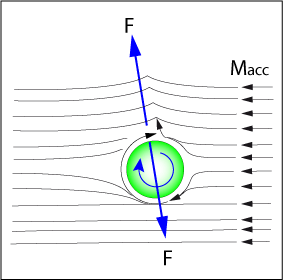
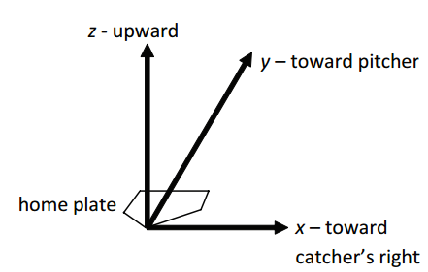
**Introduction**



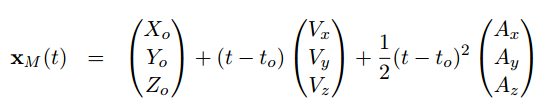
An average major league baseball pitcher can throw up to a hundred miles per hour -- which takes a fraction of a second for the ball to travel 60 feet from the pitcher’s mound to the home plate. One can see that a fast-spinning baseball has a curved trajectory; in baseball, this is commonly known as a *curveball.* In physics, this phenomenon is called the *Magnus effect*. The moving ball is subject to various forces as it travels through the air: gravity, air drag, and Magnus force -- and all these forces are governed by Newton’s laws of motion. The Magnus force can be explained as the force of the air flow acting on the ball, which is perpendicular to the direction of the trajectory; the spinning ball pushes the air down, while the body of air pushes the ball upwards. As a result of the gravitational force compounded with this aerodynamic drag, the ball experiences an exaggerated angular deflection, creating a huge ‘break’ in flight. This paper will investigate the dynamics of curveball pitches modeled using basic physics principles and ordinary differential equations.

**Preliminary Theories**



Several preliminary theories have to be explained before proceeding with the project. Newton’s kinematic equations, the way PITCHf/x tracks the pitches, and the coordinate space are three things that have to be addressed before proceeding to the analysis section of this project. Since this problem happens in real space and time, the coordinate space contains three variables *x, y* and *z* defined with respect to time. This coordinate system as its origin at the back point of the home plate on the ground; “*the x-axis points to the catcher’s right. The y-axis is toward the pitcher. The z-axis is oriented upward.*” Refer to the illustration on the left.

Understanding the PITCHf/x system and how its cameras track the pitches between pitcher and batter is key in understanding the trajectory. The resulting trajectory obtained from the cameras is presented using a nine-parameter (9P) model. The components include constant acceleration, which is represented by Ax, Ay, and Az; the velocity components represented by Vx, Vy, and Vz; and the position components, x(t), y(t), and z(t). Having all these nine-components will help in the plotting of the trajectory.

Below is the basic Newton kinematic equation that we are going to use in Part 1 of Analysis:

**Mathematical Methods**

In real world situations, most scenarios dealing with differential equations cannot be represented in closed forms and thus have no exact solution. In some cases, it is extremely difficult and even impractical to integrate the higher order differential equation to find its solution. However, it is possible to approximate the solutions of such differential equations. Analytical methods can be utilized to find the solution of a series of higher order differential equations. Among these approximation methods are: the Taylor’s, Euler’s, and Runge-Kutta’s methods. Each of these different methods has its own advantages and limitations. In this project, the Runge-Kutta approximation is going to be explored in depth and applied to a multivariate higher order system of differential equations.

There are different ways to design and create a function that runs a Runge-Kutta approximation. Below is a simplified version of the Runge-Kutta function implemented to the differential equation. In a nutshell, the Runge-Kutta is a method of numerically integrating ordinary differential equations using recursion and incrementation at the midpoint of the interval.

|  |
| --- |
| **function** [Output, RK\_tmax] = RK4(f, a, b, N, alpha)  for i = 1:N  K1 = h\*f(t,x);  K2 = h\*f(t+h/2, x+k1/2);  K3 = h\*f(t+h/2, x+k2/2);  K4 = h\*f(t+h, x+k3);  x = x + (k1+2\*k2+2\*k3+k4)/6;  t = t + h;  End |

As mentioned in the previous paragraph, the function displayed is the simplified form of the Runge-Kutta approximation. This form can be modified to accommodate for vectors, which then can be used on multivariate higher order differential equations, like the one we have to solve for the project. It is important to note that a Runge-Kutta approximation works only for first order ODEs, thus a higher-order ODE cannot be directly inserted into the approximation. A conversion from a higher-order ODE to a first order ODE can be used to overcome this difficulty.

**Analysis**

**Part I: Using Newton’s Kinematic Equation to Find Ball Trajectory**

The first part of the analysis uses Newton’s kinematic equation of displacement, represented in vector form. This section assumes the acceleration to be constant. The first vector (x0, y0, z0) is the initial position of the ball. The second vector (Vx, Vy, Vz) is the initial velocity, and the third vector (Ax, Ay, Az) is the acceleration which is constant. By plugging in the necessary values to the parameters to the above Newton’s kinematic equation, the trajectory of the baseball can be projected in a 3D plot. The first equation does not take account the air drag and gravity. But it can be represented quite well, without the need to use differential equations. This first method is quite useful to find a rough approximation of the trajectory of the ball. Below is the data obtained from plugging in the values in the first part:

(a) **Pitch 1**

**Initial conditions:** X0=-2.509;Y0=50; Z0=5.928; Vx=9.182; Vy=-132.785; Vz=-10.697; Ax = -19.268; Ay = 30.713; Az = -16.580;

|  |
| --- |
| Newton’s Kinematics Formula Result:  t x y z  0 -2.5090 50.0000 5.9280  0.0004 -2.5055 49.9492 5.9238  0.0008 -2.5020 49.8983 5.9196  ...  0.3813 -0.4086 1.6024 0.5411  0.3817 -0.4079 1.5560 0.5345  0.3828 -0.4058 1.4170 0.5146 |

(b) **Pitch 2**

**Initial conditions:** X0=-2.43;Y0=50; Z0=6.46; Vx=9.46; Vy=-143.17; Vz=-9.15; Ax = -23.08; Ay = 34.2; Az = -26.09;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Newton’s Kinematics Formula Result =**   |  |  |  |  | | --- | --- | --- | --- | | **t** | **x** | **y** | **z** | | 0 | -2.43 | 50 | 6.46 | | 0.0001 | -2.42905 | 49.98568 | 6.459085 | | 0.0002 | -2.42811 | 49.97137 | 6.458169 | | 0.0003 | -2.42716 | 49.95705 | 6.457254 | | ... | | | | | 0.3541 | -0.52718 | 1.447617 | 1.584314 | | 0.3542 | -0.52705 | 1.434512 | 1.582475 | | 0.3543 | -0.52692 | 1.421406 | 1.580636 | |

**Part II: Fourth-Order Runge-Kutta Analysis of Ball Trajectory based on Hall-Nathan Method**

This second analysis part, which is where the meat of the project lies, deals with using Nathan’s equation three and applying this equation to the Runge-Kutta 4 function. Since Nathan’s equation 3 is a second order multivariate differential equation, it is not possible to directly insert this ODE into the Runge-Kutta function. Thus, a conversion of this second order ODE to a first order system is necessary. The conversion is shown below:

|  |
| --- |
| Higher Order ODE to First Order Conversion  X = x(1)  Y = x(2)  Z = x(3)  X’= x(4)  Y’= x(5)  Z’= x(6)  X’’ = -K\*Cd\*V(x(4), x(5), x(6))\*x(4) - K\*CL\*V(x(4), x(5), x(6))\*x(5)\*sin(phi\_s);  Y’’ = -K\*Cd\*V(x(4), x(5), x(6))\*x(5) - K\*CL\*V(x(4), x(5), x(6))\*(x(4)\*sin(phi\_s - x(6)\*cos(phi\_s)));  Z’’ = -K\*Cd\*V(x(4), x(5), x(6))\*x(6) + K\*CL\*V(x(4), x(5), x(6))\*x(5)\*cos(phi\_s) - g]; |

After this conversion is performed, the system will be in first order form, which will be able to be inserted into the Runge-Kutta approximation. In order to check if the Runge-Kutta 4 (RK4) is fully functional, a check between the RK4 and the ODE45 would be required. After running the Runge Kutta and the ODE45 using the same constraints, we are able to obtain identical values.

**Compare RK4 method with built-in differential equations solver from MATLAB:**

|  |
| --- |
| RK\_Output =  t x y z Vx Vy Vz  0 -2.5090 50.0000 5.9280 9.1820 -132.7850 -10.9670  0.0004 -2.5053 49.9469 5.9236 9.1733 -132.7701 -10.9735  0.0008 -2.5017 49.8938 5.9192 9.1645 -132.7551 -10.9799  ...  0.3292 -0.5769 8.1399 1.4229 2.8161 -121.9435 -16.4501  0.3296 -0.5757 8.0911 1.4164 2.8093 -121.9316 -16.4569 |

|  |
| --- |
| ODE45\_Output =  t x y z Vx Vy Vz  0 -2.5090 50.0000 5.9280 9.1820 -132.7850 -10.9670  0.0004 -2.5053 49.9469 5.9236 9.1733 -132.7701 -10.9735  0.0008 -2.5017 49.8938 5.9192 9.1645 -132.7551 -10.9799  ...  0.3292 -0.5769 8.1399 1.4229 2.8161 -121.9435 -16.4501  0.3296 -0.5757 8.0911 1.4164 2.8093 -121.9316 -16.4569 |

Once we have made sure that the RK4 is fully functional, we are able to perform the analysis on the different trajectories.

(a) **Pitch 1**

**Initial conditions:** *X0=-2.509;Y0=50; Z0=5.928; Vx=9.182; Vy=-132.785; Vz=-10.697; Ax = -19.268; Ay = 30.713; Az = -16.580;*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RK\_Output =   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **t** | **x** | **y** | **z** | **Vx** | **Vy** | **Vz** | | 0 | -2.509 | 50 | 5.928 | 9.182 | -132.785 | -10.967 | | 0.0001 | -2.50808 | 49.98672 | 5.926903 | 9.179814 | -132.781 | -10.9686 | | 0.0002 | -2.50716 | 49.97344 | 5.925806 | 9.177628 | -132.778 | -10.9702 | | 0.0003 | -2.50625 | 49.96017 | 5.924709 | 9.175442 | -132.774 | -10.9718 | | 0.0004 | -2.50533 | 49.94689 | 5.923612 | 9.173256 | -132.77 | -10.9735 | |  |  |  | ... |  |  |  | | 0.3845 | -0.44694 | 1.441928 | 0.487142 | 1.888522 | -120.303 | -17.3953 | | 0.3846 | -0.44675 | 1.429898 | 0.485403 | 1.886878 | -120.3 | -17.397 | | 0.3847 | -0.44656 | 1.417868 | 0.483663 | 1.885233 | -120.297 | -17.3987 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ODE45 Output =   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **t** | **x** | **y** | **z** | **Vx** | **Vy** | **Vz** | | 0 | -2.509 | 50 | 5.928 | 9.182 | -132.785 | -10.967 | | 0.00962 | -2.42168 | 48.72433 | 5.821749 | 8.972487 | -132.427 | -11.1227 | | 0.01924 | -2.33636 | 47.45209 | 5.713998 | 8.764571 | -132.072 | -11.2787 | | 0.02886 | -2.25304 | 46.18325 | 5.604745 | 8.558234 | -131.72 | -11.4351 | | 0.03848 | -2.1717 | 44.91779 | 5.493986 | 8.353455 | -131.371 | -11.5918 | |  |  |  | ... |  |  |  | | 0.36556 | -0.48567 | 3.725747 | 0.813536 | 2.202143 | -120.861 | -17.0709 | | 0.37518 | -0.46526 | 2.564428 | 0.648522 | 2.042325 | -120.577 | -17.2356 | | 0.3848 | -0.44638 | 1.405839 | 0.481923 | 1.883588 | -120.294 | -17.4004 | |

(b) **Pitch 2**

**Initial conditions:** X0=-2.43;Y0=50; Z0=6.46; Vx=9.46; Vy=-143.17; Vz=-9.15; Ax = -23.08; Ay = 34.2; Az = -26.09;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RK\_Output =   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **t** | **x** | **y** | **z** | **Vx** | **Vy** | **Vz** | | 0 | -2.43 | 50 | 6.46 | 9.46 | -143.17 | -9.15 | | 0.0001 | -2.42905 | 49.98568 | 6.459085 | 9.457396 | -143.166 | -9.15269 | | 0.0002 | -2.42811 | 49.97137 | 6.458169 | 9.454792 | -143.162 | -9.15537 | | 0.0003 | -2.42716 | 49.95705 | 6.457254 | 9.452189 | -143.158 | -9.15806 | | 0 | -2.43 | 50 | 6.46 | 9.46 | -143.17 | -9.15 | |  |  |  | ... |  |  |  | | 0.3552 | -0.57413 | 1.431658 | 1.551383 | 1.344232 | -130.705 | -18.3932 | | 0.3553 | -0.57399 | 1.418588 | 1.549543 | 1.342229 | -130.702 | -18.3957 | | 0.3554 | -0.57386 | 1.405518 | 1.547703 | 1.340226 | -130.698 | -18.3982 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ODE45\_Output =   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **t** | **x** | **y** | **z** | **Vx** | **Vy** | **Vz** | | 0 | -2.43 | 50 | 6.46 | 9.46 | -143.17 | -9.15 | | 0.008893 | -2.34697 | 48.72947 | 6.377643 | 9.229439 | -142.825 | -9.38841 | | 0.017785 | -2.26599 | 47.462 | 6.293169 | 9.000472 | -142.482 | -9.62642 | | 0.026678 | -2.18703 | 46.19756 | 6.206583 | 8.773084 | -142.141 | -9.86404 | | ... | | | | | | | | 0.337915 | -0.60085 | 3.733105 | 1.870651 | 1.69832 | -131.27 | -17.9491 | | 0.346808 | -0.58656 | 2.568041 | 1.710175 | 1.518713 | -130.984 | -18.1738 | | 0.3557 | -0.57386 | 1.405518 | 1.547703 | 1.340226 | -130.698 | -18.3982 | |

**Part III: Comparison between Newton’s Kinematics Formula vs. Hall-Nathan Method**

After describing part 1 and part 2 and obtaining all the required values, we are able to compare the different pitch trajectories using the two different methods. We are going to use three methods to compare the two different trajectories: Plotting the two pitches on a same 3D plot to visualize the flight path of the trajectories, comparing the trajectories on each of the planes (x, y, z), plotting the absolute error between the three components, and the ratio between the results obtained from part 1 and part 2.

**Pitch 1**

|  |
| --- |
|  |

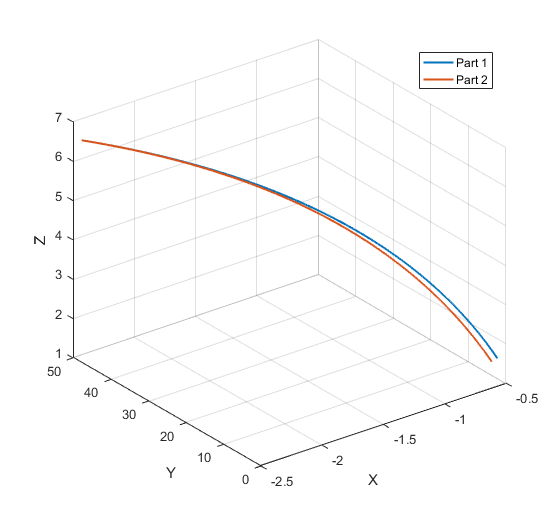
|  |  |  |
| --- | --- | --- |
| **X-Plane** | **Y-Plane** | **Z-Plane** |

|  |  |
| --- | --- |
| **Difference** | **Ratio** |

As shown above, the error between the two pitches in all three components is very minute. It may seem like the Y error is really big compared to the X and Z errors in the difference graph, however, we must take into account that the Y coordinate starts from Y=50 and ends at Y=1.417, whereas the X and Z starts and ends at lower numbers. In other words, the huge “error” in Y is caused by a scaling problem.

We have also included the ratio plot in order to visualize the deviation between the values obtained from part 1 and part 2. From the plot, we can see that the X component only deviates dramatically towards the end of the trajectory, which is what we see in the X-plane comparison plot.

**Pitch 2**



|  |  |  |
| --- | --- | --- |
| X-plane | Y-Plane | Z-Plane |

|  |  |
| --- | --- |
| Difference | Ratio |

Notice that both pitches deviate similarly -- notice that the Y-component absolute error is huge, this is probably due to the Magnus effect that is not accounted in the first part (using Newton’s kinematic formula). The error is relatively big because the range of the Y-component is also relatively huge compared to the other components (it ranges from 0 to 50). That is why we decided to compare the two using ratio. One can see that the ratio is really close to 1 at the beginning (when time is near 0), but as the time progresses, we can see it deviates -- the ratio of Y and Z goes below 1, while the X ratio goes above 1. This is true for both the first and second pitches. Although the ratio for the second pitch has a relatively closer to 1 (within the range of 0.9 to 1.1), while for the first pitch the ratio ranges from (0.7 to 1.5).

|  |  |  |
| --- | --- | --- |
|  | **Pitch 1 Final Velocity** | **Pitch 2 Final Velocity** |
| **Part 1** | 122.275 | 132.3452 |
| **Part 2** | 121.5578 | 131.4442 |
| **Error** | 0.71715 | 0.901007 |

The difference between the two trajectories at their final position for the respective pitches:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pitch 1*** | **x** | **y** | **z** |
| **Part 1** | -0.4058 | 1.417 | 0.5146 |
| **Part 2** | -0.4466 | 1.41787 | 0.48366 |
| **Difference** | 0.0408 | 0.000868 | 0.030937 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pitch 2*** | **x** | **y** | **z** |
| **Part 1** | -0.5269 | 1.42141 | 1.58064 |
| **Part 2** | -0.5737 | 1.39245 | 1.54586 |
| **Difference** | 0.0468 | 0.028958 | 0.028958 |

From seeing the difference for the two pitches above, we can clearly see that the absolute difference between the values obtained from part 1 and part 2 is very minute; this means that the end positions that resulted from part 1 and part 2 is very similar.

**Conclusion**

After performing the whole analysis, it is clear that the values obtained from the Hall-Nathan method is actually similar to the ones obtained using Newton’s kinematic equation -- the difference accounted in the second method is due to the effect of gravity, air resistance, and the Magnus force. To approximate

**Appendix**

|  |
| --- |
| **MAIN FUNCTION**  %CONSTANTS  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %PITCH 1  % Cd = 0.3926485; CL = 0.255819;  % phi\_s = 236.0038\*pi/180; phi\_m = 146;  % theta = 3.9; spin = 2388;  % g = 32.2; K = 0.005152949;  % x0 = -2.509; y0 = 50; z0 = 5.928;  % Vx0 = 9.182; Vy0 = -132.785; Vz0 = -10.967;  % Ax = -19.268; Ay = 30.713; Az = -16.580;  %PITCH 2  Cd = 0.3512265; CL = 0.216346;  phi\_s = 4.591151161;  spin = 2419;  g = 32.174; K = 0.005316103;  x0 = -2.43; y0 = 50; z0 = 6.46;  Vx0 = 9.46; Vy0 = -143.17; Vz0 = -9.15;  Ax = -23.08; Ay = 34.2; Az = -26.09;  %PART ONE  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  syms t  Xx = @(t) x0 + t\*Vx0 + 0.5\*Ax\*t.^2;  Xy = @(t) y0 + t\*Vy0 + 0.5\*Ay\*t.^2;  Xz = @(t) z0 + t\*Vz0 + 0.5\*Az\*t.^2;  % To find the tmax for part 1  eqn = Xy(t) == 1.417;  tmax = solve(eqn, t);  tmax = tmax(1);  time = 0:1/10000:tmax;  Table = double([time.' Xx(time).' Xy(time).' Xz(time).']);  %PART TWO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Declarations  V = @(x, y, z) sqrt(x.^2 + y.^2 + z.^2);  f = @(t, x)[x(4); x(5); x(6);  -K\*Cd\*V(x(4), x(5), x(6))\*x(4) - K\*CL\*V(x(4), x(5), x(6))\*x(5)\*sin(phi\_s);  -K\*Cd\*V(x(4), x(5), x(6))\*x(5) - K\*CL\*V(x(4), x(5), x(6))\*(x(4)\*sin(phi\_s - x(6)\*cos(phi\_s)));  -K\*Cd\*V(x(4), x(5), x(6))\*x(6) + K\*CL\*V(x(4), x(5), x(6))\*x(5)\*cos(phi\_s) - g];  alpha = [x0; y0; z0; Vx0; Vy0; Vz0];  % Run the RK and the RK Output  [RK\_Output, RK\_tmax]=RK4(f, 0, 1/10000, alpha);  disp(RK\_tmax);  % Run the ODE45 and the Output  [t, ODE45\_Output] = ode45(f, [0 RK\_tmax], alpha);  % OUTPUT FROM ABOVE ARE IDENTICAL  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % PLOT BOTH IN SAME GRAPH  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  PT1 = plot3(Xx(time), Xy(time), Xz(time));  hold on  grid on  RKX = RK\_Output(:, 2);  RKY = RK\_Output(:, 3);  RKZ = RK\_Output(:, 4);  PT2 = plot3(RKX, RKY, RKZ);  set(PT1, 'LineWidth', 1.5);  set(PT2, 'LineWidth', 1.5);  hold off  legend('Part 1', 'Part 2')  % %X GRAPH  % hold on  % t2 = RK\_Output(:, 1);  % P2X = plot(t2, RKX);  % P1X = plot(time, Xx(time));  % legend('Part 1', 'Part 2')  % xlabel('Time (s)');  % ylabel('X');  %  % %Y GRAPH  % hold on  % t2 = RK\_Output(:, 1);  % P2X = plot(t2, RKY);  % P1X = plot(time, Xy(time));  % legend('Part 1', 'Part 2')  % xlabel('Time (s)');  % ylabel('Y');  %  % %Z GRAPH  % hold on  % t2 = RK\_Output(:, 1);  % P2X = plot(t2, RKZ);  % P1X = plot(time, Xz(time));  % legend('Part 1', 'Part 2');  % xlabel('Time (s)');  % ylabel('Z');  % MAX VELOCITY  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  H =1/10000;  r\_1 = size(Table,1);  r\_2 = size(RK\_Output,1);  VF\_PT2 = vpa(V(RK\_Output(r\_2, 5), RK\_Output(r\_2, 6), RK\_Output(r\_2, 7)));  Vfx = (Table(r\_1, 2) - Table(r\_1 - 1, 2))/H;  Vfy = (Table(r\_1, 3) - Table(r\_1 - 1, 3))/H;  Vfz = (Table(r\_1, 4) - Table(r\_1 - 1, 4))/H;  VF\_PT1 = vpa(V(Vfx, Vfy, Vfz));  % MAX TIME  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  TM\_PT1 = vpa(tmax);  TM\_PT2 = RK\_tmax;  % ERROR GRAPH  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % ErrorX = []; ErrorY = []; ErrorZ = [];  %  % for i = 1:size(Table,1)  % ErrorX = [ErrorX abs(Table(i, 2)-RK\_Output(i, 2))];  % ErrorY = [ErrorY abs(Table(i, 3)-RK\_Output(i, 3))];  % ErrorZ = [ErrorZ abs(Table(i, 4)-RK\_Output(i, 4))];  % end  %  % t = 0:1/10000:TM\_PT1;  % hold on  % plot(t, ErrorX);  % plot(t, ErrorY);  % plot(t, ErrorZ);  % hold off  % legend('Error X', 'Error Y', 'Error Z');  % xlabel('Time (s)');  % ylabel('Difference');  % axis([0, 0.4, 0, 0.25]) |
| **RK4 Function**  function [Output, RK\_tmax] = RK4(f, a, h, alpha)  %Constants  % Cd = 0.3926485; CL = 0.255819;  % phi\_s = 236.0038\*pi/180; phi\_m = 146;  % theta = 3.9; spin = 2388;  % g = 32.2; K = 0.005152949;  % x0 = -2.509; y0 = 50; z0 = 5.928;  % Vx0 = 9.182; Vy0 = -132.785; Vz0 = -10.967;  % Ax = -19.268; Ay = 30.713; Az = -16.580;  Cd = 0.3512265; CL = 0.216346;  phi\_s = 4.591151161;  spin = 2419;  g = 32.174; K = 0.005316103;  x0 = -2.43; y0 = 50; z0 = 6.46;  Vx0 = 9.46; Vy0 = -143.17; Vz0 = -9.15;  Ax = -23.08; Ay = 34.2; Az = -26.09;  % Conditions/Variables  V = @(x, y, z) sqrt(x.^2 + y.^2 + z.^2);  f = @(t, x)[x(4); x(5); x(6);  -K\*Cd\*V(x(4), x(5), x(6))\*x(4) - K\*CL\*V(x(4), x(5), x(6))\*x(5)\*sin(phi\_s);  -K\*Cd\*V(x(4), x(5), x(6))\*x(5) - K\*CL\*V(x(4), x(5), x(6))\*(x(4)\*sin(phi\_s - x(6)\*cos(phi\_s)));  -K\*Cd\*V(x(4), x(5), x(6))\*x(6) + K\*CL\*V(x(4), x(5), x(6))\*x(5)\*cos(phi\_s) - g];  m = size(alpha,1);  if m == 1  alpha = alpha';  end  %h = (b-a)/N; %the step size  t(1) = a;  w(:,1) = alpha; %initial conditions  for i = 1:300000  k1 = h\*f(t(i), w(:,i));  k2 = h\*f(t(i)+h/2, w(:,i)+0.5\*k1);  k3 = h\*f(t(i)+h/2, w(:,i)+0.5\*k2);  k4 = h\*f(t(i)+h, w(:,i)+k3);  w(:,i+1) = w(:,i) + (k1 + 2\*k2 + 2\*k3 + k4)/6;  t(i+1) = a + i\*h;    if (w(2,i)<1.417)  RK\_tmax=t(i);  break;  end    end  Output = [t' w']; |