ECSE 443 – Final Project

McGill University

Tristan Bouchard - 260747124

April 11, 2019

Introduction

This final project presented us with a collection of problems that required us to use the methods and techniques we had implemented over the course of the past assignments, as well as develop new solutions to new problems. Many of the problems combined two or more methods in the same question, which is where code modularity from past assignments came in handy. In this report, I will attempt, for the sake of brevity, to not re-describe the theory behind the algorithms that were developed in previous assignments. Instead, I will focus on explaining my approach to the problem and how I solved it, explaining the theory to new algorithms as necessary.

Question 1 – Single Variable Integration

This question asked us to evaluate the following integral:

$$I = \int_0^\infty \frac{1}{e^x + e^{-x}} dx$$

Using the Simpson method. By graphing this function, a few things are obvious:

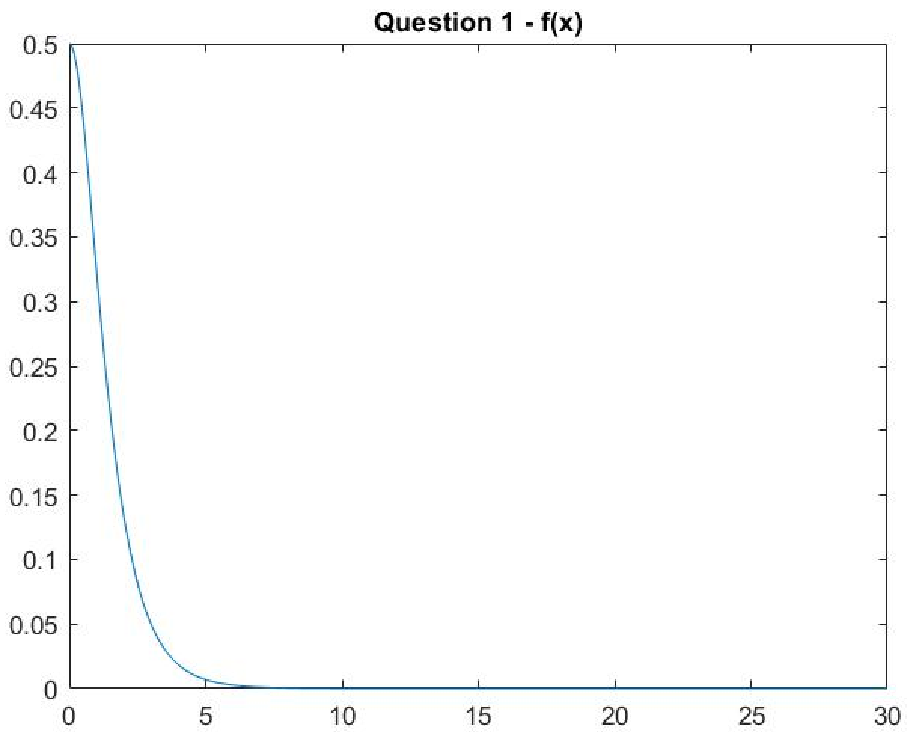


Figure 1 - Graph of f(x) for question 1

First, I have calculated the true value of the integral using the inbuilt MATLAB function, int(), which gives me the result:

$$I_{true} = \frac{\pi}{4} \cong \; 0.785398163 \ldots$$

It is easy to see that much of the information in the integral is encoded in the range of x between [0 15]. This function decays very rapidly to an asymptote of $x = 0$, which means that past approximately $x = 15$, the area under the curve is negligible. As we are using a numerical method that does, however, make approximations and is not exact, I chose to integrate the function from $x = 0$ to $x = 30$.

The result I obtain is the following:

$$I_{computed} = \; 0.785398088205527$$

Which has a relative error of $-9.57 * 10^{-8}$ with the true value above. The result was computed using 50 segments, each having a length of $x = 0.6$ each.

Question 2 – Single Variable Integration

This question asked us to evaluate the following integral:

$$I = \int_{0}^{1} \frac{x^3}{\sqrt{1-x^2}}$$

Using the Midpoint integral method, with segments of length $h = 0.01$. Using the code I had written for assignment 4 and pulling it out of the function it was in originally, such that I could use it and modify it to depend on the number of divisions, instead of it being based on the error. The result I obtained is as follows:

$$I_{h=0.01} = \; 0.623775746323755$$

Question 3 – Double Variable Integration

This question required us to verify Gauss's law for electric fields by performing the integration to both sides of the equation independently. Gauss's law is defined as:

$$\int AdL = \oiint D\, dS$$

Where the left integral describes the total charge present on a line of charge over a certain length (A being the line charge density, in Coulombs per meter) and the right portion describes the electric flux leaving the surface of a closed Gaussian surface. The scenario for the integral is as follows: It is necessary to find the total flux of the electric field caused by a wire of charge of infinite length passing through the origin by using a Gaussian surface in the shape of a sphere of 2m in radius, centered at the origin. Now, the left integral is formed:

$$I = \int_{-2}^{2} 12.4 * 10^{-6} dx$$

It was necessary to integrate this value using an interval length of $h = 0.01$ and as such, I re-used the code I had written in question 2 and modified the values to compute the integral. The calculation of the integral was composed of 400 segments of length 0.01, which yields the result:

$$I_{h=0.01} = 4.960000000000009 * 10^{-5}$$

Now, the right side of the integral required a little more work, and a little dusting off of Calculus techniques as well as old EMF material. First off, it was necessary to integrate the field in spherical coordinates, as this was the coordinate system imposed by the spherical Gaussian surface. The spherical coordinate convention I used it as follows:

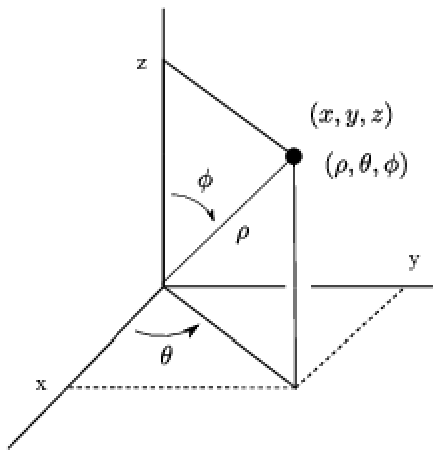


Figure 2 - Spherical Coordinates system used in question 3

With this out of the way, it was time to set up the integral. The D field of a wire of charge is described by:

$$D = \frac{\rho}{2\pi r}$$

Which is defined in cylindrical coordinates. Now, the integral looks like:

$$\iint \frac{\rho}{2\pi r}\, dS = \iint \frac{12.4 * 10^{-6}}{2\pi r}\, dS$$

However, to convert this integral to spherical coordinates, it is necessary to convert the radius described in cylindrical coordinates, as well as the dS vector, into spherical coordinates. This is achieved through the transformation of the Jacobian matrix:

$$\iint \frac{12.4 * 10^{-6}}{2\pi r}\, dS = \int_0^{2\pi}\int_0^{\pi} \frac{12.4 * 10^{-6}}{2\pi r} * r^2 \sin(\phi)\, d\phi d\theta$$

Where,

$$r = 2$$

Now, integrating using the Simpson method for two variables:

$$I = 4.960016825415667 * 10^{-5}$$

This result was computed using 10 segments in x of length 0.62832 each. The computed error between the left integral and the right integral:

$$Error = 1.682541565789638 * 10^{-10}$$

Which confirms that the results are well within 10% of each other.

Question 4 – Second Order Response: Curve Fitting and Interpolation

This question is one of the questions which I mentioned earlier which couples many techniques seen in the past assignments into one problem. We are presented with a set of data points and are asked to obtain the coefficients of the second order ODE. The method to obtain these coefficients is using the method developed in assignment 2, which is of normal equations. However, to perform this, it is necessary to obtain values to the first and second derivatives of the function, as well as values of the function itself, at different points in time. Fortunately, we have access to the data points of the function. We can thus use the method of finite difference, coupled with normal equations, to solve for our coefficients. The function is defined as:

$$A * \frac{d^2y(t)}{dt^2} + B * \frac{dy(t)}{dt} + c * y(t) = x(t)$$

For the sake of brevity in notation, I will use

$$\frac{d^2y(t)}{dt^2} = f''(t)$$

And

$$\frac{dy(t)}{dt} = f'(t)$$

Using first and second order forward difference derivatives:

$$f'(t_i) = \frac{f(t_{i+1}) - f(t_i)}{h}$$

$$f''(t_i) = \frac{f(t_{i+2}) - 2 * f(t_{i+1}) + f(t_i)}{h^2}$$

And backward difference derivatives:

$$f'(t_i) = \frac{f(t_i) - f(t_{i-1})}{h}$$

$$f''(t_i) = \frac{f(t_i) - 2 * f(t_{i-1}) + f(t_{i-2})}{h^2}$$

And the Normal Equations:

$$\begin{bmatrix} f''(t_1) & f'(t) & f(t_1) \\ \vdots & \vdots & \vdots \\ f''(t_n) & f'(t_n) & f(t_n) \end{bmatrix} * \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_n) \end{bmatrix}$$

However, a little problem stands in our way before beginning: The fact that we have no data point at $t = 0.25$. This messes up our problem, as we absolutely need evenly spaced points to be able to use the finite difference method. As such, I chose to use my Cubic Splines method which I made back in assignment 2, to compute a value for that data point. The data therefore graphs as following:

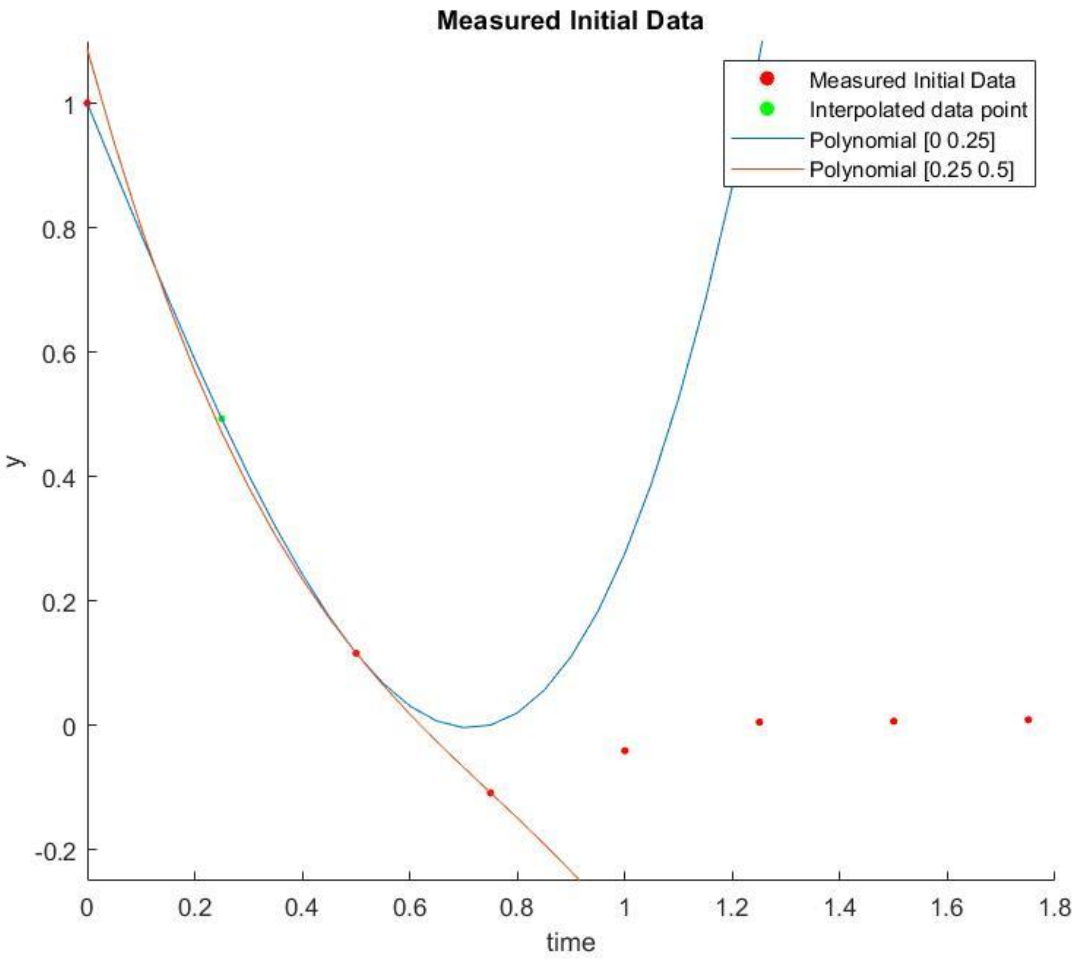


Figure 3 - Question 4 datapoints

So, using the interpolated point to compute the normal equations with the finite difference derivatives we can obtain the coefficients to the ODE. Note that I used the forward difference equations for datapoints 1 through 6, and the backward difference equations for points 7 and 8.

The coefficients computed by my algorithm:

A	B	C
-0.021893989616317	-0.674262825491594	-0.216244070156047

Question 5 – LU Factorization

This question required us to perform LU factorization on a system of equations and solve for the values of x, as was done in assignment 1. I chose to perform this using the standard LU factorization, and then using

$$L * U = A$$

To compute the values of x. Then, by solving the equation

$$U^{-1} * L^{-1} * y$$

I obtain the values for x as follows:

Variable	Computed Value	True Value	Square Error
X1	-1.077179830921179	-1.077179830921178	3.155443620884047e-30
X2	1.990205466334710	1.990205466334710	4.930380657631324e-32
X3	1.474706574375529	1.474706574375529	4.930380657631324e-32
X4	-1.906432108560638	-1.906432108560640	1.232595164407831e-30

Question 6 – Fixed Point Iteration

This question asked us to perform fixed point iteration method on the system of equations from question 5. As per the example seen on *myCourses*, it is simply a question of defining each individual variable in terms of the other ones, and to set each variable equal to zero at the start. Then iterate though the equations, plugging in the new value of each variable as iterations continue. To use this method, it is important to place the largest values of the row in the diagonal of the matrix. For example, I have rearranged the system from:

$$\begin{bmatrix} 3 & -5 & 47 & 20 \\ 11 & 16 & 17 & 10 \\ 56 & 22 & 11 & -18 \\ 17 & 66 & -12 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 18 \\ 26 \\ 34 \\ 82 \end{bmatrix}$$

To

$$\begin{matrix} 56 & 22 & 11 & -18 \\ 17 & 66 & -12 & 7 \\ 3 & -5 & 47 & 20 \\ 11 & 16 & 17 & 10 \end{matrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{matrix} 34 \\ 82 \\ 18 \\ 26 \end{matrix}$$

Where the main diagonals are the largest values in the rows. Then, solving for each variable row by row, for example:

$$x_1 = \frac{34 - 22x_2 - 11x_3 + 18x_4}{56}$$

We begin iterating at $x_1 = x_2 = x_3 = x_4 = 0$, then progressively plug in the values as we iterate. The values obtained after 1000 iterations:

Variable	Computed Value	True Value	Square Error
X1	-1.077179830921179	-1.077179830921178	1.774937036747277e-30
X2	1.990205466334711	1.990205466334710	7.888609052210118e-31
X3	1.474706574375531	1.474706574375529	2.415886522239349e-30
X4	-1.906432108560642	-1.906432108560640	5.965760595733902e-30

Question 7 – Secant Method

This question required us to find the next real root of the function

$$f(x) = \cos(x)\cosh(x) - 1$$

Using the secant method developed and explained in assignment 3, to which I give as starting points $x = 4.5$ and $x = 5$ the result for the root:

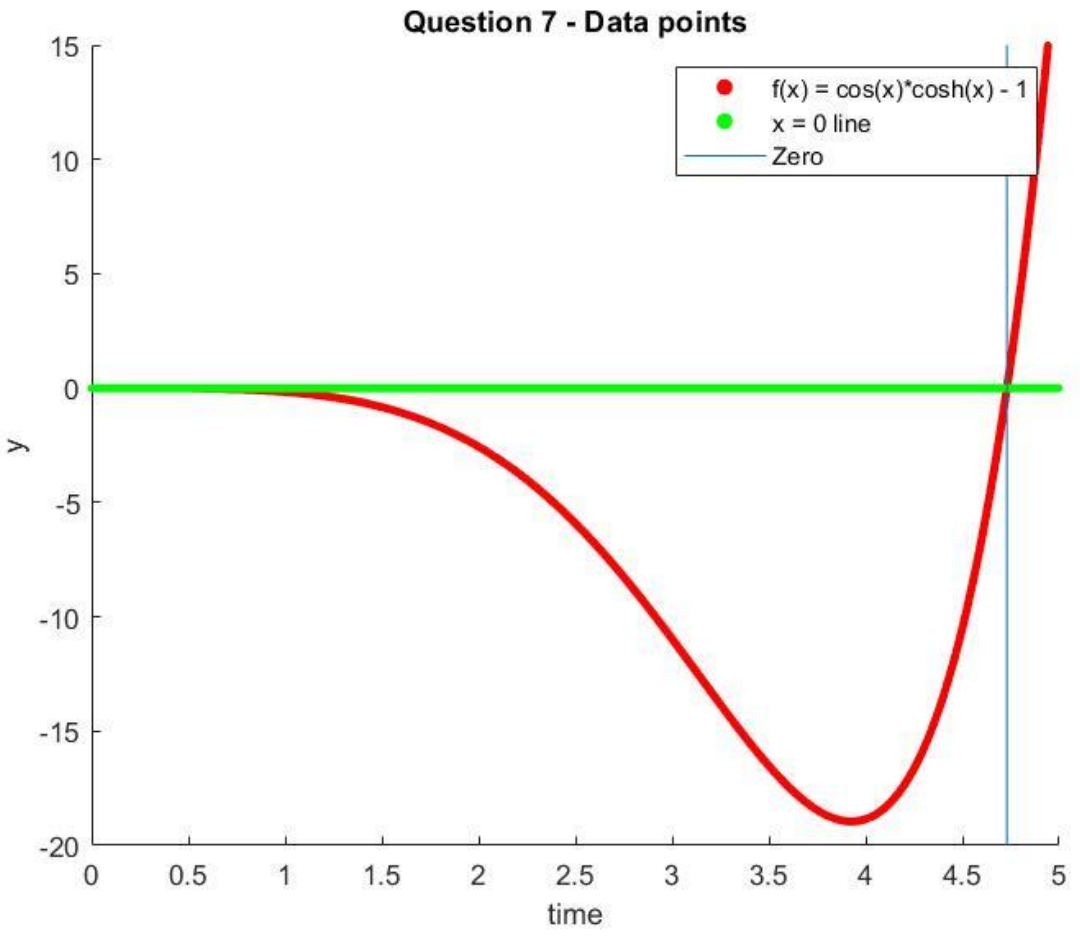


Figure 4 - Question 7 function

$$root = 4.730040742490636$$

This result was computed using the algorithm I developed in assignment 2 using a precision of 10^{-4}.

Question 8 – Polynomial Curve Fitting

This question required us to make use of our curve fitting algorithm from assignment 3. More particularly, it is specified that the set of points supplied to us in the prompt appears to follow a second order polynomial. Perfect, we have developed a curve fitting algorithm in assignment 3, which made use of normal equations to compute the second order polynomial which fit best a set of data! Feeding the datapoints into this function, we obtain the following graph:

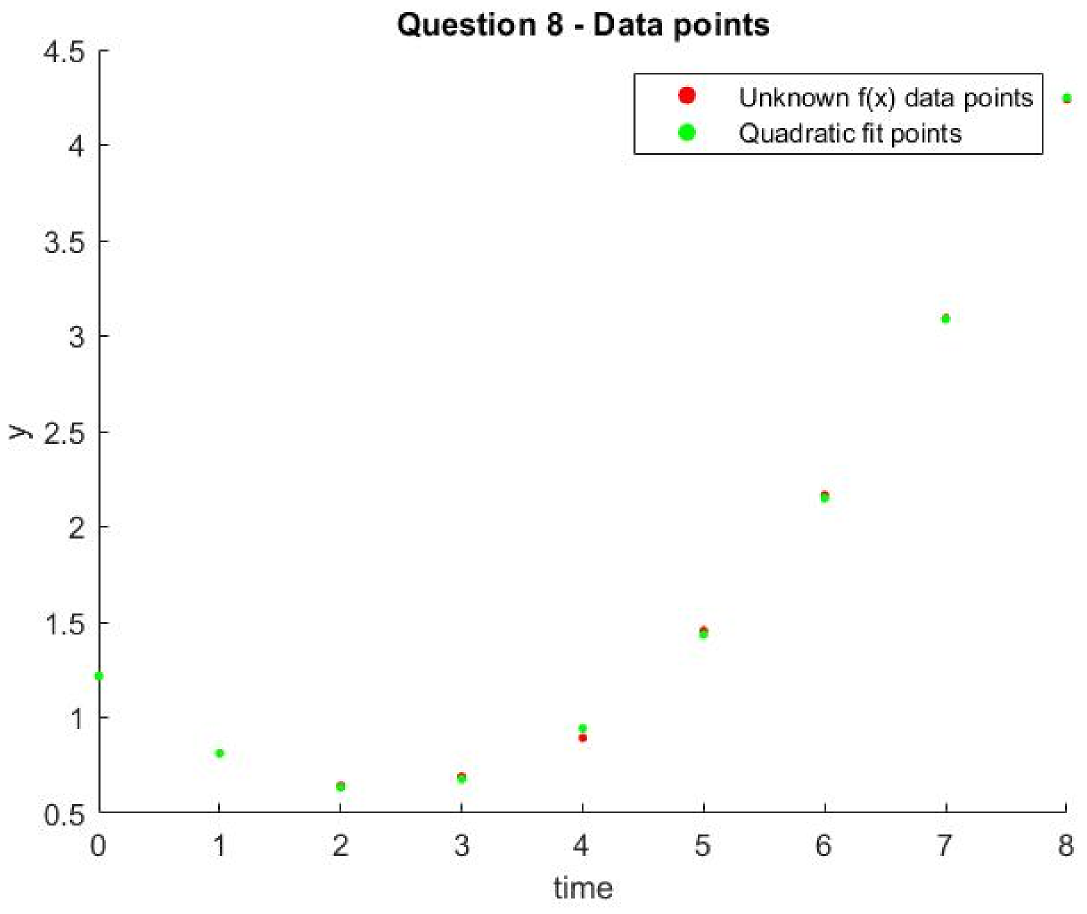


Figure 5 - Question 8 - Data points and quadratic fit points

The error between each of the data points is follows:

Time	Original Data	Quadratic Data	Error
1	0.812	0.813999999999998	-0.001999999999998
2	0.642	0.633428571428570	0.008571428571430
3	0.691	0.676642857142857	0.014357142857143
4	0.893	0.943642857142857	-0.050642857142857
5	1.454	1.434428571428572	0.019571428571428
6	2.164	2.149000000000000	0.015000000000000
7	3.092	3.087357142857143	0.004642857142857
8	4.24	4.249499999999999	-0.009499999999999

By observing the error between the original data points and the points computed by the quadratic fit function, it is obvious that the datapoint at $t = 4$ is erroneous, as it has the biggest error. To correct it, I would set its value equal to that of the quadratic fit function, i.e. $f(4) = 0.943642857142857$.

Question 9 – ODE's: Euler and Runge-Kutta-4 Methods

This question prompts us to solve an ODE using two different numerical methods: The Euler and fourth order Runge-Kutta methods. Given the ODE:

$$\frac{d^2y(t)}{dt^2} + 2\frac{dy(t)}{dt} + 4y(t) = 0$$

For $0 \le t \le 5$, we are to compare both methods.

To perform this numerical solution, it is necessary to convert the second order equation into two first order equations. This is done by letting $\frac{dy(t)}{dt} = u(t)$. Thus, solving for $\frac{d^2y(t)}{dt^2}$:

$$f_1(t, u, y) = \frac{dy(t)}{dt} = u(t)$$

$$f_2(t, u\,, y) = \frac{d^2y(t)}{dt^2} = u'(t) = -2 * u(t) - 4 * y(t)$$

Then, Euler's method is applied, using both equations above to compute the next point of the function:

$$y_{i+1} = y_i + h * f_1(t_i, u_i, y_i)$$

$$u_{i+1} = u_i + h * f_2(t_i, u_i, y_i)$$

Using a step size of $h = 0.1$, as prompted in the problem, it is then possible to obtain the data points to the function, from 0 to 5.

The Runge-Kutta4th order method makes use of the Euler method, but obtains more information:

$$k_1 = f_1(t_i, y_i, u_i)$$

$$k_2 = f_1(t_i + \frac{h}{2}, u_i + \frac{l_1 h}{2} y_i + \frac{k_1 h}{2})$$

$$k_3 = f_1(t_i + \frac{h}{2}, u_i + \frac{l_2 h}{2}, y_i + \frac{k_2 h}{2})$$

$$k_4 = f_1(t_i + \frac{h}{2}, u_i + \frac{l_3 h}{2}, y_i + \frac{k_3 h}{2})$$

$$l_1 = f_2(t_i, y_i, u_i)$$

$$l_2 = f_2(t_i + \frac{h}{2}, u_i + \frac{l_1 h}{2}, y_i + \frac{k_1 h}{2})$$

$$l_3 = f_2(t_i + \frac{h}{2}, u + \frac{l_2 h}{2}, y_i + \frac{k_2 h}{2})$$

$$l_4 = f_2(t_i + \frac{h}{2}, u_i + \frac{l_3 h}{2}, y_i + \frac{k_3 h}{2})$$

Then, using these computed values, the Runge-Kutta4th method uses the following equations to compute the next points in the function:

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

$$u_{i+1} = u_i + \frac{h}{6}(l_1 + 2l_2 + 2l_3 + l_4)$$

Graphing the solutions from the above equations:

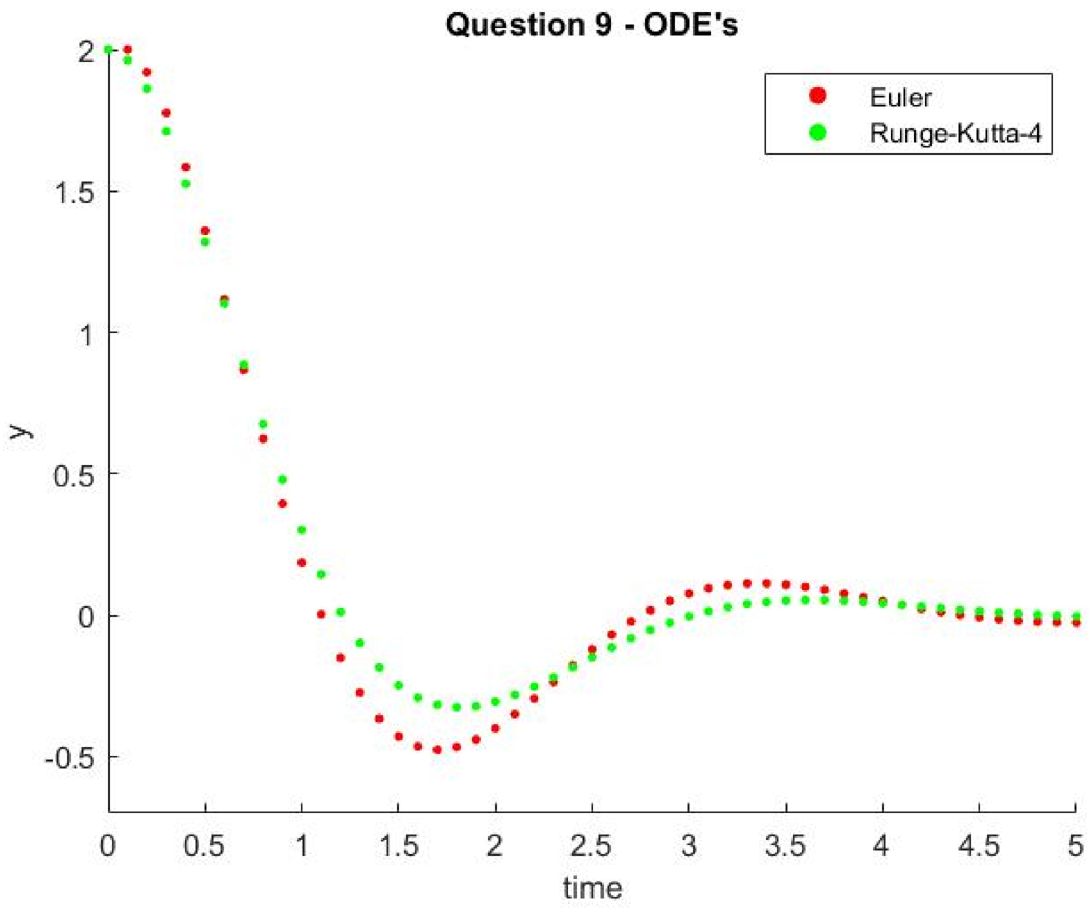


Figure 6 - Numerical Methods used in question 9

Comparing both methods, it is clear to see that there are some discrepancies between the two models. This is to be expected, simply due to the precision of either algorithm: The Euler Method is a first order method, while the Runge-Kutta4 method is a 4th order method which is expected to be more precise. Also, it is known that the Euler methods tends to accrue error faster as it proceeds, which is why it oscillates more compared to a higher order method, such that the Runge-Kutta4 method is more precise.

Question 10 – ODE's: Runge-Kutta-2 Method

This problem required us to perform a very similar analysis to question 9, but this time instead using the second order Runge-Kutta2 method on a simpler, first order differential equation. The equation:

$$\frac{dy}{dx} = 1.2y + 7e^{-0.3x}$$

Is to be evaluated from $x = 0$ to $x = 2$ with a step size of $h = 0.1$, given that $y(x = 0) = 3$. In a very similar manner to question 9 above, it is necessary to define K's, which the Runge-Kutta2 Method uses to achieve greater precision. However, there are many types of methods for the second order method, and such I chose to use the Heun method, defined as:

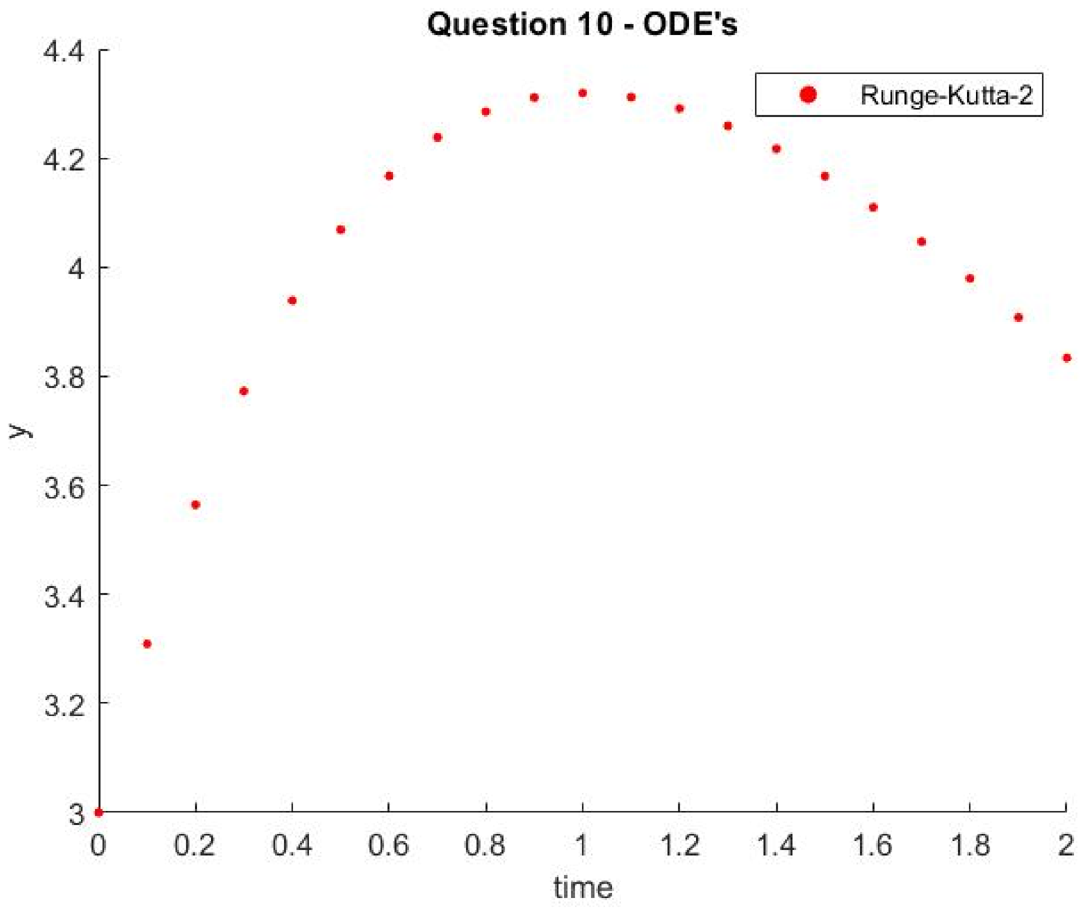
$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + k_1 h)$$

And

$$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2)$$

Using this method to solve for the array of points numerically, we obtain:



Question 11 – ODE's: BVP with Finite Difference Method

This question required us to obtain the points to an ODE using the finite difference method. The ODE

$$\frac{d^2y(t)}{dt^2}+\frac{1}{4}\frac{dy(t)}{dt}=8$$

Is given, along with the boundaries $y(0)=0$ and $y(10)=0$. We are also to use steps $\Delta x=1$. Using the central difference equations makes sense in this problem, as we know the endpoints of the function. As such, for the first and second order central finite difference:

$$\frac{dy(t)}{dt}=\frac{y_{i+1}-y_{i-1}}{2h}$$

$$\frac{d^2y(t)}{dt^2}=\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}$$

When substituting into the above ODE, we obtain:

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}+\frac{1}{4}\frac{y_{i+1}-y_{i-1}}{2h}=8$$

Which yields, after simplification (using $h=\ \Delta x=1$):

$$1.125y_{i+1}-2y_i+0.875y_{i-1}=8$$

Such, it is possible to create a matrix of inner nodes, which is 9 by 9, as there are 9 inner points. Note that the first and last inner nodes integrate the boundary conditions, such that in the first row, $y_{i-1}=y(0)=\ 0$ and in the ninth row, $y_{i+1}=y(10)=0$.

$$\begin{bmatrix} -2 & 1.125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.875 & -2 & 1.125 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.875 & -2 & 1.125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.875 & -2 & 1.125 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.875 & -2 & 1.125 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.875 & -2 & 1.125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.875 & -2 & 1.125 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.875 & -2 & 1.125 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.875 & -2 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{bmatrix}=\begin{bmatrix} 8 \\ 8 \\ 8 \\ 8 \\ 8 \\ 8 \\ 8 \\ 8 \\ 8 \end{bmatrix}$$

Solving for the values of y_1-y_8 and adding the boundary conditions to the beginning and end of the array, we can plot the computed ODE:

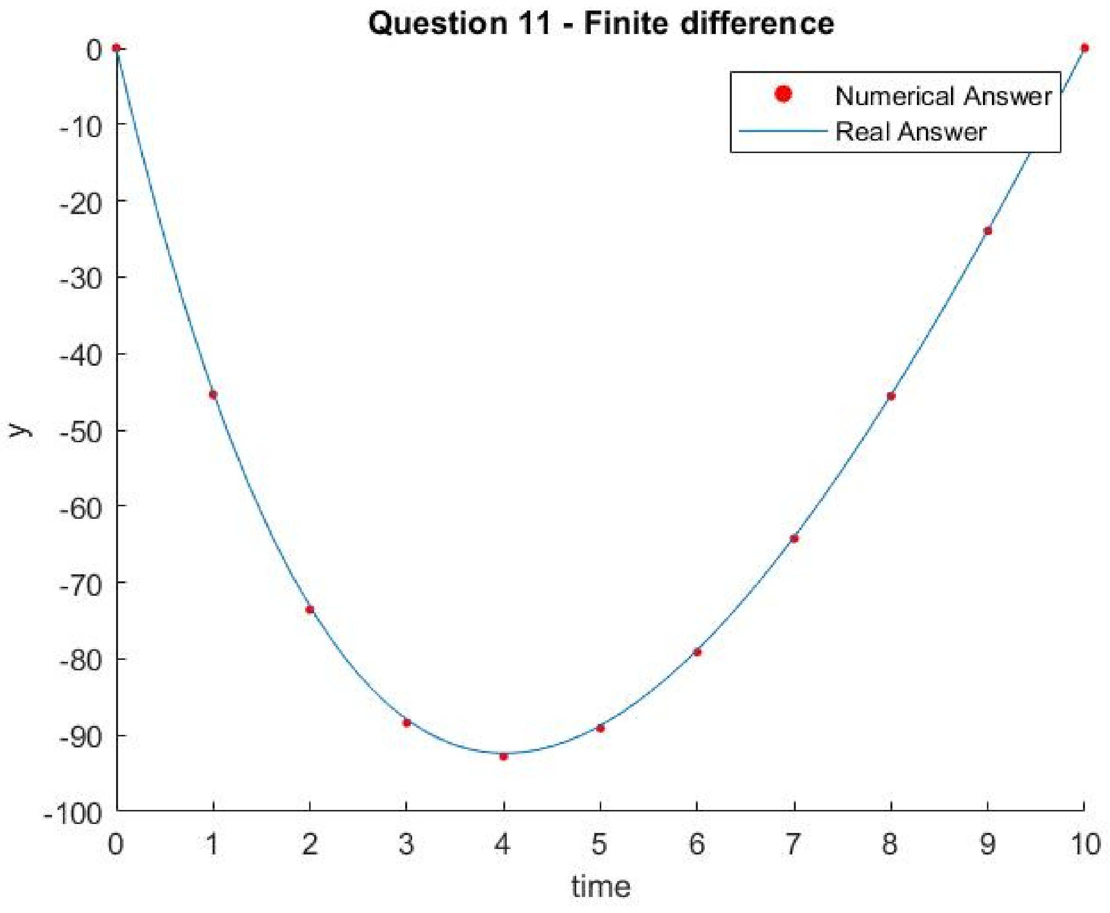


Figure 7 - Question 11 ODE's

The true solution was found using the built-in MATLAB functions “diff” and “dsolve”. It is obvious to see that the Finite difference method is very precise for this type of ODE, as it matches nearly perfectly with the true answer.

Question 12 – ODE’s: BVP with Shooting Method

This problem required us to solve for the points of the ODE using the Shooting method. Essentially, the goal of the method is, given the initial value of the function and the final value of the function, to use a method such as Runge-Kutta4 (as used in question 9) to compute the points to the ODE, and iterate over initial values of $\frac{dy(t=0)}{dt}$, such that the value of the computed point at $t = 10$ matches as much as possible to the actual boundary, $y(10) = 0$.

Such, looking at the graph from the previous question, it is obvious that the first derivative of the function at $t = 0$ is somewhere between $[-100 - 25]$. As such, I attempt the Runge-Kutta4 method using each of these values, incrementing in steps of 0.01. My goal is to find the initial value of the derivative which forces the ODE to pass through $y(10) \ = \ 0$.

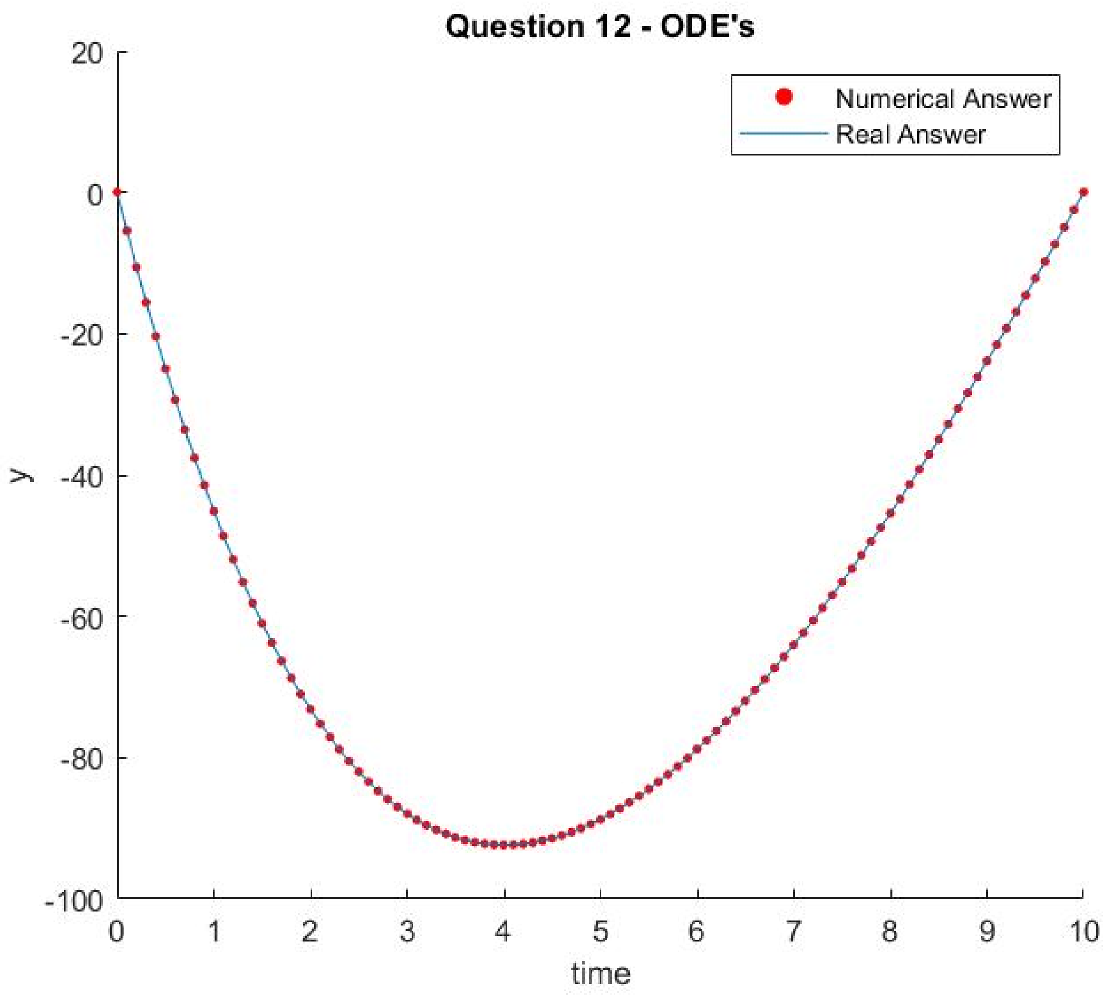


Figure 8 - Question 12: Shooting method

The experimental value of the first derivative using the Shooting Method:

$$y'(0) = -55.150000000000006$$

Which has an error at $y(10) = 0$ of:

$$error = 0.014830758063006$$

It is fairly obvious to see that this method is very precise yet is iterative and as such is more of a brute force method to solving these equations.