HW5 Report

Jesse harper
UID:204669893

Introduction 1.1:

This problem was on the use of splitting and averaging a given vector to reshape it their points. Two functions, one for splitting and another for averaging, were written and used in a driver function. Once these functions were written, they were tested with a number of weight vectors to see if there was any predictable pattern to how the weights affect the outcome of each plot.

Models and Methods 1.2:

Two functions, named *splitPts* and *averagePts* were written to produce the split and averaged vectors. The split function double the number of elements in the input vector and repopulates the new vector, every other element, with the old vector's contents. It then takes the mean of adjacent elements from the old one and incerts those values into the empty slots of the new vector and returns the new vector. The average function take a vector and a weight vector and performs a similar operation as split, except it keeps the same number of elements and it now does a weighted average of adjacent and local elements: using the normalized elements of the weight vector. Again this new vector is then returned.

A driver was written to iterate through successive calls of an X and Y initial vector. X and Y are averaged using the same Weight. These Vectors are iteratively reaveraged until the difference between their weighted average and split are less than 1e-3. Once this is finished a plot is made of X and Y.

Calculations and Results 1.3:

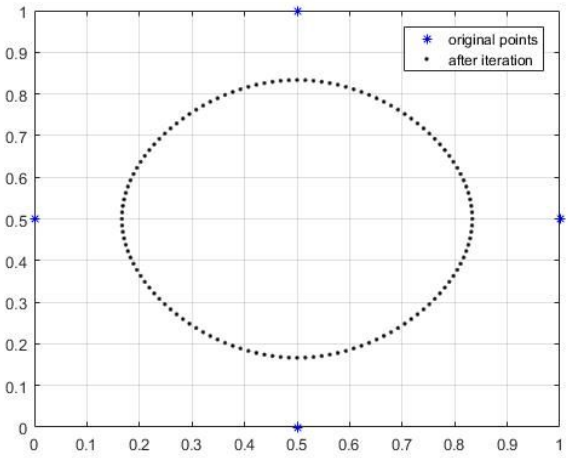

Figure 1.2.1: W = [1 2 1]

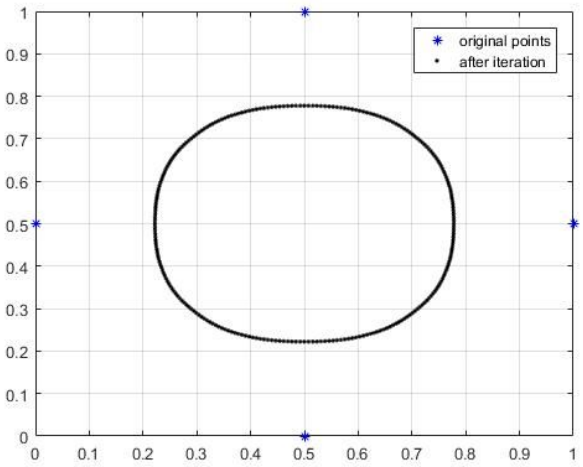

Figure 1.2.1: W = [2 1 2]

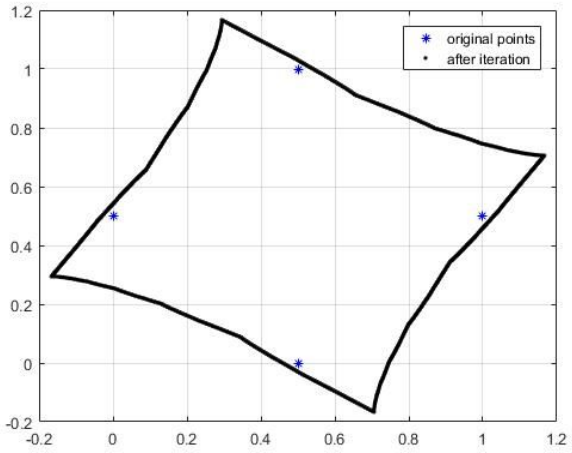

Figure 1.2.1: W = [10 1 -1]

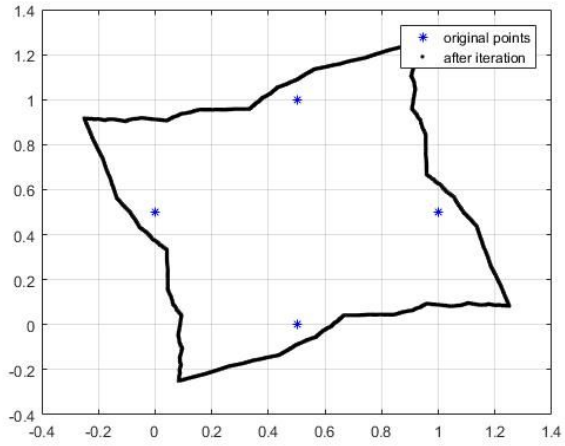

Figure 1.2.1: W = [-1 1 2]

Discussion 1.4:

A number of weights were tried and used to run the driver function on the same X and Y initial vectors. Not all weights would converge. Some converged faster than others. However I could not see a pattern to this. Weights that would converged usually did so un under 10 iterations. Some took longer, but typically 10 was enough. Additionally, different weights, when applied to the same X and Y vectors, yielded different shapes. A sample of four of the trials i ran are provided in the calculations and results section. Unfortunately, I could not see a pattern with this feature either.

Introduction 2.1:

This problem examines the method of Runge- Kutta to show its effectiveness in reducing error in numerical methods, when compared to first order approximation methods. The radioactive decay of carbon-15 was used to test this method. A function driven approach was used to implement the Runge-Kutta method as well as the first and second order approximations. Finally an average error was recorded for each method and results were compared to show that Runge-Kutta is a superior method for use in numerical methods problems.

Models and Methods 2.2:

The Runge-Kutta method is a fourth order numerical methods approximation which uses successive averages of slope in order to find a low error approximation.

$$y(k+1) = y(k) + \frac{1}{6}C1 + \frac{1}{3}C2 + \frac{1}{3}C3 + \frac{1}{6}C4 \qquad \textbf{eqn. 2.2.1}$$

$$C1 = \Delta t * (-ln(2)/2.45) * Yk \qquad \textbf{eqn 2.2.2}$$

$$C2 = \Delta t * (-ln(2)/2.45) * (Yk + \frac{C1}{2}) \qquad \textbf{eqn 2.2.3}$$

$$C3 = \Delta t * (-ln(2)/2.45) * (Yk + \frac{C2}{2}) \qquad \textbf{eqn 2.2.4}$$

$$C4 = \Delta t ** (-ln(2)/2.45) * (Yk + C3) \qquad \textbf{eqn 2.2.5}$$

The function *advanceRK* takes three parameters. The first is yk. This is the local value of y at time tk. The next one is the time step delta t. The last is a specifier for the method; Rk1, Rk2, or Rk4. At the beginning of the function, three local helper functions are defined which actually calculate the next time step value of Y. The rest of the program uses a switch case statement to decide which of the three methods to use and calls the appropriate helper function.

Calculations and Results 2.3:

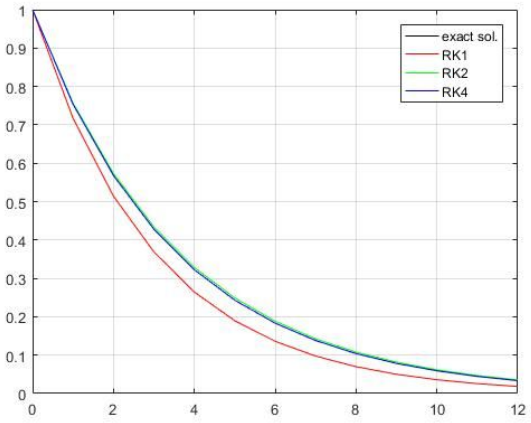

Figure 2.3.1: plots of y with dt =1

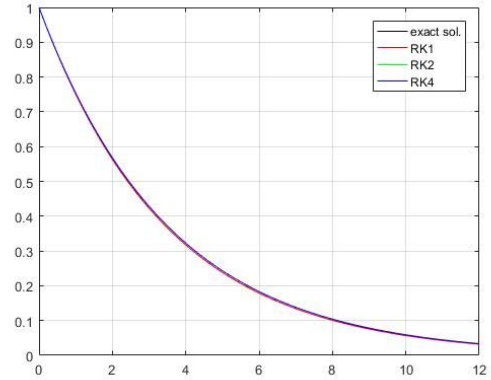

Figure 2.3.3: plots of y with dt =0.1

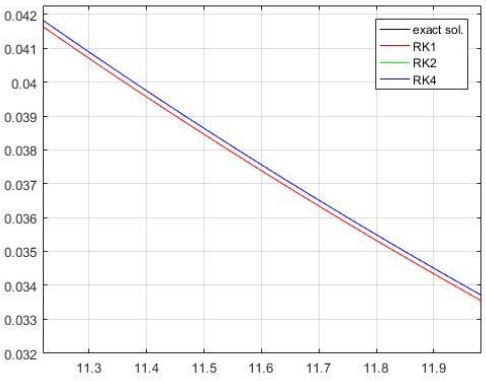

Figure 2.3.3: plots of y with dt =0.01

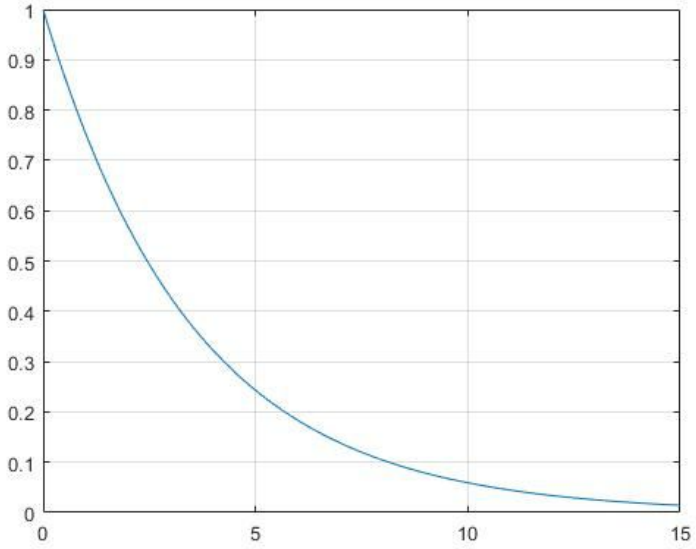

Figure 2.3.4: plot of RK4 out to t = 15

Following is the console output for the averages of each methods' error.

```
 dt         RK1         RK2         RK2
1.00:   3.60e-02   3.91e-03   1.58e-05
0.10:   3.56e-03   3.40e-05   1.36e-09
0.01:   3.55e-04   3.36e-07   1.34e-13
```

Discussion 2.4:

It's pretty clear, even with a time step of 1, that the fourth order Runge-Kutta method is much more accurate than the first and second order approximations. You can tell this from figure 2.3.1, the exact solution and fourth order solution are nearly overlapped. When examining the table of average errors for each time step and method, the order of magnitude that the error decreases per time step shows the order of approximation. The first order solution decrease by one order for every order of magnitude the step size decreases. Similarly, the second order solution decreases by two for every order and four for the fourth order approximation. Finally, using the fourth order approximation and a time step size of 0.01s the remaining carbon 15 at 15 seconds is only 1.4%.