PHYS 410
Melvin Mathews
57181142

Assignment 2 Report

Part 1: rk4step

In this section, a single step of the fourth order Runge-Kutta step was to be implemented.

Methodology

The Runge-Kutta method involves using slopes calculated at multiple points to approximate solutions to ordinary differential equations. The fourth order uses 4 slopes calculated at 4 points within a step. The first of which is calculated at the start of an interval, two which are calculated at the middle of the interval, and the final one calculated at the end of an interval. An average of these slopes is then taken and used to increment the current step. Using k_i to denote each slope and h to denote the width of the step, this can be summarized by the equation below:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

The fourth order Runge-Kutta is known to have a local error on the order of the timestep to the power of 5. Which is to say that the quotient between the absolute errors of two Runge-Kutta steps with differing step sizes would simply be the quotient of the step sizes to the power of 5.

File Descriptions

- rk4step.m: This file contains the function which given a differential equation (in canonical form) function handle, initial condition and step size, will return the value y_{n+1} from the equation above.
- fcn.m: File containing the differential equation describing simple harmonic motion with unit angular frequency and cast in canonical form. This is used to test rk4step as well as the methods discussed in part 2 and 3 of this report. The equation(s) can be explicitly seen in the equations below:

$$\frac{d^2y(t)}{dt^2} = -y$$
$$\frac{dy_1}{dt} = y_2, \frac{dy_2}{dt} = -y_1$$

  It should be mentioned that while this is not explicitly stated in this file, this function was always tested with the corresponding initial conditions:

$$y_1(0) = y(0) = 0, y_2(0) = \frac{dy}{dt}(0) = 1$$

- rk4step_test.m: This file was used to test the rk4step function with varying step sizes and the function and initial conditions described above. The step sizes ranged from 0.1 to 0.0125 seeing a factor 2 reduction for a total of 4 step sizes.

Results

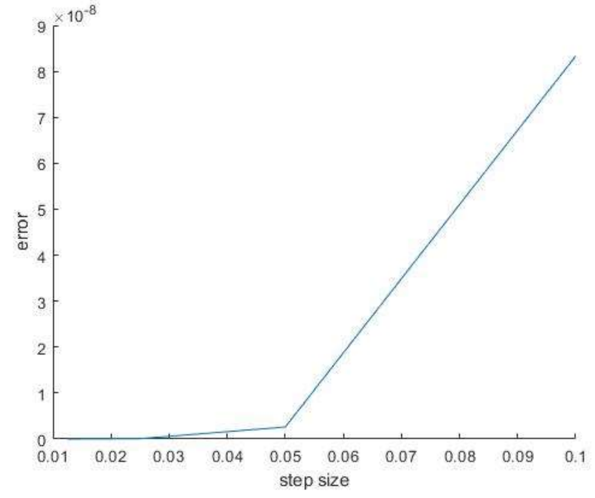
The procedure described in the last sentence of the methodology section was carried out for the tested step sizes to see the expected result of approximately 32. An explanation for why 32 is expected:

$$\frac{{\Delta t_2}^5}{{\Delta t_1}^5} = \frac{0.05^5}{0.1^5} = 2^5$$

To be precise, the function describing simple harmonic motion was used along with various step sizes as input into rk4step. The output y values were then compared with the analytical solution to simple harmonic motion to generate the absolute error values.

It is observed that when comparing errors of 2 correspondingly smaller step sizes, this quotient is also closer to 32. This provides not only assurance that the method works for a single pair of step sizes but also that there is convergence behaviour at play. A rather obvious exponential plot of how error increases as step size increases was also generated and can be seen below:

Figure 1. Plot illustrating the relationship between absolute error and step size for the rk4step function.



Part 2: rk4

In this part, the rk4step function was utilized to perform a complete fourth order Runge-Kutta approximation on the simple harmonic motion differential equation as well as the Van der Pol Oscillator equation.

Methodology

Given a certain interval, the rk4 function simply iterates over it at a specified step size and calls the rk4step function per call.

This function was tested by performing a convergence test on the solutions corresponding to the simple harmonic motion equations as well as observing the desired behaviour in position and phase in the Van der Pol Oscillator solutions. As the name implies, in the convergence test, the errors between solutions corresponding to different levels were observed for convergence behaviour. In the position plot of the Van der Pol Oscillator solutions a stable and consistent oscillation was expected. In the phase plot, an eventual stable phase shape was expected.

PHYS 410
Melvin Mathews
57181142

File Descriptions

- rk4.m: This file contains the function which given an array of valid independent variables, a differential equation (in canonical form) function handle, and an initial condition, produces an approximate solution using the fourth order Runge-Kutta method.
- rk4_harmonic_test.m: In this file, the conditions and parameters provided by the project description were used to run the rk4 function. A convergence test was performed with the output solutions and the relevant tests were generated.
- vanderpol.m: File containing the Van der Pol Oscillator differential equation cast in canonical form. This is used to test rk4 as well as the method discussed in part 3 of this report. The equation(s) can be explicitly seen in the equations below:

$$\frac{d^2y(t)}{dt^2} + a(y^2 - 1)\frac{dy}{dt} + y = bsin(wt)$$
$$\frac{dy_1}{dt} = y_2, \frac{dy_2}{dt} = bsin(wt) - y_1 - a({y_1}^2 - 1)y_2$$

  It should be mentioned that while this is not explicitly stated in this file, this function was always tested with the corresponding initial conditions:

$$y_1(0) = y(0) = 0, y_2(0) = \frac{dy}{dt}(0) = -6$$

  As well as the parameters:

$$a = 5, b = 0$$

- rk4_vanderpol_test.m: In this file, the conditions and parameters provided by the project description were used to run the rk4 function and relevant plots were generated.

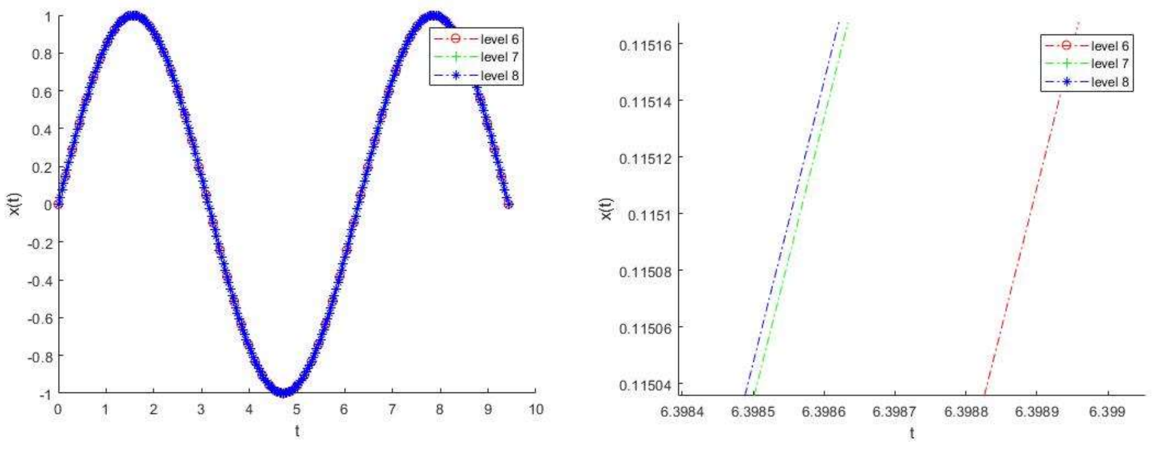
Results

The following section is split into two sections detailing the results produced by the harmonic and vanderpol testing.

Harmonic

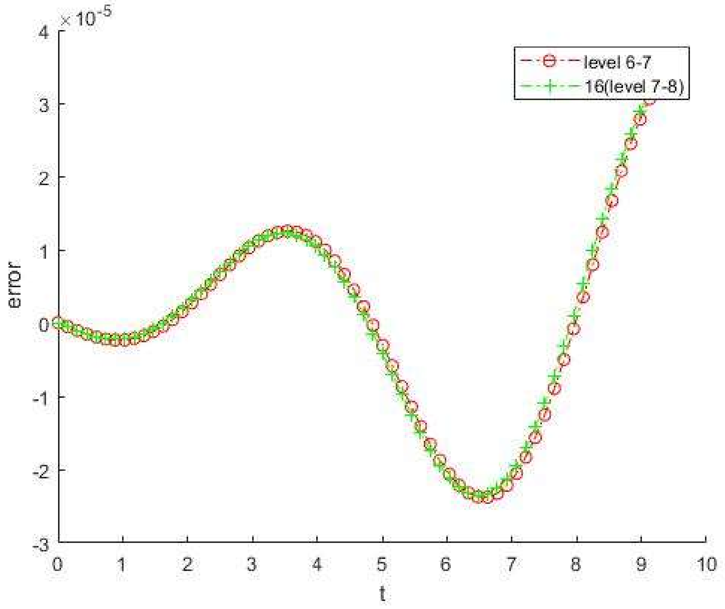
Levels 6, 7, and 8 were used to generate linearly spaced independent variable values as input into the rk4 function. The plots of the output can be seen in **Figure 2** below.

Figure 2. Plot of dependent variable x(t) with respect to t for varying levels 6, 7, 8. A zoomed in version of the plot is also shown.



The outputs corresponding to the levels were then subtracted and scaled appropriately to generate the figure below. The level 7-8 difference was scaled up by a factor of 16 to prove that a factor n decreases in Δt can be accounted for by a 4^n decrease in error. The overlapping plots in **Figure 3** confirm that rk4 converges on the order of Δt^4.

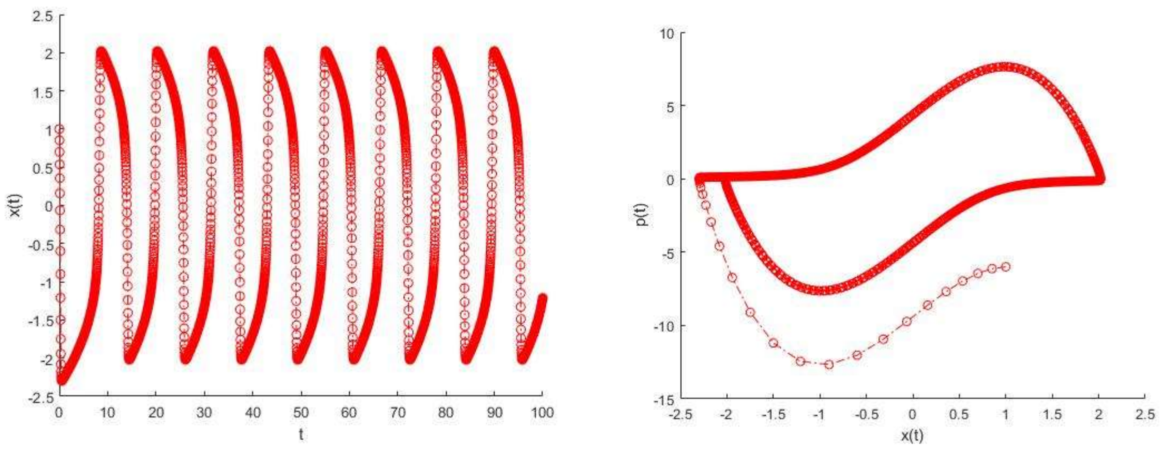
Figure 3. Plot of Scaled error levels with respect to independent variable t.



Van der Pol

A level of 12 with a span going from 0 to 100 as per the project documentation was input into rk4 along with the Van der Pol function specified earlier. The output was used to generate appropriate oscillator position and phase plots which can be seen in **Figure 4** below.

Figure 4. Plots illustrating the oscillator displacement (left) and phase space evolutions (right) of the Van der Pol Oscillator generated by rk4



The left figure displays expected behavior for the Van der Pol Oscillator by coming to and maintaining a consistent change in position with respect to time. The right figure is also expected as the system starts in a non-equilibrium position but soon reaches a limit cycle.

Part 3: rk4ad

In this part, an adaptive Runge-Kutta method was developed such that the step size could be modified to reduce local error.

Methodology

There are a few key logical steps in the algorithm developed: local error calculation, nested while loops, and dt modification.

Local Error Calculation

Much of the calculation is outlined in the project description so the following equations will only include the key equations relevant to the algorithm. The first step is to define a y_C that is the output of the rk4step function from above for a full step size. This y_C can also be equated such that:

$$y_c(t_0 + \Delta t) \approx y_{exact}(t_0 + \Delta t) + k(t_0)\Delta t^5$$

This equation comes directly from the definition of the Runge-Kutta method. Any calculated solution is simply the exact solution plus some error. It is for this reason that we can determine a value e_C:

$$e_C = y_c(t_0 + \Delta t) - y_{exact}(t_0 + \Delta t) \approx k(t_0)\Delta t^5$$

The rest of the procedure involves calculating this error without knowing the exact solution. This is significant as e_C is a very powerful quantity in regards to evaluating the robustness of this algorithm but it is not feasible to calculate it from the above equation in a case where the y_{exact} cannot be analytically determined. It is for this reason we introduce y_f which is achieved by performing the Runge-Kutta method on the same interval as y_c but in half increments such that we define:

$$y_f(t_0 + \Delta t) \approx y_{exact}(t_0 + \Delta t) + 2k(t_0)(\frac{\Delta t}{2})^5$$

Now subtracting y_f from y_c the y_{exact} cancels out:

$$y_c(t_0 + \Delta t) - y_f(t_0 + \Delta t) \approx k(t_0)\Delta t^5 - 2k(t_0)(\frac{\Delta t}{2})^5$$

$$y_c(t_0 + \Delta t) - y_f(t_0 + \Delta t) \approx k(t_0)\Delta t^5 - \frac{2}{32}k(t_0)\Delta t^5$$

$$y_c(t_0 + \Delta t) - y_f(t_0 + \Delta t) \approx \frac{30}{32}k(t_0)\Delta t^5$$

$$y_c(t_0 + \Delta t) - y_f(t_0 + \Delta t) \approx \frac{15}{16}e_C$$

Finally isolating for e_C gives the equation essential for the algorithm:

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

Nested While Loops

Although not entirely performance efficient, the final algorithm involves 2 nested while loops within the overall larger for loop. The larger for loop serves the same purpose as in rk4 of iterating through the independent variable values.

The first of the two nested while loops only breaks when the current value of the independent variable is greater than the next value in tspan. Once the first while loop breaks, the corresponding calculated y value is written to the output matrix. This ensures that the approximate solution is only calculated at precisely the times defined in tspan. Of course, this introduces an inefficiency in that the step size is made smaller than strictly necessary, but it is a justified cost to meet the requirements.

The second for loop iteratively calculates the local error and reduces step size by a factor discussed in the next subsection until the calculated error is less than the reltol parameter. It should be noted that a minimum value of dt is enforced here in that dt is set to 1.0e-4 if it is smaller than that value.

dt Modification

A suitable method for reducing dt had to be determined such that it wasn't too large of a change that inefficiencies were needlessly introduced but also not too small of a change such that the program would take an unreasonable amount of time to execute. An external resource was consulted: NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING. More specifically, section 16.2 of the book provided the following equation for step size change:

$$\Delta t_{new} = \Delta t_{old} \left|\frac{reltol}{e_c}\right|^{0.2}$$

However, it was found that the exponent of 0.2 took too long to execute for diminishing returns and was thus increased to 0.7 for a final equation:

$$\Delta t_{new} = \Delta t_{old} \left|\frac{reltol}{e_c}\right|^{0.7}$$

File Descriptions

- rk4ad.m: This file contains the function which given an array of valid independent variables, a differential equation (in canonical form) function handle, a relative tolerance and an initial condition, produces an approximate solution using the fourth order Runge-Kutta method.
- rk4ad_harmonic_test.m: In this file, the conditions and parameters provided by the project description were used to run the rk4 function. A convergence test was performed with the output solutions and the relevant tests were generated.
- rk4ad_vanderpol_test.m: In this file, the conditions and parameters provided by the project description were used to run the rk4 function and relevant plots were generated.

Results

The following section is split into two sections detailing the results produced by the harmonic and vanderpol testing.

Harmonic

The provided tspan in the project description along with the specified reltol values were used to test the rk4ad function. These outputs were then subtracted from the analytical solution of trigonometric sin tog generate the relevant plots seen in **Figure 5**. It makes sense that the error increases dramatically as the independent variable increases since that means that more error is being accumulated. The shape of the curves being slightly periodic can be attributed to the periodic nature of the exact solution.

Van der Pol

The provided tspan in the project description along with the specified reltol value were used to test the rk4ad function further. The oscillator displacement and phase space evolution plots (**Figure 6**) were then generated using the solution provided by rk4ad. Both graphs are very similar to **Figure 4** and display expected behaviour for the same reasons mentioned earlier.

Figure 5. Plots showing relative error evolution between the approximate solution produced by rk4ad and the analytical solution of sin.

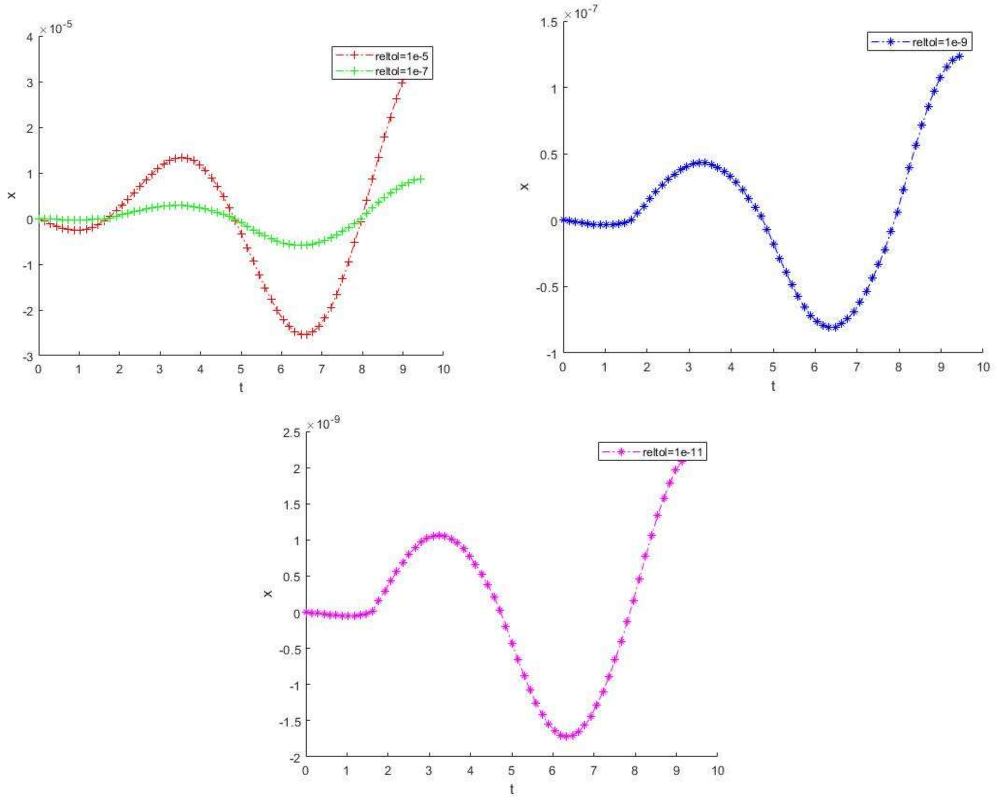
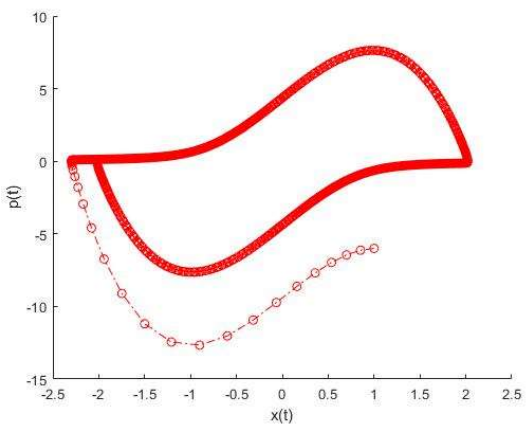
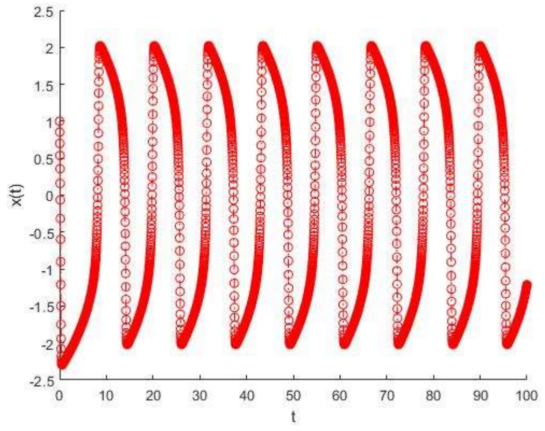


Figure 6. Plots illustrating the oscillator displacement (left) and phase space evolutions (right) of the Van der Pol Oscillator generated by rk4ad


p(t)
x(t)
10
5
0
-5
-10
-15
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


x(t)
t
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
10
20
30
40
50
60
70
80
90
100