Raymond Waidmann

18157816, rcw5k2

MAE 3100 Final Project Report

Part 1.1

The decision to test 3rd, 4th, and 5th order polynomials was because the data appeared to have 2 changes in concavity when plotted. It is common convention for polynomials of degree n to have n-2 changes in concavity, for example, a quadratic polynomial (degree 2) never changes concavity but a cubic (degree 3) changes concavity once. Thus, with 2 observed changes in concavity, it would make sense for a fourth order polynomial to fit best. To follow the project requirements and to get better results, it was decided to test the polynomials surrounding fourth order (3rd and 5th) and thus it was determined which polynomials we would be testing. Though the orbit of a satellite is probably more cyclic in nature – a sin or cosine function – the decision to use basic polynomials was done to keep an already complex problem as simple as possible.

See MatLab Command Window for table comparing error for different orders of polynomials. By looking at this table, it can clearly be seen that the fifth order polynomial is the one with the smallest squared residual; both the third and fourth order errors are on the magnitude of 1010 while the fifth order is on the magnitude of 109. Since the fifth order squared residual is the smallest, it can be concluded that is most accurately models the trajectory of the satellite from the given data and thus we will use this fit for the rest of part one.

Part 1.2

It can be seen from the comparison of the squared residuals in the command window that the 5th order polynomial is the best fit; the 5th order polynomial has the smallest squared residual. Thus, the resulting polynomial will be of the form:

Since we know the objective function is equal to the sum of the squares of the residuals, we can solve for that by solving for ei and squaring:

* Where n = length(t) = 32001

To minimize the objective function, we must take the partial derivates of the objective function with respect to each of the unknown coefficients and set them equal to zero. These are our necessary conditions:

By rearranging our necessary conditions, we can find the equations needed to determine the polynomial coefficients:

Since all the summations are known values, we can use Gauss Elimination to solve for a0­­, a­1, a2­­, a­3, a4­, and a­5. Using a program I made, called “GaussEliminationFnct\_FS19”, the coefficients of the above equations are sent to the function in matrix form and the function returns the solution to the coefficients of the regression polynomial. The solution from using “GaussEliminationFnct\_FS19” is as follows:

These coefficients result in a final regression polynomial of the following form:

Part 1.3

It can be clearly seen that the coefficients found via Gauss Elimination are identical to the ones found using the polyfit function within MatLab. The only difference is that “GaussEliminationFnct\_FS19” returns the coefficients in increasing powers of t, while polyfit returns the coefficients in decreasing powers of t. Nonetheless, the solutions from both methods are identical and can be easily seen within the MatLab workspace and command window.

Part 1.4

See MatLab Code and Figure 1. It is worth noting that the graphs from both polyfit and gauss elimination for each order of polynomial are plotted in the same color but with different line types. It may be difficult to see on the graph, but for a given polynomial order, the two graphs are identical and overlap. This further shows the results in Part 1.3 are identical regardless of the method used.

Part 1.5

It was decided to use golden section search because it provides relatively precise results while being straightforward to implement in MatLab. Two functions: “GoldSecSearchMax\_FS19”, and “GoldSecSearchMin\_FS19” were created to implement this part of the project while simultaneously keeping the main program clutter free. The time coordinates and corresponding distances are displayed in the command window and are circled on Figure 1.

It is worth noting that the global minimum of the fifth order polynomial within the bounds of the data occurs at t=0. However, this point is not considered in our search for the minimum as it occurs at the edge of our given data set. Thus, although the found min is not global minimum within the given interval, it is a more accurate representation of the true minimum of the given data set.

Part 2.1

See “RK\_FS19”. It was decided to use Ralston’s method for RK second order where a1 = 1/3, a2 = 2/3, and p1 = q11 = 3/4. “Ode\_Function\_FS19” simply takes the input position and velocities. The first half of the returned vector is the velocities, and the accelerations are calculated from the given function in the project outline. This returned vector is multiplied by the step size to approximate the positions and velocities at the next time step in the “RK\_FS19” function.

Part 2.2

See lines 121-146 in “FinalCode\_FS19”.

Part 2.3

See Figures 2, 3, and 4. It can be seen particularly for the first order Runge Kutta method – Figure 2 - that with a large step size, the approximation diverges very quickly from the actual orbit of the satellite. As the step size is decreased, the trajectory does seem to approach a more regular orbit, but even in figure four it significantly diverges from the trajectories given by second and fourth order methods. It is worth noting that as the step size decreases, the axes enclose a smaller range suggesting that the methods ‘close in’ on the actual trajectory as high orders of Runge Kutta are used to approximate the position of the satellite. It can also be seen that second and fourth order methods produce regular orbits at all three step sizes suggesting that they are quite accurate to the actual trajectory of the satellite, especially at the smallest step size where the second and fourth order trajectories practically overlap in figure four.

Part 2.4

See Figures 5, 6, and 7. Similarly to what we found in Part 1, it can be seen that first order Runge Kutta diverges and gets very far from Earth at the largest time step, but approaches more of a regular orbit at the smallest step size in Figure 7. It can be seen in Figure 5 that second and fourth order Runge Kutta methods are at a relatively constant distance away from the Earth in their orbit, but as the time step is reduced it can be seen in Figure 7 that these approximations are actually not at a constant distance but follow more of a cyclic pattern that appears to cycle just over once through with our given data set. The cycle appears to start around t = 0.1\*104 and end around t = 2.9\*104 where two visible minimums are visible in Figure 7. It is also worth noting in Figure 7 that second and fourth order Runge Kutta approximations practically overlap and thus it can be concluded they are good approximations of the actual distance the satellite is from the Earth at a given time.

Part 2.5

See Figures 8 - 13. Though the subplots tend to have the same shape between each of the figures, it is important to note that the magnitude of the y-axis decreased with decreasing step sizes. For example, the relative approximation error for Runge Kutta first order with the large step size over 1000%, but is under 400% for the smallest step size. This trend of decreasing error with decreasing step size can be seen with all three orders of Runge Kutta in all three of these figures. This trend is also observable in figures 11, 12, and 13 for out absolute error plots. It is worth noting that in Figure 10, the Runge Kutta fourth order approximation of the trajectory only has a maximum relative approximation error of about 0.4% from ode45 suggesting that both approximations are very accurate to the actual trajectory of the satellite. This decreasing in y-axis

It was decided to plot both relative approximation error and absolute error so that we can see exactly how far our measurements deviate from the solution – absolute error – while also looking at what magnitude these measurements deviate from the solution – relative approximation error. This was easy as all we had to do was change the way the error was calculated, but the plotting code stayed nearly identical. As labeled, the relative approximation error graphs are in figures 8, 9, and 10, while figures 11, 12, and 13 are for absolute error.

Summary of Appendices

Appendix A: FinalCode\_FS19.m

Appendix B: DrawEarth.m

Appendix C: GaussEliminationFnct\_FS19.m

Appendix D: GoldSecSearchMax\_FS19.m

Appendix E: GoldSecSearchMin\_FS19.m

Appendix F: Ode\_Function\_FS19.m

Appendix G: RK\_FS19.m

Appendix H: Figures 1-13