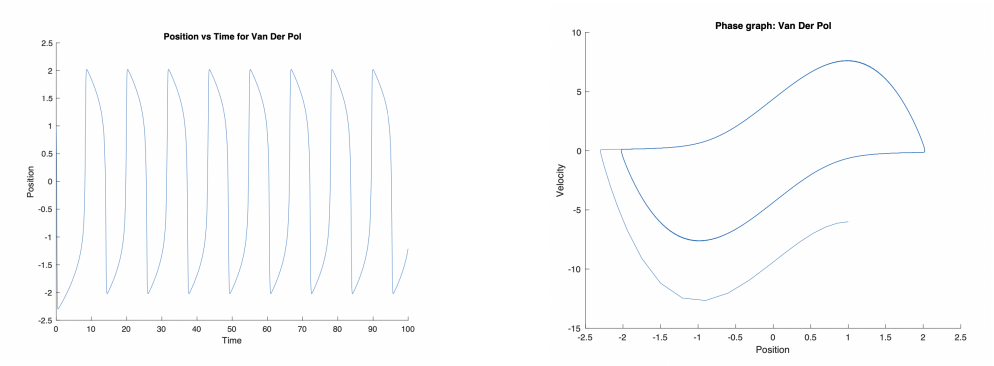


Figure 3: Error comparison relative to SHO solution

(to reproduce run trk4ad_sho)
From this plot of the errors relative to the solution of the Simple Harmonic Oscillator we see that the smaller the relative tolerance the closer relative errors are to 0. This makes sense, because we expect that for higher tolerances like 10^{-11} dt is being continuously halved until the error is smaller than either the

relative or the floor which results in smaller dt in the Runge-Kutta steps. And if the Runge-Kutta step sizes are very small, then the approximation method can converge to the solution more accurately.

And Implementing the same initial conditions for the Van Der Pol system we get the following position and phase graphs:



(a) Van der Pol: Position (b) Van der Pol: Phase

Figure 4: Adaptive Position and Phase graphs of Van Der Pol

These graphs appears very similar if not slightly smoother than in the non-adaptive case, but without the analytical solution or actual data we cannot say for certain that the adaptive method is more accurate than the non-adaptive for the Van Der Pol system.

(to reproduce run trk4ad_vdp)

Further Analysis And Conclusion

We were able to implement a non-adaptive and adaptive Runge-Kutta method for solving Systems of ordinary differential equations. By first creating a function to take a single fourth-order runge-kutta step we verified it by checking the error was $O(\Delta t^5)$ for a simple harmonic oscillator.
We then implemented this in the non-adaptive Runge-Kutta method that solved systems for both Simple Harmonic Oscillator and an unforced Van Der Pol Oscillator. By graphing the errors of the Simple Harmonic Oscillator we saw that errors were smallest for small values of time. This results implied that the Runge Kutta method will have smaller errors for t values closer to the initial t_0. We can generalize these results by making the suggestion that the Runge-Kutta non adaptive method should be only used for independent variables not too far from the interval values (for lower levels). We also graphed the position and phase graphs of the Van Der Pol unforced oscillator. These results show a graphical solution that provides us with information on the solution to the Van Der Pol system.

We also created an adaptive Runge Kutta method that changed the step size based on the calculated error between the course and fine Runge-Kutta methods. By setting the relative tolerance, we can control how strongly the Runge Kutta method is adapted. We explored the relation between relative tolerance and the error of the solution in Figure 3 that showed: as we increase the relative tolerance the runge-kutta estimate converges closer to the solution. However, it did seem like the two lowest relative tolerances: $10^{-9}, 10^{-11}$ had near identical error plots suggesting that the difference between the estimate and solution doesn't converge completely. However, this could be due to the adaptive runge kutta function reaching the floor in those iterations nearly every time. More research is needed to find out if very small relative tolerances converge to the solution completely. Finally we graphed the adaptive solutions to the Van der Pol system. Comparing these results to the non-adaaptive case in figure 2, it appears the two are very similar if not slightly smoother for the adaptive case. This suggests there is little to improvement in implementing the adaptive runge-kutta method to the Van Der Pol system compared to the non-adaptive, but more studies should be taken with both smaller relative tolerance values and different a values.

References

(1) Phys 410 Homework 2 Description:
http://laplace.physics.ubc.ca/410/homework/hw2.pdf

(2) Phys 410 Tutorial 4 Description:
http://laplace.physics.ubc.ca/410/tutorials/tutorial-4.html