ASEN 2003 Lab 1: Roller Coaster Design February 2, 2017

Kayla Gehring [*]
Pierre Guillaud [†]
Nicholas Renninger [‡]
University of Colorado - Boulder

The purpose of this lab was to design a roller coaster in order to study the dynamics of a particle over multiple coordinate systems and subject to different accelerations depending on its position on the track. In particular, speed and G loading were studied as a function of arc length along the track. The roller coaster contains a Zero-G drop, a helix, a parabolic hill, a loop, and finally a semicircle before reaching the braking section. The roller coaster reaches a maximum speed of 49.5 m/s, maximum lateral G's of 2.83, and maximum vertical G's of 5.73. The roller coaster runs for a total of 27.1 seconds.

Nomenclature

G	=	G-Force
g	=	Gravitational Acceleration $[m/s^2]$
G_L	=	G-force in the lateral direction of the rider
G_T	=	G-force in the tangential direction of the rider
G_V	=	G-force in the vertical direction of the rider
h_o	=	initial height of coaster $[m]$
L	=	Lateral Force [N]
m	=	Mass of Object [kg]
N	=	Normal Force [N]
s	=	arc length $[m]$
v	=	speed $[m/s]$

[*]105677160
[†]104426060
[‡]105492876

Contents

I. Introduction

In this report, we outline the design of a roller coaster and study its various properties, such as velocity, position, and acceleration. The dynamics of the various elements of the track are to be analyzed in order to better design them and to meet the various requirements and limitations for the lab. The results obtained for each individual roller coaster section will then be combined to study the overall performance of the roller coaster.

A few key assumptions were made in this lab. First, we assumed no friction or drag forces caused by the track or the surrounding environment, until we reach the last element - a braking section - which requires frictional forces to slow down the cart. Other assumptions made are that the roller coaster cart and its passengers are a single point mass. Therefore, all forces have been calculated as if they act on the same point. Another major assumption is that the cart always remains on the track, creating orthogonal forces when necessary.

The requirements for the roller coaster are as follows:[1] the track is limited to a length of 1250m and must end at ground level ($h_f = 0$m), the maximum accelerations must be below 5 G for the forward component, 4 G for the backward component, 6 G for the vertical component, 1 G for the negative vertical component, and below 3 G for the lateral components of the acceleration. There was also a requirement to have a minimum of 3 distinct elements, a section of Zero G loading, as well as at least one banked turn. Other limitations are the initial height of the track ($125\,m$), and minimum height is $0\,m$ (it cannot go below ground level).

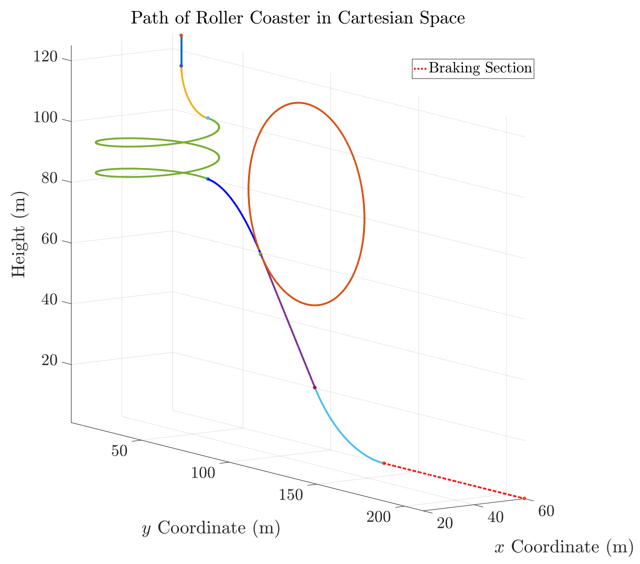

Figure 1: Path of the Roller Coaster. *Each new color represents a different segment of the entire track. The braking segment is shown with a dotted, red line. Each segment was analyzed individually in the following sections.*

Our design process consisted of first analyzing the various requirements and studying how they could be met, followed by finding the parameterization of all the track elements chosen for the roller coaster. We chose a straight drop for our zero G section, and a helix, loop, and parabolic hill as our track elements. Using the equations from parameterization allowed us to calculate the accelerations for each segment, and to finally take those values to adjust and recalculate our overall roller coaster elements where needed to ensure we stayed within the design constraints imposed by the maximum allowable G forces.

II. Design

In order to analyze the performance of the roller coaster, we will first describe each track element in detail. Below, you will find a description, mathematical specification, sketch of each track main track element, and each transition between main elements. The critical points of each main element, and each transition if applicable, are analytically described. Additionally, expressions for the acceleration and G loading of the train are provided.

II.A. Zero G Drop

II.A.1. General Description

The first section of this roller coaster is a straight drop. The roller coaster car is held in place by the track, but since we have made the assumption of no friction, since we are treating the roller coaster car as a point mass, and since the car starts from rest, there is no force acting on the car except gravity.

The section can be described mathematically by the equation: $\mathbf{r}(t) = [x_i\,\hat{\mathbf{i}},\ y_i\,\hat{\mathbf{j}},\ (-t+z_i)\,\hat{\mathbf{k}}]$ meters, where t is an arbitrary parameter not necessarily related to time, and x_i, y_i, and z_i describe the position (x, y, z) of the car as it enters the section. This Zero G section is defined over $0 \leq t \leq 10$, and is therefore 10m long. The general formula for arc length, Eq. 1, is shown below. For this track section, it is $s = \int_0^{10} -1dt$.

$$s = \int_{t_f}^{t_i} \|\boldsymbol{r}'(t)\|dt \tag{1}$$

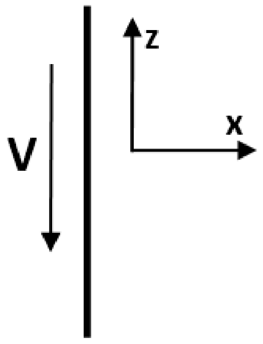

Figure 2: ZeroG section of the track

II.A.2. Force, Acceleration, and G loading

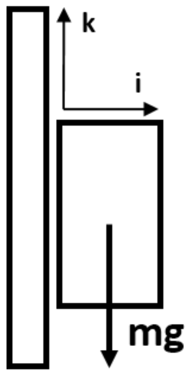

Figure 3: Free Body Diagram of the Zero-G section

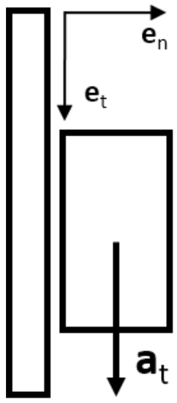

Figure 4: Acceleration on the zero-g section of the track

This track element has no notable critical points. That is, there is no point at which the derivative of $\mathbf{r}(t)$ is zero except at $t = 0$. Since this is just a straight drop, the free body diagram (FBD) and acceleration look the same through the entire section, shown in figure 3.

The G load can be determined using the equation below, Eq. (2). Unsurprisingly, the G loading over this section is zero, since there is no resistance (normal) force on the car. There is only the force of gravity, which is by definition Zero G.

$$G = N/g \tag{2}$$

II.B. First Transition, Banked

II.B.1. General Description

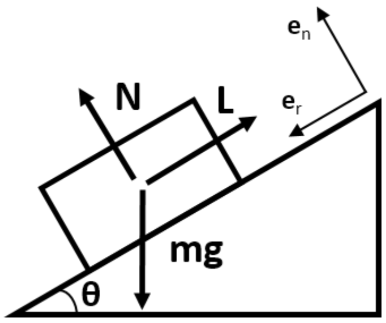

Figure 5: Free Body Diagram of a banked turn sections

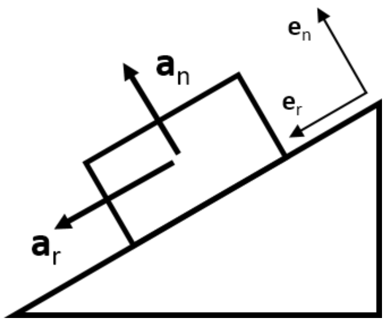

Figure 6: Acceleration on banked turn sections

There is a short transition between the Zero G section of the track and the Helix. The main feature of this transition is that the car rolls gradually out of its starting plane into a bank. This means that the roller coaster rider would experience a normal force due to the car. This transition rolls from a $\theta = 0\,\text{deg}$ to $\theta = 70°$ bank angle linearly in t (and thus it rolls slightly faster as s advances) over the course of the whole segment.

This transition can be described by the equation $\mathbf{r}(t) = [(-Rsin(t) + x_i + R)\hat{\mathbf{i}},\ (y_i)\hat{\mathbf{j}},\ (Rcos(t) + z_i)\hat{\mathbf{k}}]$ meters, where $R = 15\,m$ is the radius of the curved section. It is defined over $\pi/2 \leq t \leq 3.156$, which creates an arc length of 23.8m.

II.B.2. Force, Acceleration, and G loading

This track element has one critical point in the intrinsic coordinate frame. It is at $t_1 = \pi$. By taking the derivative of s, we can see at t_1 that the velocity is at a maximum, which makes sense given that this is the lowest point on the circle. All acceleration on this track is due to potential energy transferring to kinetic energy as height reduces. Therefore, the greatest velocity on any track section is at its lowest point. By taking the derivative of the velocity, or the second derivative of s, we can find that acceleration at this critical point is at a maximum, "pushing" the cart upwards. Note that this acceleration is in relation to the circle only and not the banking or acceleration due to gravity.

The force and acceleration also vary with θ throughout this segment. Specifically, the forces are weight mg, normal N, and lateral L forces. The lateral force is due to the track, which holds the car in place, and can be found by solving $F_y = a_y * m = L_y - N_y - cos(\theta)mg$. This is difficult to describe numerically along the bank, but was designed to keep the lateral G loading on the passenger low. To fully describe N, the forces in the x and z axes would also have to be solved, where $F_x = a_x * m = L_x - N_x - sin(\theta)mg$ and $F_z = a_z * m = N_z - mg$. Again, this section is difficult to characterize analytically, but there is acceleration in all three axes as the track goes through this circle and changes position on all three axes.

The G forces are described by the equations below, where N is the G loading normal to the passenger, and T is the tangential loading. In this case, the passenger would feel a force upwards through the seat, rather than downwards through the bar. Laterally, they would feel the lateral force on their right side as the travel in the positive x direction.

$$G_T = L/mg \tag{3}$$

$$G_N = N/mg \tag{4}$$

II.C. Helix

II.C.1. General Description

The Helix section of the track is essentially a long, continuous banked turn until it reaches the second transition element of the track, which will take care of gradually rolling out of the banked turn angle. For this reason, the forces acting on the roller coaster cart are the same as seen above in the previous section, and the free body diagrams are identical.

The radius R and the height chosen are both equal to 30 meters in order to stay within the G-force limits. The banked turn angle chosen is $\theta = 70°$, which can also be seen by the end of the previous track segment.

This section can be described by the equation $\mathbf{r}(t) = [(Rsin(2t - t_i + x_i))\hat{\mathbf{i}}, \, (Rcos(2t - t_i + y_i - R))\hat{\mathbf{j}}, (-30 * (t - t_i)/(2\pi) + z_i)\hat{\mathbf{k}}]$, and is defined over $0 \leq t \leq 2 * \pi$. The arc length then is 252.19 m, going through 2 total loops.

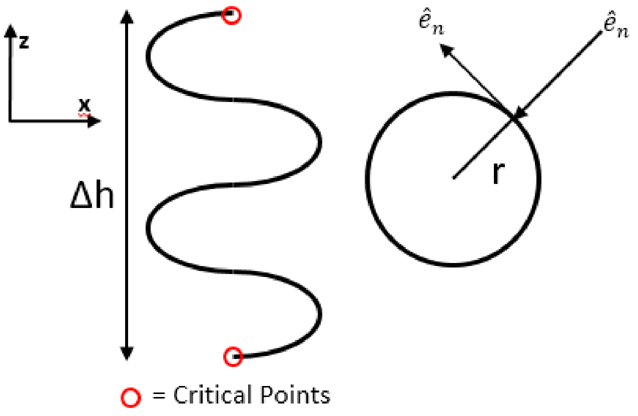

Figure 7: Plot of the Helix over the x-z plane and the tangential and normal plane

II.C.2. Force, Acceleration, and G loading

This track element has two critical points, one at the beginning of the helix and one at the end point. The acceleration depends on the velocity of the cart on the track, the first critical point represents the lowest acceleration the point mass will experience, while the final critical point represents the point at which the velocity is the highest, and therefore the point at which the cart will experience the highest acceleration during this section of the track. The critical points are shown in Fig. 7.

The equations used to calculate the G loads are as follow:

$$G_N = v^2/(rg) + sin(\theta) \tag{5}$$

$$G_R = cos(\theta) + a_b/g \tag{6}$$

Where G_N is the G-force in the e_N direction and G_R is the G-force in the e_R direction. The whole helix is a continuous banked turn, and its Free Body Diagram will therefore be the same as of the banked turn sections, Figs. 5 and 6.

II.D. Parabolic Hill

II.D.1. General Description

This section is a parabolic hill, changing in the x and z axes. It acts as a transition between the helix and the loop. Similar to the first transition, it slowly rotates from a banked angle into an angle that is parallel to the ground. That is, it rolls from $70° \geq \theta \geq 0°$.

This section can be described by the equation $\mathbf{r}(t) = [(t + x_i)\hat{\mathbf{i}}, \, (y_i)\hat{\mathbf{j}}, \, (-\frac{1}{50}(t - (25 * \hat{\mathbf{T}}_{z,prev})^2) + z_i)\hat{\mathbf{k}}]$. $\hat{\mathbf{T}}_{z,prev}$ is the z-component of the tangent vector from the previous section at $t_{f,helix}$, which ensures a smooth track transition from the helix. This section is defined over $0 \leq t \leq 30$, creating a total arc length of 37.3m. Note: Note once again that the parameter t does not necessarily describe time, and t does not carry between equations for track sections.

II.D.2. Force, Acceleration, and G loading

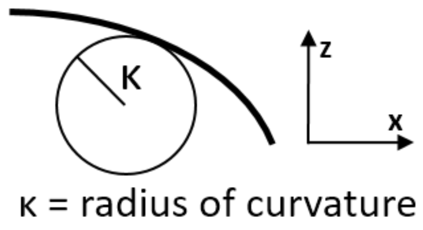

Figure 8: Parabola section of the track

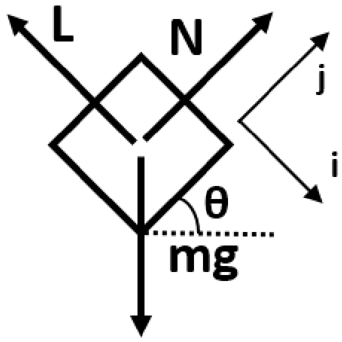

Figure 9: Free Body Diagram of the cart on the parabolic hill during banked turn adjustment

This section does not have a critical point over its path, though the equation does have one critical point where it reaches it maximum height. Solving for $\frac{dr}{dt} = 0$ gives the maximum at $t = -1.98$. At this theoretical point, the velocity is at a minimum over the section. The acceleration, however, is constant throughout the entire section at $a_z = \frac{1}{25}t\ \frac{m}{s^2}$. In the x and y axes, the FBD looks nearly identical to the one in Fig. 5, just with θ being measured from the opposite direction. Its G loading and acceleration are also very similar to those of the first transition and can be described with the same equations.

II.E. Loop

II.E.1. General Description

The loop section of the track is a circular path of defined radius where the cart begins and ends at the same location and at the same slope. Calculating the length of this section is relatively simple due to it's very simple geometry. All the equations used will depend on its intrinsic tangential and normal frame relative to the inertial Cartesian frame. The initial position of the loop is not necessarily at the bottom of the loop, and is represented by s_0. The angle θ depends entirely on the position of the initial position. The track can be described by $\mathbf{r}(t) = [(-R * sin(t) + x_i + Rsin(t_i))\hat{\mathbf{i}}, (y_i)\hat{\mathbf{j}}, (R * cos(t) + z_i - R * cos(t_i))\hat{\mathbf{k}}]$, and is defined over $2.23 \leq t \leq 2.23 + 2\pi)$, giving an arc length of $207\,m$.

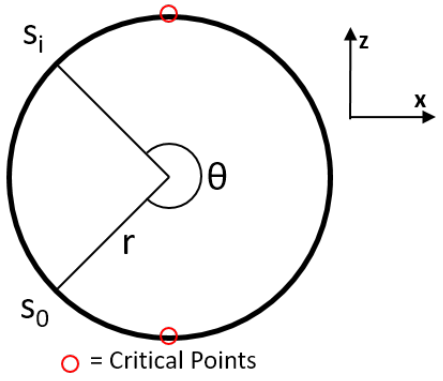

Figure 10: Loop section of the track

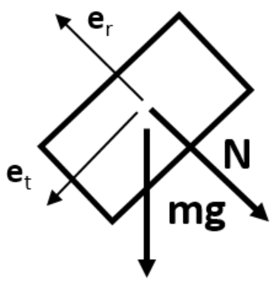

Figure 11: Free Body Diagram of the loop

II.E.2. *Force, Acceleration, and G loading*

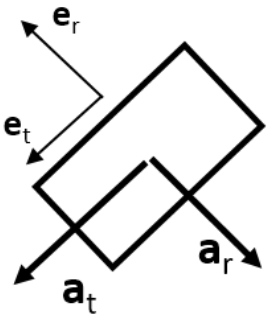

Figure 12: Acceleration of the cart on the loop

This track element has two critical points, at it's highest height, and therefore it's lowest velocity, and at it's lowest height, therefore at it's highest velocity. This is shown in the equation below where the acceleration is proportional to the square of the velocity. Those critical points can be seen on the sketch of the loop above.

$$G_T = 0 \tag{7}$$

$$G_R = v^2/(g * r) - sin(\theta) \tag{8}$$

Where G_N and G_T are the G-forces in both directions indicated on the free body diagram.

II.F. Second Transition - Out of Loop

II.F.1. *General Description*

This is a transition out of the loop. This transition is just a straight line matching the final slope of the loop. While this transition could have been skipped, for all practical purposes, the roller coaster looks better as a result of having an extra straight-line transition section. This section can be described as $\mathbf{r}(t) = [(T_{x,\hat{prev}} t + x_i)\hat{\mathbf{i}}, /, (y_i)\hat{\mathbf{j}}, (\hat{\mathbf{T}}_{z,prev} + z_i)\hat{\mathbf{k}}]\, m$. Again, the $\hat{\mathbf{T}}_{prev}$ components allow a smooth transition for the car by ensuring the slope is identical to the final slope as the car exits the loop. This transition is defined over $0 \leq t \leq 38.748$, making an arc length of $48.748\, m$.

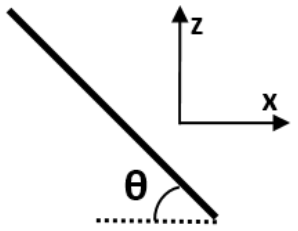

Figure 13: Sketch of the second transition

II.F.2. *Force, Acceleration, and G loading*

This track element has no critical points. The FBD is extremely basic, consisting only of a normal force, lateral force and weight. The acceleration is only downward, and the G forces are only upwards. While it would seem that there should be an x component to the normal force, there is no friction to create a tangential reaction force. Solving $F = -ma = N - mg$ is all that's needed to find the G loading over this section.

$$G_n = (g - a/, sin(\theta))/g \tag{9}$$

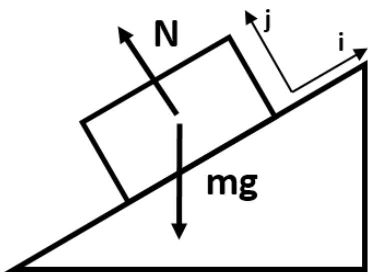

Figure 14: Free Body Diagram of the second transition

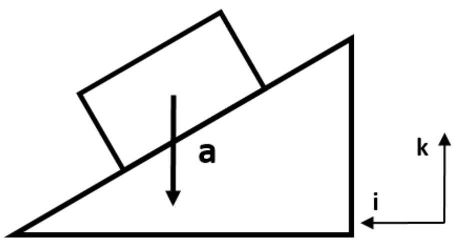

Figure 15: Acceleration of the second transition

II.G. Third Transition - Into Braking Section

II.G.1. General Description

This is the final transition and exists to bring the car to the ground before braking. It is a segment of a circle with radius $R = 55m$, and is defined over $2.23 \leq t \leq \pi$. It is described by $\mathbf{r}(t) = [(-Rsin(t) + x_i + R * sin(t_i))\hat{\mathbf{i}},\ (y_i)\hat{\mathbf{j}},\ (Rcos(t) + z_i - R * cos(t_f))\hat{\mathbf{k}}]$, and has an arc length of $49.9\,m$.

II.G.2. Force, Acceleration, and G loading

This is just another circle segment, so it can be described the same way as the loop. There is one critical point at $t = \frac{3}{4}\pi, \pi$. The maximum acceleration due to the track at $t = \pi$ is upwards. There is of course also still a force due to gravity, and this segment reaches a final height of zero, so the velocity reaches its absolute maximum at $t = \pi$.

$$G_t = 0 \tag{10}$$

$$G_R = v^2/(g * r) - sin(\theta) \tag{11}$$

II.H. Braking Section

II.H.1. General Description

The braking section of the track is a flat segment at zero height and at the end of which the roller coaster ends. Its primary use is to create friction on the cart in order to slow it down to a stop at the exact point at which the track ends. Its length depends entirely on the velocity of the cart at the beginning of the segment, due to the limitations in terms of the backward acceleration the cart may feel. This part of the track will not have any change in acceleration in the normal or tangential directions, and will not have any acceleration lateral directions due to it's shape. The section can be described by $\mathbf{r}(t) = [(t + x_i)\hat{\mathbf{i}},\ (y_i)\hat{\mathbf{j}},\ (z_i)\hat{\mathbf{k}}]$, and it takes a total length of $80\,m$ to bring the car to a stop over the interval $0 \leq t \leq 80$. This corresponds to a constant tangential acceleration a_T of -15.321 m/s^2 over the entire braking section. This allows the cart to stop in 3.23084 seconds, and results in a backward G-force, G_T of 1.56 during the braking section. This can be seen on Fig. 19, where the last section bounded by two green, vertical lines (the braking section) is the only part of the track where $G_T \neq 0$.

II.H.2. Force, Acceleration, and G loading

This section of the track will not have any critical point due to it's constant acceleration in every direction. The equation to find the tangential acceleration is simply:

$$G_T = \frac{v}{Lg} \tag{12}$$

Where v is the initial velocity and L is the length of this section of the track.

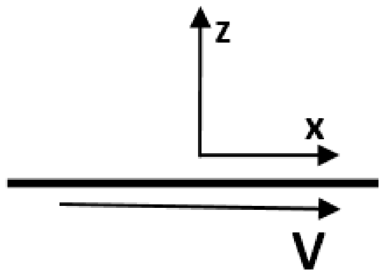

Figure 16: Braking section of the track

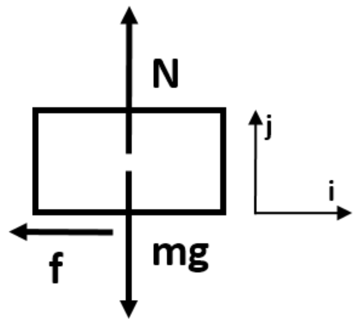

Figure 17: Free Body Diagram of the Braking section

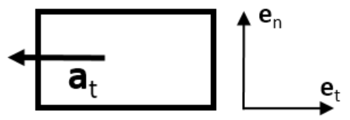

Figure 18: Acceleration on the Braking section

III. Performance Analysis

Now that the roller coaster has been described in sufficient detail, we can do a general performance analysis of the track. In particular, we have analyzed the G loads, speed, and time, which are what the theoretical riders of the roller coaster would be most interested in. There is also a short discussion on the practicality and safety of the roller coaster.

III.A. G Loads

As talked about in the Design section of this lab, the G loading is calculated as $N/mg = G$, where N is the normal force. This means that free fall has no G loading, since there is no normal force, and standing on the floor has exactly 1 G ($N = mg$, $G = mg/mg$). In order to calculate the G forces as a function of distance along the track, we first defined a fixed Δt, then calculated the Δs for that period. We know the velocity at these points as a function of height and as a function of t. We also derived that the change in speed is equal to the average speed over the differential times the rotation angle ψ of the $\hat{\mathbf{N}}$ about the $\hat{\mathbf{B}}$ direction. Then, using the TNB frame calculated at each point and ψ, we can approximate the total normal acceleration as:

$$a_N = \hat{\mathbf{N}} \cdot \mathbf{g} + \frac{\psi * \mathbf{v}}{\Delta t} \tag{13}$$

from (13) G_V and G_L can be calculated based on the bank angle θ as defined in Fig. 5 as:

$$G_V = \frac{sin(\theta)\, a_N}{g} \tag{14}$$

$$G_L = \frac{cos(\theta)\, a_N}{g} \tag{15}$$

G_T is assumed to be zero throughout the track, until the braking section, as the track has no way of applying a force opposite to the direction of motion of the rider (friction-less track). During the braking section, G_T is calculated through basic kinematics equations and 12. For each Δt we calculate G_T, G_L, and G_V, along with the new value of s. This allows us to numerically re-parameterize the behavior of the G-forces throughout each track segment in terms of s. We developed code that could take any track segment parameterization in terms of t and calculate its G's as a function of arc length numerically given its domain and TNB framework, using the ideas developed above. For the implementation of this algorithm, see the `theOriginalGs.m` function and its documentation in Appendix B.

There are five directions of G loading that the passengers of this roller coaster may experience: forward, back, up, down, and lateral. These have respective upward limits of 5, 4, 6, 1, and 3 G. Each subsection in the Design section of this report provides the equations used to calculate the G's, and the actual values have been calculated and compiled in Fig 19 as a function of arc length.

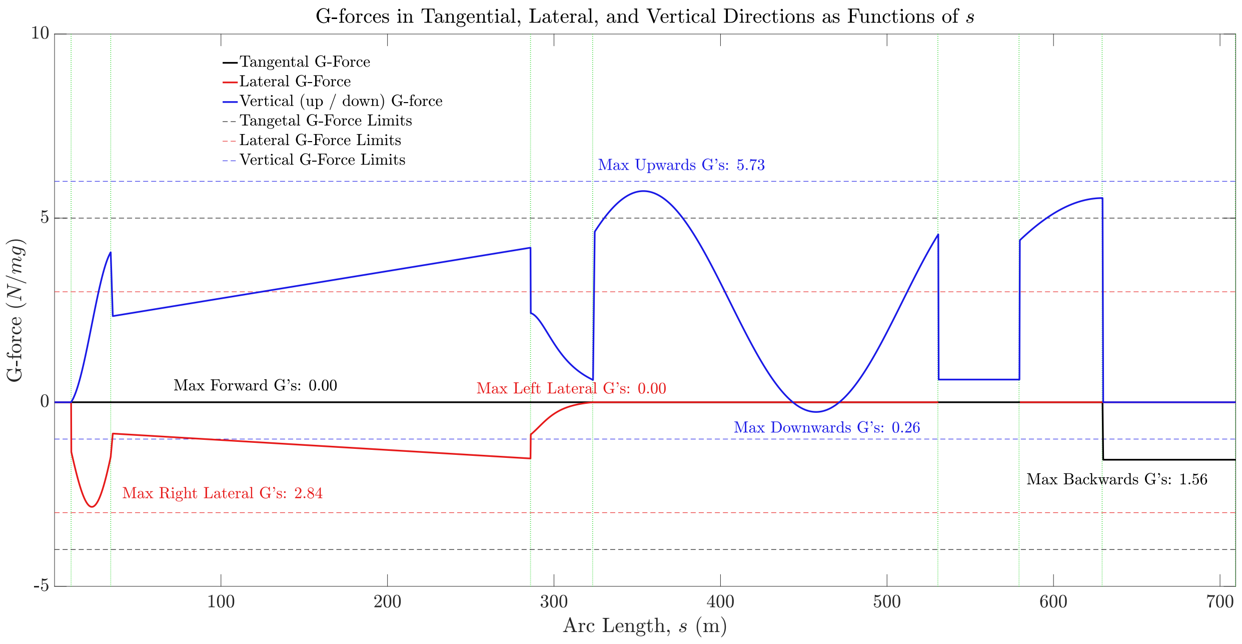

Figure 19: G Loading on the Passenger as a function of position s along the track. *Each green, vertical dotted line denotes the beginning of the next track section. Therefore, a track segment is contained within each set of two vertical lines.*

III.B. Speed

Since we have assumed no dissipative forces over the roller coaster, velocity is purely a function of height at any given point, until the braking section is reached. This means that the velocity can always be calculated from $v(h) = \sqrt{2g(h_0 - h)}$. In order to plot the velocity as a function of arc length, instead of height, we used the same process as described in the "G Loads" subsection above, just stopping before calculating acceleration. This plot can be found in Fig. 20.

III.C. Time

In order to find the total time, the domain (described by the arbitrary parameter t up to this point) of each segment was taken and divided up into 1000 segments for which the arc length (Eq. 1) and the average speed was found to be $26.2m/s$, as can be seen as the black, dotted line on Fig. 20. Finding the time for each segment was simply dividing the length over the speed. The total time found for the roller coaster ride was 27.1 seconds.

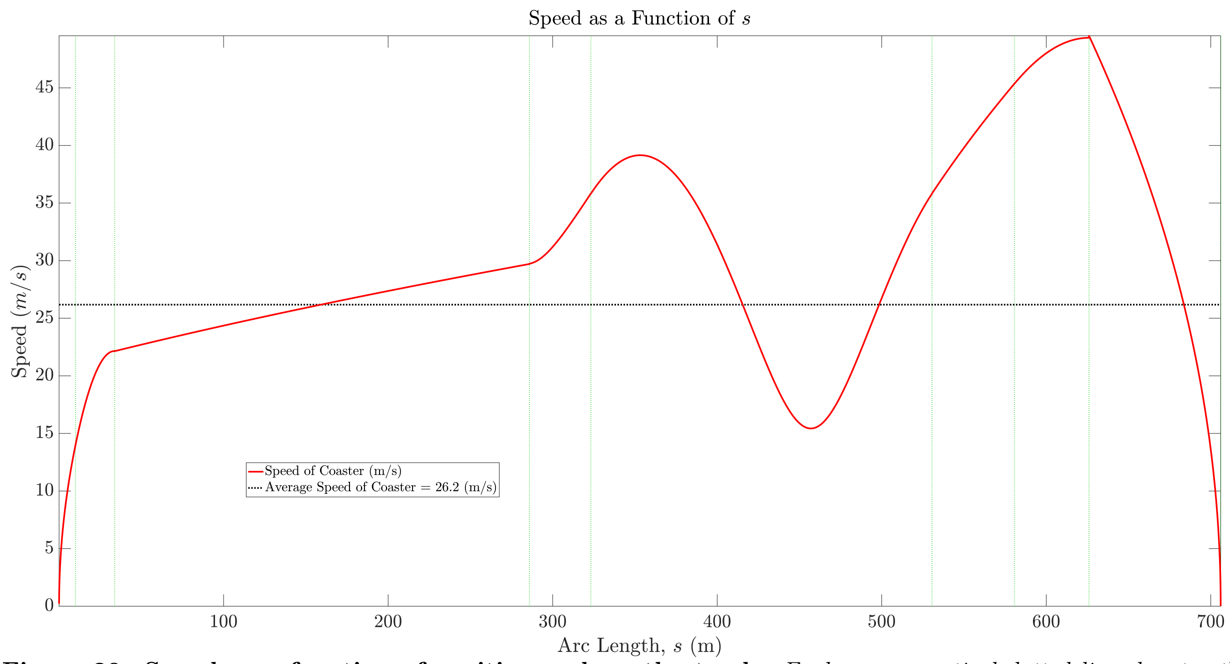

Figure 20: Speed as a function of position s along the track. *Each green, vertical dotted line denotes the beginning of the next track section. Therefore, a track segment is contained within each set of two vertical lines.*

III.D. Practicality and Safety

Our design could be improved by making the zero-G section a parabolic curve rather than a straight drop. The straight drop at the beginning of the roller coaster alone makes this a particularly impractical and dangerous design. Setting up the cars in such a way that you could safely load passengers would be a logistic challenge that would not be worth a short section of Zero G.

The assumption of zero friction also makes this design impractical, as chances are very high that the cart would have lost too much kinetic energy by the time it reaches the loop to make it all the way through. Besides the straight drop and assumed lack of friction, our roller coaster is a reasonably practical design with appropriate transitions to ensure the relative comfort of the passengers.

IV. Conclusions and Recommendations

By designing this roller coaster, we gained some practical experience in design and particle dynamics. Figuring out how to design the roller coaster using separate parts and combining them was a different experience to the other design experiences we have had in this program. We also learned how to develop generalized numeric analysis tools that allowed for changes to the design to be reanalyzed algorithmically, thus greatly improving up design time.

All requirements for this roller coaster were met, as we never passed the G forces we were limited to, and had a final track length of 709.2 meters after 27.1 seconds. The greatest possible improvement would be to lose the Zero G drop and replace it somewhere on the track - perhaps between the helix and loop - with a parabolic curve that simulates free fall from its initial velocity. Adding in a calculation for friction so that the maximum height of the loop was more accurate is another obvious improvement. More possible improvement could be to have a section of track set up to give the rider a break, and to overall increase the time of the ride, though these improvements would purely be for the benefit of the rider.

In terms of learning objectives, this assignment could possibly be improved by requiring the use of all common coordinate frames. In terms of making the lab more engaging, setting it up so we have a code-framework for plotting the positions of the track (even as a ribbon, so we could see the banking) might make the force and acceleration components easier to visualize and understand. From there we could still be expected to figure out how to calculate the velocity and G loading as a function of arc length on our own.

References

[1]Nerem, Steven. ASEN 2003 Lab 1: Roller Coaster Design. CU, 2017. PDF.

Acknowledgments

We would like to thank Marshall Herr for his collaboration on the code.

Appendix A

The primary contributions of each lab member are as follows:

Kayla Gehring: Theory for calculating G and v as a function of arc length, code of transition and main v.1; Document: layout, design, performance analysis, practicality and safety, conclusions and recommendations.
Initials: **KG**

Pierre Guillaud: Force and acceleration diagrams, track element sketches; Document: introduction, design, performance analysis.
Initials: **PG**

Nicholas Renninger: Theory for calculating G and v as a function of arc length, all plotting, final code andfinal track design. Document: Document class and template.
Signature:

[illegible]

Appendix B: MATLAB Code

```
% This is the driver script that calls all of the functions nessecary to
% design and build a rollercoaster track. Has the folowing dependencies:
%
%                     Arrow.m and Derivative.m
%                   (for test code. find these on FEX)
%
% Author: Nicholas Renninger
% Date Modified: 2/2/17


%% Housekeeping

clear variables
close all
clc

% initializations
h_init = 125; % [m]
g = 9.81; % [m/s^2]
NUM_STEPS = 50;
total_time = 0; % [s]
track_length = 0; % [m]
segment_lengths = zeros(1, 8);
segment_idx = 1;
speed = @(h) sqrt(2 * g * (h_init - h)); % [m/s]

% Plot setup
figure_title = sprintf('Path of Roller Coaster in Cartesian Space');
xlabel_string = sprintf('$y$ Coordinate (m)');
ylabel_string = sprintf('$x$ Coordinate (m)');
zlabel_string = sprintf('Height (m)');
LINEWIDTH = 3;
FONTSIZE = 28;

% initialize plot handles
path_plot = figure('Name', 'Path of Coaster');
scrz = get(groot,'ScreenSize');
set(path_plot, 'Position', scrz)
set(groot, 'defaultLegendInterpreter', 'latex');

gPlot = figure('Name', 'G-Force');
vPlot = figure('Name', 'Speed');


%% Create transition to helix segment

%%% 1st transistion Element - 0 G


% define constants
rollAngle = @(t) (0) * (pi / 180) ; % [rad]
t_start = 0; % [s]
t_end = 10; % [s]
```

```
pos_in = [15, 60, h_init]; % [m]

%%% create segment
[R1_t, N_t, T_t, T_out, arc_length] = transition1(t_start, pos_in, ...
                                                  path_plot, t_end, ...
                                                  LINEWIDTH);

% Define ending location of track segment
R1_t_end = [R1_t{1}(t_end), R1_t{2}(t_end), R1_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_and_s_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R1_t, ...
                                            T_t, N_t, false, rollAngle);

% update track lengt
track_length = track_length + arc_length;
segment_lengths(segment_idx) = g_and_s_mat(end, 1);
segment_idx = segment_idx + 1;

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% create 2nd trasnition element %%%

%%% define constants
rollAngle = @(t) ((-70 / 1.5852) * (t - pi/2)) * (pi / 180) ; % [rad]
t_start = pi/2; % [s]
t_end = 3.156; % [s]
pos_in = R1_t_end; % [m]

[R1_t, N_t, T_t, T1_out, arc_length] = transition1andHalf(t_start,....
                                                            pos_in,...
                                                            path_plot, ...
                                                            t_end, ...
                                                            LINEWIDTH);
% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

% Define ending location of track segment
R1_t_end = [R1_t{1}(t_end), R1_t{2}(t_end), R1_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R1_t, ...
                                            T_t, N_t, false, rollAngle);
```

```
%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% Create 2nd segment - helix

%%% define constants
helix_max_radius = 20; % [m]
helix_num_loops = 2;
helix_height = 20; % [m]
helix_bank_angle = @(t) -70 * (pi / 180); % [rad]
t_start = 0; % [s]
pos_in = R1_t_end; % [m]

%%% create helix
[arc_length, R2_t, T2_out,...
 N_t, t_end, T_t] = Helix(helix_max_radius, helix_num_loops, ...
                     helix_height, t_start, pos_in, path_plot, LINEWIDTH);

% Define ending location of track segment
R2_t_end = [R2_t{1}(t_end), R2_t{2}(t_end), R2_t{3}(t_end)];

% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R2_t, ...
                                         T_t, N_t, false,
                                            helix_bank_angle);

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% next track segment - parabolic hill

%%% define constants
rollAngle = @(t) (-(2/3) * t - 70) * (pi / 180) ; % [rad]
```

```
t_start = 0; % [s]
t_end = 30; % [s]
pos_in = R2_t_end; % [m]
T_in = T2_out;

%%% create segment
[R3_t, N_t, T_t, T3_out, arc_length] = para_hill(t_start, pos_in, T_in,...
                                                path_plot, t_end, ...
                                                LINEWIDTH);
% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

% Define ending location of track segment
R3_t_end = [R3_t{1}(t_end), R3_t{2}(t_end), R3_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R3_t, ...
                                          T_t, N_t, false, rollAngle);

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% Create 3rd track segment - loop

%%% define constants
rollAngle = @(t) -90 * (pi / 180) ; % [rad]
t_start = acos(-T3_out(1)); % [s]
t_end = t_start + 2 * pi; % [s]
pos_in = R3_t_end; % [m]
r = 33; % [m]

%%% create segment
[R4_t, N_t, T_t, T4_out, arc_length] = loop(t_start, r, pos_in, ...
                                            path_plot, t_end, LINEWIDTH);
% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

% Define ending location of track segment
R4_t_end = [R4_t{1}(t_end), R4_t{2}(t_end), R4_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
```

```
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R4_t, ...
                                            T_t, N_t, false, rollAngle);

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% transition from loop Section

%%% define constants
rollAngle = @(t) (45) * (pi / 180) ; % [rad]
t_start = 0; % [s]
t_end = abs(R4_t_end(3) / T4_out(3)) - 26.827; % [s]
pos_in = R4_t_end; % [m]
T_in = T4_out;

[R5_t, N_t, T_t, T5_out, arc_length] = transition2(t_start, pos_in, ...
                                                    path_plot, t_end, ...
                                                    T_in, LINEWIDTH);
% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

% Define ending location of track segment
R5_t_end = [R5_t{1}(t_end), R5_t{2}(t_end), R5_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R5_t, ...
                                          T_t, N_t, true, rollAngle);

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% transition into braking section

%%% define constants
rollAngle = @(t) (-90) * (pi / 180) ; % [rad]
t_start = acos(-T5_out(1)); % [s]
```

```
t_end = pi; % [s]
pos_in = R5_t_end; % [m]

[R6_t, N_t, T_t, T6_out, arc_length] = transition3(t_start,....
                                                   pos_in,...
                                                   path_plot, ...
                                                   t_end, ...
                                                   LINEWIDTH);
% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;

% Define ending location of track segment
R6_t_end = [R6_t{1}(t_end), R6_t{2}(t_end), R6_t{3}(t_end)];

%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, segment_time] = theOriginalGs(Domain, NUM_STEPS, R6_t, ...
                                        T_t, N_t, false, rollAngle);

%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);


%% Braking section

%%% define constants
rollAngle = @(t) (-90) * (pi / 180) ; % [rad]
t_start = 0; % [s]
t_end = 80; % [s]
pos_in = R6_t_end; % [m]
T_in = T6_out;

[R7_t, N_t, T_t, T7_out, arc_length] = Brake(t_start, pos_in, ...
                                         path_plot, t_end, LINEWIDTH);

% Define ending location of track segment
R7_t_end = [R7_t{1}(t_end), R7_t{2}(t_end), R7_t{3}(t_end)];

% update track length
track_length = track_length + arc_length;
segment_lengths(segment_idx) = arc_length;
segment_idx = segment_idx + 1;


%%% calculate G's as a function of s
Domain = [t_start, t_end];
[g_temp_mat, ~] = theOriginalGs(Domain, NUM_STEPS, R7_t, ...
                                                    T_t, N_t, false, rollAngle);


%%% Re-calc tangental g's to account for braking
A_T = -15.321; % [m/s^2] based on 80m of track to slow down from 49.5 m/s
G_T_braking = A_T / g; % F_t / mg = G_t = A_t / g
segment_time = 3.23084; % [s] accounting for the decreasing velocity

% update G's matrix
g_temp_mat(:, 2) = g_temp_mat(:, 2) + G_T_braking;


%%% Calculate speed through braking section:

% calculate the change in time for the distance traveled - this is
% done using dV = A_T * dt. Create vector of linearly spaced time
   intervals
time_vec = linspace(0, segment_time, NUM_STEPS);

% initializations
[new_speed_vec, position_vec] = deal(zeros(1, NUM_STEPS));
initial_speed = speed(0); % [m/s]
initial_position = track_length - arc_length; % [m]
idx = 1;

% define initial velocity & position
new_speed_vec(1) = initial_speed;
position_vec(1) = initial_position;

% define position of cart along track as fcn of time
position = @(t) ( (1/2) * A_T * t.^2 ) + (initial_speed .* t) + ...
                initial_position;

% calc speed at each step
for i = 1:NUM_STEPS

    if i > 1 % skip first step

        % calculate arc length along the track at each time
        current_time = time_vec(i);
        position_vec(i) = position(current_time);

        % calc dt
        dt = time_vec(i) - time_vec(i - 1);

        % calculate dSpeed = A_t dt
        dV = A_T * dt;

        % define the new speed after the dt interval
        new_speed_vec(idx) = new_speed_vec(idx - 1) + dV;

    end

    idx = idx + 1;
```

```
end


% create braking_V vector
braking_V(:, 1) = position_vec; % position is one elem longer than V
braking_V(:, 2) = new_speed_vec;


%%% Update G's matrix and total time

% add on change in time to total
total_time = total_time + segment_time;
avg_speed = track_length / total_time;

% add on prev arcLen
g_temp_mat(:, 1) = g_temp_mat(:, 1) + g_and_s_mat(end, 1);
g_and_s_mat = cat(1, g_and_s_mat, g_temp_mat);



%% Plot G's % V over track

clc
makeGPlot(g_and_s_mat, segment_lengths, gPlot);
makeVPlot(g_and_s_mat, braking_V, segment_lengths, avg_speed, ...
          NUM_STEPS, vPlot);

% configure path plot
figure(path_plot)
set(gca, 'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
zlabel(zlabel_string)

%% Print statistics
fprintf('Total Time Elapsed: %0.3g s\n', total_time);
fprintf('Length of the TrackL %0.3g m\n', track_length);
fprintf('Average Speed: %0.3g m/s\n', avg_speed);

function [arc_length, R_t_out, T_end,...
          N_t, t_end, T_t] = Helix(helix_max_radius, ...
                                   helix_num_loops, ...
                                   helix_height,...
                                   t_start, pos_in, path_plot, LINEWIDTH)


    % [arc_length, R_t_out, T_end,...
    %  N_t, t_end, T_t] = Helix(helix_max_radius, ...
    %                                    helix_num_loops, ...
    %                                    helix_height,...
    %                                    t_start, pos_in,...
    %                                    path_plot, LINEWIDTH)
    %
```

```
% Author: Nicholas Renninger
% Date Modified: 2/2/17
%
%
% Describes the loop track segment, after the parabolic hill.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:    t_start - starting value of the domain of the track
%                      segment, given as a non-dimension value of t.
%
% helix_max_radius r - the radius of the loop in meters.
%
%    helix_num_loops - the number of complete revolutions of the track
   .
%
%      helix_height  - the proposed change in height of the track
%                      segment in meters.
%
%            pos_in  - starting position of the track given as
%                      cartesian vector (x, y, z), in meters.
%
%       path_plot - figure handle containing the plot of the track
   path
%
%       LINEWIDTH - width of the line used to plot the track segment
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: R_t_out - conatins parameterization of the path of the
   track
%                    segment. Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the path in each cartesian direction.
%                    Returns position vector in meters.
%
%            T_end - Is a vector that describes the unit Tangental
%                    vector of the TNB frame of the coaster in each
%                    cartesian direction at the last point of the
%                    track segment. I.e. the ending unit
%                    tangental vector of the track segment.
%
%              N_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Normal vector of the TNB frame
%                    of the coaster in each cartesian direction.
%
%              T_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Tangental vector of the TNB
%                    frame of the coaster in each cartesian direction.
%
%            T_out - Is a vector that describes the unit Tangental
%                    vector of the TNB frame of the coaster in each
%                    cartesian direction at the last point in the
%                    domain. I.e. the ending unit tangental vector of
%                      the track segment.
%
%        arc_length - The arc length of the track segment over the
%                      specified domain given by t_start and t_end.
   Given
%                      in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = 2*pi + t_start;
t_end = MAX_T_INTERVAL;
t_interval = MAX_T_INTERVAL - MIN_T_INTERVAL;
t_avg = (MAX_T_INTERVAL + MIN_T_INTERVAL) / 2;

g = 9.81; % m / s^2
H_INIT = 125; % m

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of Helix %%
syms t s t_new
xt = helix_max_radius * sin(helix_num_loops * (t - t_start)) + x_start
   ;
yt = helix_max_radius * cos(helix_num_loops * (t - t_start)) + (
   y_start - ...
                                                      helix_max_radius)
                                                        ;
zt = -helix_height * (t - t_start) / t_interval + z_start;

% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt);
yt_f = matlabFunction(yt);
zt_f = matlabFunction(zt);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];

% T vector
vt = diff(r_vec, t);
T = vt ./ norm(vt);

T_x = matlabFunction(T(1));
```

```
    T_y = matlabFunction(T(2));
    T_z = matlabFunction(T(3));

    T_t = {T_x, T_y, T_z};

    % N vector
    T_prime = diff(T, t);
    N = T_prime ./ norm(T_prime);

    N_x = matlabFunction(N(1));
    N_y = matlabFunction(N(2));
    N_z = matlabFunction(N(3));

    N_t = {N_x, N_y, N_z};

    % B vector
    B = cross(vpa(N), vpa(T)) ./ norm(cross(vpa(N), vpa(T)));

    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T);
    N_t_fcn = matlabFunction(N);
    B_t_fcn = matlabFunction(B);

    % defining T_end, the T vec at the last point of the helix
    T_end = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2 + diff(yt)^2 + diff(zt)^2);
    %speed = sqrt(2 * g * (H_INIT - zt));
    arc_length = double( int(norm_of_deriv, t,...
                            MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%

    figure(path_plot)
    scrz = get(groot,'ScreenSize');
    set(path_plot, 'Position', scrz)

    % plot helix
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
           LINEWIDTH)

    hold on

    % plot start location of helix
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')

%     %%% Draw TNB arrows %%%
%     [x, y, z] = parametric_fcn(t_avg);
%     start = [x, y, z];
%
%     % T
%     stop_T = T_t_fcn(t_avg) * 2 + start;
%     arrow(start, stop_T, 'EdgeColor','g','FaceColor','g');
```

```
%
%      % N
%      stop_N = N_t_fcn(t_avg) * 2 + start;
%      arrow(start, stop_N, 'EdgeColor','r','FaceColor','r');
%
%      % B
%      stop_B =  B_t_fcn(t_avg) * 2 + start;
%      arrow(start, stop_B, 'EdgeColor','b','FaceColor','b');

    %%% set the aspect ratio to see the TNB vectors correctly %%%
    set(gca,'DataAspectRatioMode','manual')
    set(gca,'PlotBoxAspectRatioMode','manual')
    set(gca,'DataAspectRatio',[1 1 1])
    set(gca,'PlotBoxAspectRatio',[1 1 1])

end

function [R_t_out, N_t, T_t, ...
          T_out, arc_length] = loop(t_start, r, pos_in, ....
                                    path_plot, t_end, LINEWIDTH)

    % [R_t_out, N_t, T_t, ...
    %           T_out, arc_length] = loop(t_start, r, pos_in, ....
    %                                     path_plot, t_end, LINEWIDTH)
    %
    % Author: Nicholas Renninger
    % Date Modified: 2/2/17
    %
    %
    % Describes the loop track segment, after the parabolic hill.
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs: t_start - starting value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %               r - the radius of the loop in meters.
    %
    %         pos_in  - starting position of the track given as cartesian
    %                   vector (x, y, z), in meters.
    %
    %       path_plot - figure handle containing the plot of the track
       path
    %
    %           t_end - ending value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %       LINEWIDTH - width of the line used to plot the track segment
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Outputs: R_t_out - conatins parameterization of the path of the
       track
    %                    segment. Is a cell array of the functions (of the
    %                    non-dimensional parametric variable t) that
    %                    describes the path in each cartesian direction.
    %                    Returns position vector in meters.
```

```
%
%               N_t - Is a cell array of the functions (of the
%                     non-dimensional parametric variable t) that
%                     describes the unit Normal vector of the TNB frame
%                     of the coaster in each cartesian direction.
%
%               T_t - Is a cell array of the functions (of the
%                     non-dimensional parametric variable t) that
%                     describes the unit Tangental vector of the TNB
%                     frame of the coaster in each cartesian direction.
%
%             T_out - Is a vector that describes the unit Tangental
%                     vector of the TNB frame of the coaster in each
%                     cartesian direction at the last point in the
%                     domain. I.e. the ending unit tangental vector of
%                     the track segment.
%
%        arc_length - The arc length of the track segment over the
%                     specified domain given by t_start and t_end.
   Given
%                     in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;

g = 9.81; % [m/s^2]

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = -r * sin(t) + (x_start + r*sin(MIN_T_INTERVAL));
yt = y_start;
zt = r * cos(t) + (z_start - r*cos(MIN_T_INTERVAL));


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};
```

```
    %% Defining TNB Vectors %%
    r_vec = [xt, yt, zt];

    % T vector
    vt = diff(r_vec, t);
    T = vt ./ norm(vt);

    T_x = matlabFunction(T(1), 'Vars', t);
    T_y = matlabFunction(T(2), 'Vars', t);
    T_z = matlabFunction(T(3), 'Vars', t);

    T_t = {T_x, T_y, T_z};

    % N vector
    T_prime = diff(T, t);
    N = T_prime ./ norm(T_prime);

    N_x = matlabFunction(N(1), 'Vars', t);
    N_y = matlabFunction(N(2), 'Vars', t);
    N_z = matlabFunction(N(3), 'Vars', t);

    N_t = {N_x, N_y, N_z};


    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T);

    % defining T_end, the T vec at the last point of the helix
    T_out = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2 + diff(zt)^2);
    arc_length = double( int(norm_of_deriv, t,...
                           MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
           LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')



end

function [R_t_out, N_t, T_t, T_out, arc_length] = Brake(t_start, ...
                                                      pos_in, ....
                                                      path_plot, ...
                                                      t_end, LINEWIDTH)

% [R_t_out, N_t, T_t, T_out, arc_length] = Brake(t_start, ...
%                                              pos_in, ....
%                                              path_plot, ...
%                                              t_end, LINEWIDTH)
%
% Author: Nicholas Renninger
% Date Modified: 2/2/17
%
%
% Describes the loop track segment, after the parabolic hill.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs: t_start - starting value of the domain of the track segment,
%                   given as a non-dimension value of t.
%
%         pos_in  - starting position of the track given as cartesian
%                   vector (x, y, z), in meters.
%
%       path_plot - figure handle containing the plot of the track
   path
%
%           t_end - ending value of the domain of the track segment,
%                   given as a non-dimension value of t.
%
%       LINEWIDTH - width of the line used to plot the track segment
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: R_t_out - conatins parameterization of the path of the
   track
%                    segment. Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the path in each cartesian direction.
%                    Returns position vector in meters.
%
%              N_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Normal vector of the TNB frame
%                    of the coaster in each cartesian direction.
%
%              T_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Tangental vector of the TNB
%                    frame of the coaster in each cartesian direction.
%
%            T_out - Is a vector that describes the unit Tangental
%                    vector of the TNB frame of the coaster in each
%                    cartesian direction at the last point in the
%                    domain. I.e. the ending unit tangental vector of
%                    the track segment.
%
%       arc_length - The arc length of the track segment over the
%                    specified domain given by t_start and t_end.
```

```
    Given
%                        in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = t + x_start;
yt = y_start;
zt = z_start;


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];

% T vector
vt = diff(r_vec, t);
T = vt ./ norm(vt);

T_x = matlabFunction(T(1), 'Vars', t);
T_y = matlabFunction(T(2), 'Vars', t);
T_z = matlabFunction(T(3), 'Vars', t);

T_t = {T_x, T_y, T_z};

% N vector
T_prime = diff(T, t);
N = T_prime ./ norm(T_prime);

N_x = matlabFunction(N(1), 'Vars', t);
N_y = matlabFunction(N(2), 'Vars', t);
N_z = matlabFunction(N(3), 'Vars', t);
```

```
    N_t = {N_x, N_y, N_z};


    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T, 'Vars', t);

    % defining T_end, the T vec at the last point of the helix
    T_out = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2);
    arc_length = double( int(norm_of_deriv, t,...
                                MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    zt = @(t) z_start;
    p1 = fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'r:', ...
            'LineWidth', LINEWIDTH);

    hold on

    legend(p1, 'Braking Section', 'location', 'best')

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    [x_e, y_e, z_e] = parametric_fcn(MAX_T_INTERVAL);

    x = [x, x_e];
    y = [y, y_e];
    z = [z, z_e];


    scatter3(x, y, z, 'filled')



end

function makeGPlot(g_and_s_mat, segment_lengths, gPlot)

    % makeGPlot(g_and_s_mat, segment_lengths, gPlot)
    %
    % Authors: Nicholas Renninger, made in collaboration with Marchall
    %          Herr
    % Last Modified: 1/31/17
    %
    %
    % Function takes matrix containing arc lengths and the variation of
       the
    % tangental, vertical, and lateral G's as functions of the length
       along
    % the track
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:
%           gPlot - must be the figure handle to plot the G's on
%
% segment_lengths - vector containing  the length of each track
%                   segment.
%
%     g_and_s_mat - a matrix with arc length [m] in col. 1,
%                   tangent Gs in col. 2, lateral Gs in col. 3,
%                   up/down Gs in col. 4, the ideal roll angle
%                   during the track segment in rad in col. 5,
%                   the size of the t step used during the
%                   calculations in col. 6, and the speed [m/2]
%                   of the coaster at each t value along the
%                   track in col. 7.
%

%% Set up
figure_title = sprintf(['G-forces in Tangential, Lateral, and ' ...
                        'Vertical Directions as Functions of $s$']);
legend_string = {'Tangental G-Force', ...
                 'Lateral G-Force', ...
                 'Vertical (up / down) G-force', ...
                 'Tangetal G-Force Limits', ...
                 'Lateral G-Force Limits', ...
                 'Vertical G-Force Limits'};
xlabel_string = sprintf('Arc Length, $s$ (m)');
ylabel_string = sprintf('G-force $\\left(N / mg\\right)$');

LINEWIDTH = 2.5;
FONTSIZE = 28;
LEGEND_LOCATION = 'best';
G_RANGE = [-5, 10];

%% format data for plotting
s = g_and_s_mat(:, 1)';
g_T = g_and_s_mat(:, 2)';
g_L = g_and_s_mat(:, 3)';
g_V = g_and_s_mat(:, 4)';

s_min = s(1);
s_max = s(end);

%%% Max and Min G in each direction %%%

% Tangental
max_T = max(g_T);
max_T_s = s(find(g_T==max_T,1,'last'));

min_T = min(g_T);
min_T_s = s(find(g_T==min_T,1,'first'));

% Lateral
max_L = max(g_L);
```

```
max_L_s = s(find(g_L==max_L,1,'last'));

min_L = min(g_L);
min_L_s = s(find(g_L==min_L,1,'first'));

% Vertical
max_V = max(g_V);
max_V_s = s(find(g_V==max_V,1,'last'));

min_V = min(g_V);
min_V_s = s(find(g_V==min_V,1,'first'));

%%% G-force limits %%%
num_pts = length(s);

% Tangental
g_T_high = ones(1, num_pts) * 5;
g_T_low = ones(1, num_pts) * -4;

% Lateral
g_L_high = ones(1, num_pts) * 3;
g_L_low = ones(1, num_pts) * -3;

% Vertical
g_V_high = ones(1, num_pts) * 6;
g_V_low = ones(1, num_pts) * -1;

%% draw plots

%%% labels
max_T_str = sprintf('Max Forward G''s: %0.2f', max_T);
min_T_str = sprintf('Max Backwards G''s: %0.2f', abs(min_T));
max_L_str = sprintf('Max Left Lateral G''s: %0.2f', max_L);
min_L_str = sprintf('Max Right Lateral G''s: %0.2f', abs(min_L));
max_V_str = sprintf('Max Upwards G''s: %0.2f', max_V);
min_V_str = sprintf('Max Downwards G''s: %0.2f', abs(min_V));

%%% curve plotting
figure(gPlot)
scrz = get(groot,'ScreenSize');
set(gPlot, 'Position', scrz)

% plot zero G line
plot(linspace(s_min, s_max, 100), zeros(1, 100), ':k', 'LineWidth',...
     LINEWIDTH - 1.2)
hold on

% plotting segment-dividing lines
G_line = linspace(G_RANGE(1), G_RANGE(2), 100);

for i = 1:length(segment_lengths)

    segment_line = ones(1, 100) .* ( segment_lengths(i) + ...
                                     sum(segment_lengths(1:i-1)) );
    plot(segment_line, G_line, ':', ...
            'Color', [0 0.8 0], 'LineWidth', LINEWIDTH - 1.2);
        hold on

end


% plotting G limits
p4 = plot(s, g_T_high,'--', s, g_T_low, '--', ...
            'Color', [0 0 0], 'LineWidth', LINEWIDTH - 1.7);
p5 = plot(s, g_L_high, '--', s, g_L_low, '--', ...
            'Color', [0.9 0.1 0.1], 'LineWidth', LINEWIDTH - 1.7);
p6 = plot(s, g_V_high, '--', s, g_V_low, '--', ...
            'Color', [0.1 0.1 0.9], 'LineWidth', LINEWIDTH - 1.7);

% plotting G-curves
p1 = plot(s, g_T, 'k', 'LineWidth', LINEWIDTH);
p2 = plot(s, g_L, 'r', ...
            'Color', [0.9 0.1 0.1], 'LineWidth', LINEWIDTH);
p3 = plot(s, g_V, 'b', ...
            'Color', [0.1 0.1 0.9], 'LineWidth', LINEWIDTH);


% labeling
text(max_T_s - 500, 0.8, max_T_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0 0 0], 'interpreter', 'latex');
text(min_T_s - 40, min_T*1.3, min_T_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0 0 0], 'interpreter', 'latex');
text(max_L_s*1.1, max_L - 1, max_L_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0.9 0.1 0.1], 'interpreter', 'latex');
text(min_L_s + 80, min_L*1.1, min_L_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0.9 0.1 0.1], 'interpreter', 'latex');
text(max_V_s - 20, max_V*1.1, max_V_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0.1 0.1 0.9], 'interpreter', 'latex');
text(min_V_s*1.1, min_V*1.1, min_V_str, ...
      'HorizontalAlignment', 'center', 'FontSize', FONTSIZE - 10, ...
      'Color', [0.1 0.1 0.9], 'interpreter', 'latex');

% more plot formatting
set(gca, 'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
xlim([s_min, s_max])
ylim(G_RANGE)
title(figure_title)
xlabel(xlabel_string)
ylabel(ylabel_string)
legend([p1, p2, p3, p4(1), p5(1), p6(1)], legend_string, 'location',
    ...
       LEGEND_LOCATION, 'FontSize', FONTSIZE-8, 'interpreter', ...
       'latex', 'Box','off')
```

```
end

function makeVPlot(g_and_s_mat, braking_V, segment_lengths, avg_speed, ...
                   NUM_STEPS, vPlot)

    % makeVPlot(g_and_s_mat, braking_V, segment_lengths, avg_speed, ...
    %           NUM_STEPS, vPlot)
    %
    % Authors: Nicholas Renninger, made in collaboration with Marchall
    %          Herr
    % Last Modified: 1/31/17
    %
    %
    % Function takes matrix containing arc length, speed and plots speed
      as
    % a function of the length along the track
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs:
    %           vPlot - must be the figure handle to plot the velocity on
    %
    %       NUM_STEPS - number of data points per track section
    %                   segment_lengths - an array containg the arc_length
       of
    %                   seach segment
    %
    %       braking_V - matrix containing s in the first column and V in
      the
    %                   second column during the braking section.
    %
    % segment_lengths - vector containing  the length of each track
    %                   segment.
    %       avg_speed - the avg. speed thruoughout the track segment
    %
    %       braking_V - the speed of the coaster as a function of s during
    %                   braking.
    %
    %     g_and_s_mat - a matrix with arc length [m] in col. 1,
    %                   tangent Gs in col. 2, lateral Gs in col. 3,
    %                   up/down Gs in col. 4, the ideal roll angle
    %                   during the track segment in rad in col. 5,
    %                   the size of the t step used during the
    %                   calculations in col. 6, and the speed [m/2]
    %                   of the coaster at each t value along the
    %                   track in col. 7.
    %


    %% Set up
    figure_title = sprintf('Speed as a Function of $s$');
    xlabel_string = sprintf('Arc Length, $s$ (m)');
    ylabel_string = sprintf('Speed $\\left(m / s\\right)$');
    legend_string = {sprintf('Speed of Coaster (m/s)'), ...
                     sprintf('Average Speed of Coaster = %0.3g (m/s)', ...
                              avg_speed)};
LEGEND_LOCATION = 'best';

LINEWIDTH = 2.5;
FONTSIZE = 28;
G_RANGE = [0, 50];

%% format data for plotting
s = g_and_s_mat(1:end - NUM_STEPS, 1);
V = g_and_s_mat(1:end - NUM_STEPS, end);

s_brake = braking_V(:, 1);
v_brake = braking_V(:, 2);

s = cat(1, s, s_brake);
V = cat(1, V, v_brake);

s_min = s(1);
s_max = s(end);

%% draw plots

%%% curve plotting
figure(vPlot);
scrz = get(groot, 'ScreenSize');
set(vPlot, 'Position', scrz)

% plotting segment-dividing lines
V_line = linspace(G_RANGE(1), G_RANGE(2), 100);

for i = 1:length(segment_lengths)

    segment_line = ones(1, 100) .* ( segment_lengths(i) + ...
                                     sum(segment_lengths(1:i-1)) );
    plot(segment_line, V_line, ':', ...
          'Color', [0 0.8 0], 'LineWidth', LINEWIDTH - 1.2)
    hold on

end

% plot avg speed
avg_speed_vec = ones(1, length(s)) * avg_speed;
p1 = plot(s, avg_speed_vec', ':k', 'LineWidth', LINEWIDTH);

% plotting the line
p2 = plot(s, V, 'r', 'LineWidth', LINEWIDTH);

% more plot formatting
set(gca, 'FontSize', FONTSIZE)
set(gca, 'defaulttextinterpreter', 'latex')
set(gca, 'TickLabelInterpreter', 'latex')
legend([p2, p1], legend_string, 'location', ...
   LEGEND_LOCATION, 'FontSize', FONTSIZE - 8, 'interpreter', ...
   'latex')
xlim([s_min, s_max])
```

```
    ylim([0, max(V)])
    title(figure_title)
    xlabel(xlabel_string)
    ylabel(ylabel_string)

end

function [R_t_out, N_t, T_t, T_out, arc_length] = para_hill(t_start, ...
                                                    pos_in, ....
                                                    T_in, ...
                                                    path_plot, ...
                                                    t_end, ...
                                                    LINEWIDTH)

    %  [R_t_out, N_t, T_t, T_out, arc_length] = para_hill(t_start, ...
    %                                                   pos_in, ....
    %                                                   T_in, ...
    %                                                   path_plot, ...
    %                                                   t_end, ...
    %                                                   LINEWIDTH)
    %
    % Author: Nicholas Renninger
    % Date Modified: 2/2/17
    %
    %
    % Describes the parabolic hill track segment, after the helix.
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs: t_start - starting value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %         pos_in  - starting position of the track given as cartesian
    %                   vector (x, y, z), in meters.
    %
    %            T_in - Is a vector that describes the unit Tangental
    %                   vector of the TNB frame of the coaster in each
    %                   cartesian direction at the last point of the
    %                   previous track segment. I.e. the ending unit
    %                   tangental vector of the previous track segment.
    %
    %       path_plot - figure handle containing the plot of the track
       path
    %
    %           t_end - ending value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %       LINEWIDTH - width of the line used to plot the track segment
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Outputs: R_t_out - conatins parameterization of the path of the
       track
    %                    segment. Is a cell array of the functions (of the
    %                    non-dimensional parametric variable t) that
    %                    describes the path in each cartesian direction.
    %                    Returns position vector in meters.
    %
    %               N_t - Is a cell array of the functions (of the
    %                     non-dimensional parametric variable t) that
    %                     describes the unit Normal vector of the TNB frame
    %                     of the coaster in each cartesian direction.
    %
    %               T_t - Is a cell array of the functions (of the
    %                     non-dimensional parametric variable t) that
    %                     describes the unit Tangental vector of the TNB
    %                     frame of the coaster in each cartesian direction.
    %
    %             T_out - Is a vector that describes the unit Tangental
    %                     vector of the TNB frame of the coaster in each
    %                     cartesian direction at the last point in the
    %                     domain. I.e. the ending unit tangental vector of
    %                     the track segment.
    %
    %        arc_length - The arc length of the track segment over the
    %                     specified domain given by t_start and t_end.
       Given
    %                     in meters.
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %% Initial Setup %%
    MIN_T_INTERVAL = t_start;
    MAX_T_INTERVAL = t_end;

    g = 9.81; % [m/s^2]

    % Defining start location of Helix
    x_start = pos_in(1);
    y_start = pos_in(2);
    z_start = pos_in(3);


    %% make parametrization of transition %%
    syms t s t_new
    xt = (t) + x_start;
    yt = y_start;
%     zt = -( (1/2) * g * t^2 ) + (T_in(3) * t) + z_start;
    zt = -(1/50) * (t - (25 * T_in(3)))^2 + z_start;


    % make function from symbolic eqn
    parametric_fcn = matlabFunction(xt, yt, zt);

    % make fcn for return
    xt_f = matlabFunction(xt, 'Vars', t);
    yt_f = matlabFunction(yt, 'Vars', t);
    zt_f = matlabFunction(zt, 'Vars', t);
    R_t_out = {xt_f, yt_f, zt_f};
    %% Defining TNB Vectors %%
    r_vec = [xt, yt, zt];

    % T vector
    vt = diff(r_vec, t);
    T = vt ./ norm(vt);

    T_x = matlabFunction(T(1), 'Vars', t);
    T_y = matlabFunction(T(2), 'Vars', t);
    T_z = matlabFunction(T(3), 'Vars', t);

    T_t = {T_x, T_y, T_z};

    % N vector
    T_prime = diff(T, t);
    N = T_prime ./ norm(T_prime);

    N_x = matlabFunction(N(1), 'Vars', t);
    N_y = matlabFunction(N(2), 'Vars', t);
    N_z = matlabFunction(N(3), 'Vars', t);

    N_t = {N_x, N_y, N_z};


    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T);

    % defining T_end, the T vec at the last point of the helix
    T_out = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2 + diff(zt)^2);
    arc_length = double( int(norm_of_deriv, t,...
                            MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'b', ...
           'LineWidth', LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')


end

function [g_and_s_mat, total_time] = theOriginalGs(Domain, num_steps, ...
                                                   R_t, T_t, N_t,...
                                                   isStraight, Roll)
```

```
% [g_and_s_mat, total_time] = theOriginalGs(Domain, num_steps, ...
%                                               R_t, T_t, N_t,...
%                                               isStraight, Roll)
%
% Authors: Nicholas Renninger, made in collaboration with Marshall
   Herr
% Date Modified: 2/2/17
%
%
% Function takes information about the path, TNB frame, and the bank
% angle of a segment of the track and calculates the G's acting along
% the TNB vectors of the coaster numerically. The G's are calculated
% w.r.t. s, the arc length of the track segment given in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs: Domain - Vector of the starting and ending values of the of
%                  the domain of the track segment parameterized as
   the
%                  non-dimensional parameter t.
%
%     num_steps  - number of linearly spaced steps between the
%                  beginning and end of the domain. The TNB G-forces
%                  are computed each step.
%
%        R_t_out - conatins parameterization of the path of the track
%                  segment. Is a cell array of the functions (of the
%                  non-dimensional parametric variable t) that
%                  describes the path in each cartesian direction.
%                  Returns position vector in meters.
%
%            T_t - Is a cell array of the functions (of the
%                  non-dimensional parametric variable t) that
%                  describes the unit Tangental vector of the TNB
%                  frame of the coaster in each cartesian direction.
%
%            N_t - Is a cell array of the functions (of the
%                  non-dimensional parametric variable t) that
%                  describes the unit Normal vector of the TNB frame
%                  of the coaster in each cartesian direction.
%
%     isStraight - Boolean flag that switches how the computation of
%                  the normal acceleration is done, as the TNB frame
   is
%                  not well-defined for curves with ill defined
%                  curvature.
%
%           Roll - Banking angle as a function of t.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: g_and_s_mat -  a matrix with arc length [m] in col. 1,
%                         tangent Gs in col. 2, lateral Gs in col. 3,
%                         up/down Gs in col. 4, the ideal roll angle
%                          during the track segment in rad in col. 5,
%                          the size of the t step used during the
%                          calculations in col. 6, and the speed [m/2]
%                          of the coaster at each t value along the
%                          track in col. 7.
%
%             total_time - how long it takes to go through the track
%                          segment, given in seconds.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Constants
h_init = 125; % [m]
g = 9.81; % [m / s^2]

% Checking for roll input arg, roll is optional
if nargin < 7
    Roll = @(t) 0;
end

%%% Initializations %%%
syms t
Speed = @(z) sqrt(2 * g * (h_init - z));
g_and_s_mat = zeros(length(num_steps + 1), 7);
dom_init = Domain(1);
dom_final = Domain(2);
total_time = 0; % [s]

% macros for parametric anon. fcn
x_t = R_t{1};
y_t = R_t{2};
z_t = R_t{3};


%% Numeric Caluclation of Gs %%

% define numeric interval and step size
step_size = (dom_final - dom_init) / (num_steps + 1);
first_step = dom_init;
last_step = dom_final + step_size;

% start calculating G's
for i = first_step : step_size : last_step

    if  (i ~= last_step) && (i ~= first_step)


        %%% Calc Arc Length of current and previous step using the arc
        %%% length formula
        r_t_prime = sqrt( diff(x_t(t), t)^2 + diff(y_t(t), t)^2 + ...
                          diff(z_t(t), t)^2 );

        arc_length = vpa( int(r_t_prime, first_step, i) );
```

```
prev = vpa( int(r_t_prime, first_step, i - step_size) );


%%% Defining time differential: dTime = dS/V_avg
dS = (arc_length - prev); % change in arc length
avg_Speed = ( 0.5 * ( Speed(z_t(i - step_size)) + ...
                      Speed(z_t(i)) ) );

d_time = dS / avg_Speed;


%%% Theta is defined as the rotation angle of the N_t vector
%%% about the B_t direction.

cur_N_vec = [N_t{1}(i), N_t{2}(i), N_t{3}(i)];
pre_N_vec = [N_t{1}(i - step_size), N_t{2}(i - step_size),...
             N_t{3}(i - step_size)];
pre_T_vec = [T_t{1}(i - step_size), T_t{2}(i - step_size),...
             T_t{3}(i - step_size)];

Theta = atan( dot(cur_N_vec, pre_T_vec) / ...
              dot(cur_N_vec, pre_N_vec) );


%%% Defining Normal acceleration as: Speed/R^2=
%%% Theta*Vavg/dTime
An = dot(cur_N_vec, [0,0,-1]) * g + Theta * avg_Speed / d_time
    ;

if isnan(An) && ~isStraight

    An = 0;

elseif isStraight

    % find angle between curve and xy plane
    dx = x_t(dom_final) - x_t(dom_init);
    dz = z_t(dom_final) - z_t(dom_init);
    eta = abs(atan(dz / dx));

    % find angle between curve and -k
    phi = pi/2 - eta;

end


%%% Defining the "Ideal" Banked Turn angle (roll) w.r.t. k_hat
%%% s.t. the normal acceleration is completely in the up
%%% direction for the rider e.g. the rider feels all of the
%%% force coming from their seat.

% N_t{1} = x_component of N, N_t{2} = y_component of N, N_t{3}
% = z_component of N
xy_projection_mag = sqrt( (N_t{1}(i))^2 + (N_t{2}(i))^2 );
Roll_Ideal = atan( xy_projection_mag / N_t{3}(i) );
```

```
            %%% Defining G's in tangental, lateral, and vertical
                directions
            %%% w.r.t. the rider
            G_T = 0;

            % G_L & G_V depends on the roll angle
            G_L = cos(Roll(i)) * (An / g);

            if isStraight
                G_V = sin(phi);
            else
                G_V = sin(Roll(i)) * (An / g);
            end


            %%% creating output matrix
            row_idx = round((i - first_step) / step_size);
            g_and_s_mat(row_idx, :) = [arc_length, G_T, G_L, G_V,...
                                       Roll_Ideal, d_time, avg_Speed];

            total_time = double(total_time + d_time);

        end
    end

end

function [R_t_out, N_t, T_t,...
          T_out, arc_length] = transition1(t_start, pos_in, ...
                                           path_plot, ...
                                           t_end, LINEWIDTH)

    % [R_t_out, N_t, T_t,...
    %           T_out, arc_length] = transition1(t_start, pos_in, ...
    %                                            path_plot, ...
    %                                            t_end, LINEWIDTH)
    %
    % Author: Nicholas Renninger
    % Date Modified: 2/2/17
    %
    %
    % Describes the 1st half of the 1st transition curve of the track
    % segment leading into the helix
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs: t_start - starting value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %         pos_in  - starting position of the track given as cartesian
    %                   vector (x, y, z), in meters.
    %
    %       path_plot - figure handle containing the plot of the track
      path
```

```
%
%           t_end - ending value of the domain of the track segment,
%                   given as a non-dimension value of t.
%
%       LINEWIDTH - width of the line used to plot the track segment
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: R_t_out - conatins parameterization of the path of the
   track
%                    segment. Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the path in each cartesian direction.
%                    Returns position vector in meters.
%
%              N_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Normal vector of the TNB frame
%                    of the coaster in each cartesian direction.
%
%              T_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Tangental vector of the TNB
%                    frame of the coaster in each cartesian direction.
%
%            T_out - Is a vector that describes the unit Tangental
%                    vector of the TNB frame of the coaster in each
%                    cartesian direction at the last point in the
%                    domain. I.e. the ending unit tangental vector of
%                    the track segment.
%
%       arc_length - The arc length of the track segment over the
%                    specified domain given by t_start and t_end.
   Given
%                    in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;

g = 9.81; % [m/s^2]

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = x_start;
yt = y_start;
```

```
zt = -t + z_start;


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];

% T vector
vt = diff(r_vec, t);
T = vt ./ norm(vt);

T_x = matlabFunction(T(1), 'Vars', t);
T_y = matlabFunction(T(2), 'Vars', t);
T_z = matlabFunction(T(3), 'Vars', t);

T_t = {T_x, T_y, T_z};

% N vector
T_prime = diff(T, t);
N = T_prime ./ norm(T_prime);

N_x = matlabFunction(N(1), 'Vars', t);
N_y = matlabFunction(N(2), 'Vars', t);
N_z = matlabFunction(N(3), 'Vars', t);

N_t = {N_x, N_y, N_z};


% defining function for TNB frame as functions of t
T_t_fcn = matlabFunction(T, 'Vars', t);

% defining T_end, the T vec at the last point of the helix
T_out = T_t_fcn(MAX_T_INTERVAL);

%% Arc Length of Track %%
norm_of_deriv = sqrt(diff(zt)^2);
arc_length = double( int(norm_of_deriv, t,...
                          MIN_T_INTERVAL, MAX_T_INTERVAL) );

%% Plotting %%
figure(path_plot)

% plot segment
xt = @(t) x_start;
yt = @(t) y_start;
fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
        LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')



end

function [R_t_out, N_t, T_t, ...
         T_out, arc_length] = transition1andHalf(t_start, pos_in, ...
                                                 path_plot, t_end, ...
                                                 LINEWIDTH)


    % [R_t_out, N_t, T_t, ...
    %  T_out, arc_length] = transition1andHalf(t_start, pos_in, ...
    %                                          path_plot, t_end, ...
    %                                          LINEWIDTH)
    %
    % Author: Nicholas Renninger
    % Date Modified: 2/2/17
    %
    %
    % Describes the 2nd half of the 1st transition curve of the track
    % segment leading into the helix
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs: t_start - starting value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %          pos_in  - starting position of the track given as cartesian
    %                   vector (x, y, z), in meters.
    %
    %        path_plot - figure handle containing the plot of the track
       path
    %
    %            t_end - ending value of the domain of the track segment,
    %                   given as a non-dimension value of t.
    %
    %        LINEWIDTH - width of the line used to plot the track segment
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Outputs: R_t_out - conatins parameterization of the path of the
       track
    %                    segment. Is a cell array of the functions (of the
    %                    non-dimensional parametric variable t) that
    %                    describes the path in each cartesian direction.
    %                    Returns position vector in meters.
    %
    %              N_t - Is a cell array of the functions (of the
    %                    non-dimensional parametric variable t) that
%                        describes the unit Normal vector of the TNB frame
%                        of the coaster in each cartesian direction.
%
%                 T_t - Is a cell array of the functions (of the
%                        non-dimensional parametric variable t) that
%                        describes the unit Tangental vector of the TNB
%                        frame of the coaster in each cartesian direction.
%
%               T_out - Is a vector that describes the unit Tangental
%                        vector of the TNB frame of the coaster in each
%                        cartesian direction at the last point in the
%                        domain. I.e. the ending unit tangental vector of
%                        the track segment.
%
%          arc_length - The arc length of the track segment over the
%                        specified domain given by t_start and t_end.
    Given
%                        in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;

g = 9.81; % [m/s^2]
r = 15; % [m]

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = -r * sin(t) + x_start + r;
yt = y_start;
zt = r * cos(t) + z_start;


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];
```

```
    % T vector
    vt = diff(r_vec, t);
    T = vt ./ norm(vt);

    T_x = matlabFunction(T(1), 'Vars', t);
    T_y = matlabFunction(T(2), 'Vars', t);
    T_z = matlabFunction(T(3), 'Vars', t);

    T_t = {T_x, T_y, T_z};

    % N vector
    T_prime = diff(T, t);
    N = T_prime ./ norm(T_prime);

    N_x = matlabFunction(N(1), 'Vars', t);
    N_y = matlabFunction(N(2), 'Vars', t);
    N_z = matlabFunction(N(3), 'Vars', t);

    N_t = {N_x, N_y, N_z};


    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T);

    % defining T_end, the T vec at the last point of the helix
    T_out = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2 + diff(zt)^2);
    arc_length = double( int(norm_of_deriv, t,...
                            MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
           LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')



end

function [R_t_out, N_t, T_t, T_out, arc_length] = transition2(t_start, ...
                                                         pos_in, ....
                                                         path_plot, ...
                                                         t_end, T_in, ...
                                                         LINEWIDTH)
```

```
% [R_t_out, N_t, T_t, T_out, arc_length] = transition2(t_start, ...
%                                                      pos_in, ....
%                                                      path_plot, ...
%                                                      t_end, T_in,
   ...
%                                                      LINEWIDTH)
%
% Author: Nicholas Renninger
% Date Modified: 2/2/17
%
%
% Describes the 2nd transition curve of the track segment.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs: t_start - starting value of the domain of the track segment,
%                   given as a non-dimension value of t.
%
%         pos_in  - starting position of the track given as cartesian
%                   vector (x, y, z), in meters.
%
%       path_plot - figure handle containing the plot of the track
   path
%
%           t_end - ending value of the domain of the track segment,
%                   given as a non-dimension value of t.
%
%       LINEWIDTH - width of the line used to plot the track segment
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: R_t_out - conatins parameterization of the path of the
   track
%                    segment. Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the path in each cartesian direction.
%                    Returns position vector in meters.
%
%              N_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Normal vector of the TNB frame
%                    of the coaster in each cartesian direction.
%
%              T_t - Is a cell array of the functions (of the
%                    non-dimensional parametric variable t) that
%                    describes the unit Tangental vector of the TNB
%                    frame of the coaster in each cartesian direction.
%
%            T_out - Is a vector that describes the unit Tangental
%                    vector of the TNB frame of the coaster in each
%                    cartesian direction at the last point in the
%                    domain. I.e. the ending unit tangental vector of
%                    the track segment.
%
```

```
%        arc_length - The arc length of the track segment over the
%                     specified domain given by t_start and t_end.
   Given
%                     in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;


% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = T_in(1) * t + x_start;
yt = y_start;
zt = T_in(3) * t + z_start;


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];

% T vector
vt = diff(r_vec, t);
T = vt ./ norm(vt);

T_x = matlabFunction(T(1), 'Vars', t);
T_y = matlabFunction(T(2), 'Vars', t);
T_z = matlabFunction(T(3), 'Vars', t);

T_t = {T_x, T_y, T_z};

% N vector
T_prime = diff(T, t);
N = T_prime ./ norm(T_prime);

N_x = matlabFunction(N(1), 'Vars', t);
N_y = matlabFunction(N(2), 'Vars', t);
```

```
    N_z = matlabFunction(N(3), 'Vars', t);

    N_t = {N_x, N_y, N_z};


    % defining function for TNB frame as functions of t
    T_t_fcn = matlabFunction(T, 'Vars', t);

    % defining T_end, the T vec at the last point of the helix
    T_out = T_t_fcn(MAX_T_INTERVAL);

    %% Arc Length of Track %%
    norm_of_deriv = sqrt(diff(xt)^2 + diff(zt)^2);
    arc_length = double( int(norm_of_deriv, t,...
                             MIN_T_INTERVAL, MAX_T_INTERVAL) );

    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
           LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);

    scatter3(x, y, z, 'filled')



end

function [R_t_out, N_t, T_t, ...
          T_out, arc_length] = transition3(t_start, pos_in, ...
                                           path_plot, t_end, LINEWIDTH)


    % [R_t_out, N_t, T_t, ...
    %           T_out, arc_length] = transition3(t_start, pos_in, ...
    %                                            path_plot, t_end,
    %                                            LINEWIDTH)
    %
    %
    % Author: Nicholas Renninger
    % Date Modified: 2/2/17
    %
    %
    % Describes the 3rd transition curve of the track segment leading into
    % the braking section.
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inputs: t_start - starting value of the domain of the track segment,
%                     given as a non-dimension value of t.
%
%          pos_in   - starting position of the track given as cartesian
%                     vector (x, y, z), in meters.
%
%        path_plot - figure handle containing the plot of the track
   path
%
%            t_end - ending value of the domain of the track segment,
%                     given as a non-dimension value of t.
%
%        LINEWIDTH - width of the line used to plot the track segment
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs: R_t_out - conatins parameterization of the path of the
   track
%                      segment. Is a cell array of the functions (of the
%                      non-dimensional parametric variable t) that
%                      describes the path in each cartesian direction.
%                      Returns position vector in meters.
%
%               N_t - Is a cell array of the functions (of the
%                      non-dimensional parametric variable t) that
%                      describes the unit Normal vector of the TNB frame
%                      of the coaster in each cartesian direction.
%
%               T_t - Is a cell array of the functions (of the
%                      non-dimensional parametric variable t) that
%                      describes the unit Tangental vector of the TNB
%                      frame of the coaster in each cartesian direction.
%
%             T_out - Is a vector that describes the unit Tangental
%                      vector of the TNB frame of the coaster in each
%                      cartesian direction at the last point in the
%                      domain. I.e. the ending unit tangental vector of
%                      the track segment.
%
%        arc_length - The arc length of the track segment over the
%                      specified domain given by t_start and t_end.
   Given
%                      in meters.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initial Setup %%
MIN_T_INTERVAL = t_start;
MAX_T_INTERVAL = t_end;

r = 55; % [m]

% Defining start location of Helix
x_start = pos_in(1);
y_start = pos_in(2);
```

```
z_start = pos_in(3);


%% make parametrization of transition %%
syms t s t_new
xt = -r * sin(t) + (x_start + r*sin(MIN_T_INTERVAL));
yt = y_start;
zt = r * cos(t) + (z_start - r*cos(MIN_T_INTERVAL));


% make function from symbolic eqn
parametric_fcn = matlabFunction(xt, yt, zt);

% make fcn for return
xt_f = matlabFunction(xt, 'Vars', t);
yt_f = matlabFunction(yt, 'Vars', t);
zt_f = matlabFunction(zt, 'Vars', t);
R_t_out = {xt_f, yt_f, zt_f};


%% Defining TNB Vectors %%
r_vec = [xt, yt, zt];

% T vector
vt = diff(r_vec, t);
T = vt ./ norm(vt);

T_x = matlabFunction(T(1), 'Vars', t);
T_y = matlabFunction(T(2), 'Vars', t);
T_z = matlabFunction(T(3), 'Vars', t);

T_t = {T_x, T_y, T_z};

% N vector
T_prime = diff(T, t);
N = T_prime ./ norm(T_prime);

N_x = matlabFunction(N(1), 'Vars', t);
N_y = matlabFunction(N(2), 'Vars', t);
N_z = matlabFunction(N(3), 'Vars', t);

N_t = {N_x, N_y, N_z};


% defining function for TNB frame as functions of t
T_t_fcn = matlabFunction(T);

% defining T_end, the T vec at the last point of the helix
T_out = T_t_fcn(MAX_T_INTERVAL);

%% Arc Length of Track %%
norm_of_deriv = sqrt(diff(xt)^2 + diff(zt)^2);
arc_length = double( int(norm_of_deriv, t,...
                            MIN_T_INTERVAL, MAX_T_INTERVAL) );
```

```
    %% Plotting %%
    figure(path_plot)

    % plot segment
    yt = @(t) y_start;
    fplot3(xt, yt, zt, [MIN_T_INTERVAL, MAX_T_INTERVAL], 'LineWidth', ...
           LINEWIDTH)

    hold on

    % plot start location of segment
    [x, y, z] = parametric_fcn(MIN_T_INTERVAL);
    scatter3(x, y, z, 'filled')



end
```