110770716

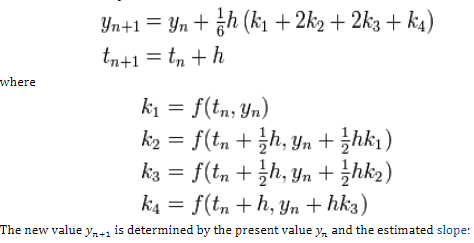
Darren Kong

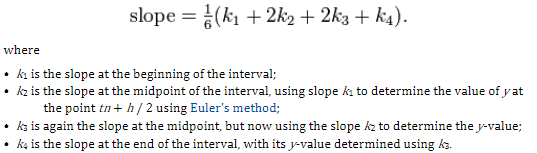
AMS326

Midterm 2

**Problem T2-2:**

This problem was straight forward and simple. The first thing that I created was a function called **f(x, y)**. This takes in two parameters x and y, and returns the value of . This is supposed to represent the differential equation . The next function is **runge\_kutta(f, x0, y0, h)**. The method takes in a function f, which was defined above. It also takes in values x0 and y0 which are the initial values given in the problem, . It also takes in one more parameter h, which is the step size specified, .01. The function approximates the next values for x and y based on the parameters. In other words, the formulas are shown below:

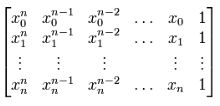




The entire **runge\_kutta()** method is thrown into a loop inside a method called problem2. The loop simply records the x and y values generated from each iteration of the runge\_kutta method and returns both the x and y values in two respective lists called **xplot** and **yplot** respectively. This is then plotted onto a graph represented by the blue dots.

The next step involves creating the functions **swap\_rows(m, i, j)**. This function simply takes a matrix m, and swaps the rows i and j and returns the resultant matrix. This is useful for the next function, which is the **guass\_elimination(m)**. The function takes in a matrix m and performs gaussian elimination on it. In other words, swapping rows and doing forward elimination until we have an upper triangular matrix. Then it does a back substitution to get the final answers.

But before we are able to use the guass\_elemination function we have to set up the correct matrix. This is where the **create\_interpolant\_matrix(Ys, Xs)** comes in. The Ys is the y-values for every Xs we chose, i.e. [.1, .2, .3, .4, .5]. It basically creates a matrix in the form below but with the “y” vector merged with the “x” matrix.



After creating the matrix, we pass it to the **guass\_elemination(m)** to get the resultant vector “a”. These will be the coefficients of our polynomial to fit . Furthermore, the method **p\_x(a\_coef, x)** is simply the above equation but returning the value given the parameters. We throw this function into a loop to get the y-values that satisfy the equation above using the same x-values from 0 to .5 with a step size of .01. The resultant y-values are stored in a list called **poly\_y.** We plot the values and this gives us the red line shown in the plot. It is a near indistinguishable fit.

