Roller Coaster Design

Aufa Amirullah[*] Abdullah Almugairin[†] Abdulla Al Ameri[‡] Mohamed Aichiouene[§]

University of Colorado Boulder, ASEN 2003, Lab 1, 1/28/2019

The purpose of this lab experiment is to use theoretical knowledge of particle dynamics and kinematics to design a track for a friction less roller coaster and model the G's experienced on a rider. The track has to be fun and safe all while consisting of at least 3 different several elements. Programming software was used to analyze the coaster, and after doing so we found that we met all requirements. We never exceeded 6 G's and the coaster was about 1114 meters long. This project showed the difficulty of practical roller coaster design.

Nomenclature

α	Theta values initialized for curves
ρ	Curvature of the bank turn
a_{picked}	Picked constant declaration rate.
D	Change in position of horizontal section, typically only one direction
FBD	Free body diagrams
g	Gravitational acceleration, 9.81 m/s2 in this case
$G_{Lateral}$	G force felt laterally to the coaster's line of motion
G_{Normal}	G force felt perpendicular to the coaster's line of motion
L_{ramp}	Length from ramp from absolute initial to final position
R	Radius/curvature of ramps,transitions, or loops
v_0	Initial instantaneous velocity in that section
v_x	Instantaneous Velocity of the coaster in the direction
X	X positions on the coaster, final array
x_0	Previous x position values before start of section
$x_{section,f}$	Final coordinate position of that specific section
$x_{section,i}$	Initial coordinate position of that specific section
$x_{section}$	x values calculated in that specific section, including loop, transitions, etc.
Y	Y positions on the coaster, final array
y_0	Previous y position values before start of section
$y_{section,f}$	Final Y coordinate position of that specific section
$y_{section,i}$	Initial Y coordinate position of that specific section
$y_{section}$	y values calculated in that specific section, including loop, transitions, etc.
Z	Z positions on the coaster, final array
$z_{section,f}$	Final Z coordinate position of that specific section
$z_{section,i}$	Initial Z coordinate position of that specific section
$z_{section}$	z values calculated in that specific section, including loop, transitions, etc.

[redacted]

I. Introduction

A. Objectives

The objective of this lab is to gain an understanding of the forces felt on an actual roller coaster. In this lab experiment, we were required to design a roller coaster that would be both fun and safe for a general populace.

We had many requirements when it came to the actual design. First, the length of the track cannot be more than 1200 meters long and the stop point of the roller coaster had to be at the ground level (i.e. height is zero). Moreover, the coaster must include at least three different type of track elements with a smooth transition between each section. There must also be one section of the track that produces zero G throughout the entire element and a banked turn at a constant or changing altitude. To meet safety limits, the G forces felt on the rider must not exceed a certain number in each direction. These limits are provided in the table below.

Direction	Maximum allowed G-force
Forward	5G
Backward	4G
Up	6G
Down	1G
Lateral	3G

Table 1: The maximum G-force someone can experience at any direction.

Furthermore, several assumptions were made regarding the behavior of the coaster to provide a simpler analysis for our purposes as students. These assumptions include treating the roller coaster train and people inside as a particle or point mass, assuming the track is friction-less and that it has no initial velocity. The train must also remain above the ground and is locked to the track. Using these premises and safety limits, track design can begin.

B. Track Design

The track design was handled using equations of motion, both rectilinear and curvillinear, as parametric functions with an arbitrary parameter. Many of these equations were made using basic kinematics and circle geometry to easily model the smooth transitions seen in the figure 1 below.

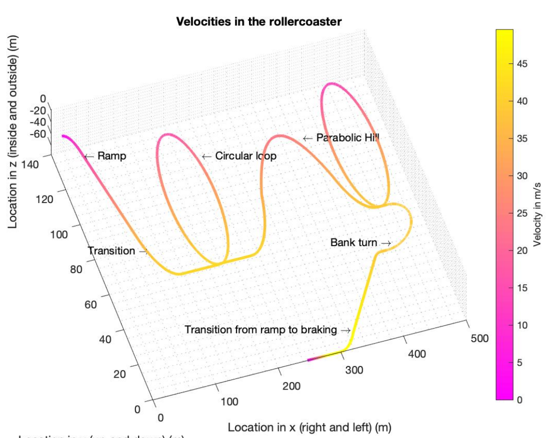

Figure 1: A 3D-plot of the roller coaster track with velocity, color coded. It is important to notice that the loops are perfectly rounded 3D, except that the scale of the x, y and z-axis are not consistent.

Our coaster met the 3 element requirement given to us starting off with a ramp and including loops, banks, and hills; all of which are tailored so that the G forces felt never exceed safety limits.

II. Design

In this section, The team will explain how each unique element in our design works along with the free body diagrams and important equations as well as the expression for the acceleration and G loading of the track throughout the entire section. The track path included seven unique elements and they are: 1- Transitions 2- Ramps 3- Horizontal Straight Lines 4- Loops 5- Parabolic Hill 6- Banked Turn 7- Deceleration segment.

In the track design, several components that were considered as the critical designs are loop, parabolic hill, and the banked turn where each of these elements will be discussed in more detail in the section below. Including the qualitative descriptions as well as the mathematical approach of modeling the number of G's experienced by the person setting inside the cart.

A. Transitions

In a roller coaster, transitions are made to connect two other elements that do not seem to be having the same path direction. In this roller coaster, specifically, there were a total of 6 transitions used throughout the entire track. To create transitions in MATLAB, we assumed that each transition is a portion of an arc length that is cut from a circle with radius R. To better analyze a transition segment, let us consider the very first transition that starts from 125 m in the y-axis, Figure 2, and is connected to the down inclined ramp.

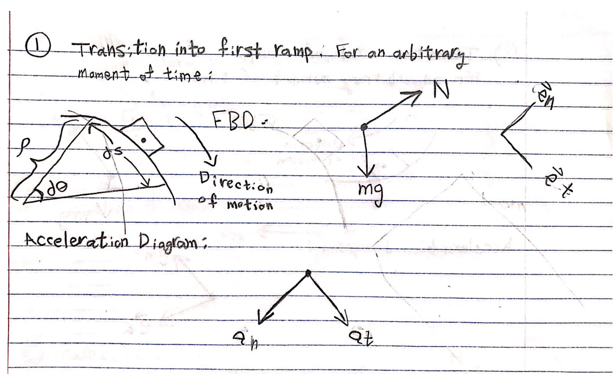

Figure 2: Free Body Diagram of Transition Section

With a free body diagram, and a tangent-normal coordinate system, the team figured out the range of angles of the circle, Figure 2. These angles were ultimately be used to compute the x, y and z position of the cart as can be seen in equations 1, 2 and 3 below.

$$X = x_{trans} - R * sind(\alpha) \tag{1}$$

$$Y = y_{trans} - R - R * cosd(\alpha) \tag{2}$$

$$Z = z_{trans} * ones(length(\alpha)) \tag{3}$$

In order to calculate the length of the track in the specific segment, the team used calculated the arc length of the circle.

B. Ramps

The roller coaster designed included two ramps. Both ramps were analyzed using tilted cartesian coordinates. A free body diagram of such a case is in Figure 3.

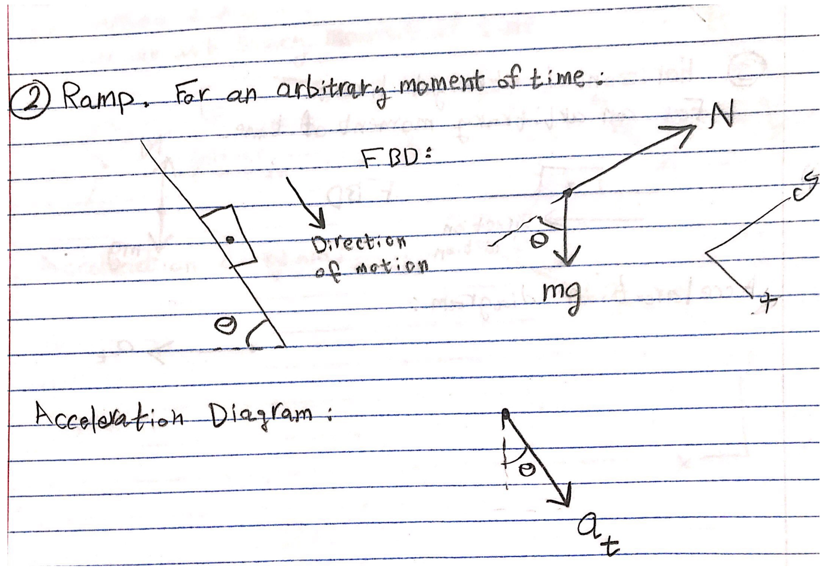

Figure 3: Free Body Diagram of Ramps Section

Furthermore, the length of the ramp itself, which is an input to the function, is what gets concatenated since it is just a 2D line. In order to find out the height and the length of the ramp, simple trigonometric principles were applied. The length of the ramp was added to the initial value of the x component to find the final x position. On the other hand, the height was subtracted from the initial height because both ramps were going down and height was lost through. Since both ramps do not vary along the z-axis, two equations were necessary to compute the new x and y position. The following equations were used:

$$y_{ramp} = L_{ramp} * sind(\alpha) \tag{4}$$

$$x_{ramp} = L_{ramp} * cosd(\alpha) \tag{5}$$

$$X = x_0 - x_{ramp} \tag{6}$$

$$Y = y_0 - y_{ramp} \tag{7}$$

C. Horizontal Straight Lines

The linear sections were straight forward. In the MATLAB code, the distance of the section is first determined. The distance was just the horizontal change in x positions. New x,y, and z position arrays are created that equal the original x,y, and z position arrays. The appropriate array gets the distance value added to it. These new arrays are then concatenated to the original x,y, and z position arrays. All of this was done in cartesian coordinates, as seen in the figure below.

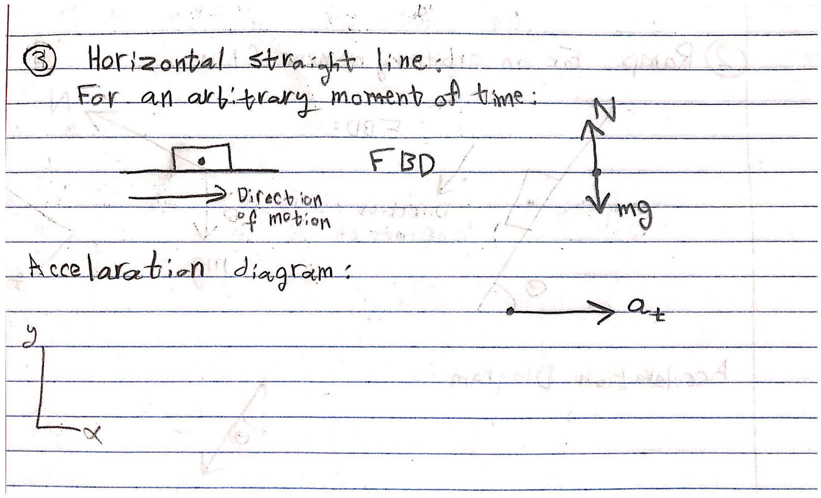

Figure 4: Free Body Diagram of Horizontal Straight Lines

The sections mostly only moved in the x direction; the equations are as follows:

$$X = x_0 + x_{line} + D \tag{8}$$

$$Y = y_0 + y_{line} \tag{9}$$

$$Z = z_0 + z_{line} \tag{10}$$

D. Loops

To design the loops of the roller coaster, a function was created that would take in user specifications about the loop and design it around them. These inputs include radius, initial velocity as it enters the loop, initial x, y, and z positions. To start, a range of angles array is created that goes from 0 to 360, simulating the degrees of a full loop, using a tangent-normal coordinate system, as shown in Figure 5 below.

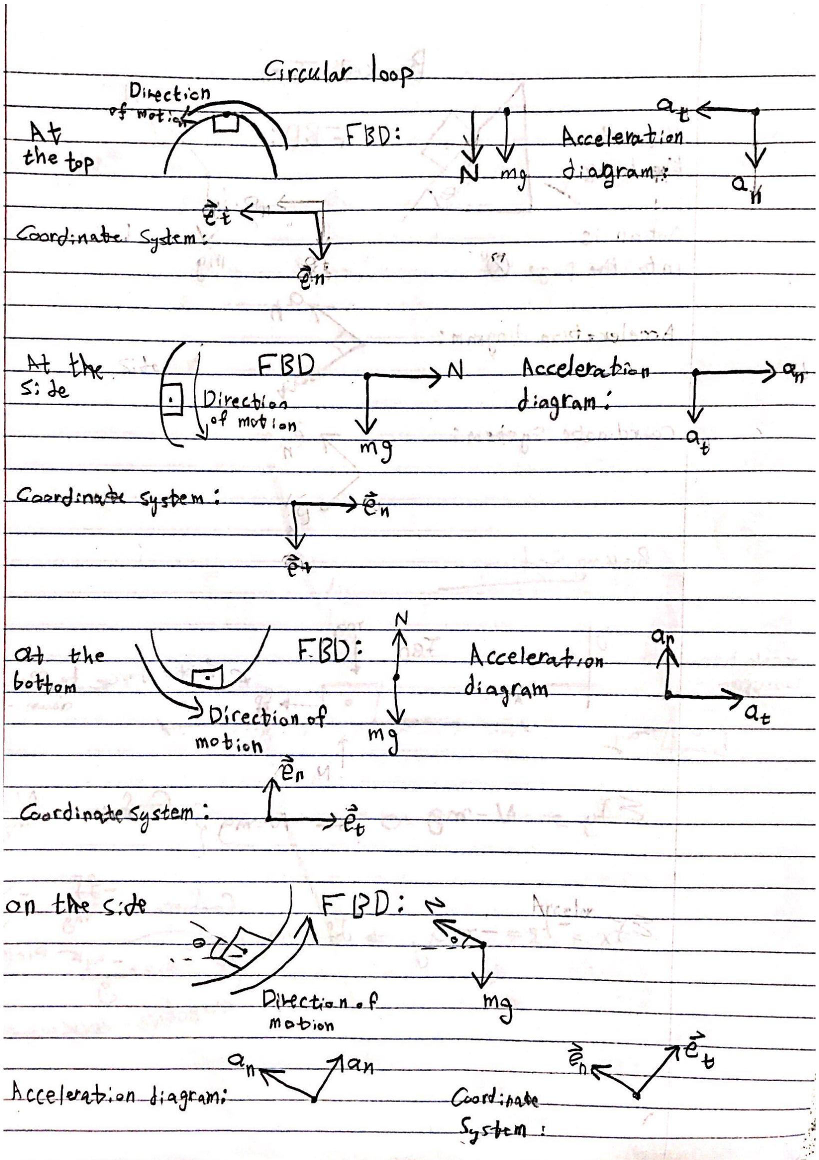

Figure 5: Free Body Diagrams of Loops Section

Those range of angles are then plugged into parametric equations to find the new x and y positions of the loop. The z position is just an array of values from the initial z value to final specified z value using the linspace MATLAB function. The distance traveled by the rider in the loops was found by calculating the circumference of those sections. These calculated values are then concatenated to the previous x and y positions arrays. The equations are as follows:

$$y_{loop} = y_{loop,i} + (R - R * cosd(\alpha)) \tag{11}$$

$$x_{loop} = x_{loop,i} + R * sind(\alpha) \tag{12}$$

$$z_{loop} = linspace(z_i, z_f, length(\alpha)) \tag{13}$$

E. Parabolic Hill

A unique function for the parabolic hill was created using a while loop. The parabolic hill starts with an initial height, goes up, then goes back to that same initial height. The figure below will show how forces acting on the coaster when the coaster is passing the parabolic section.

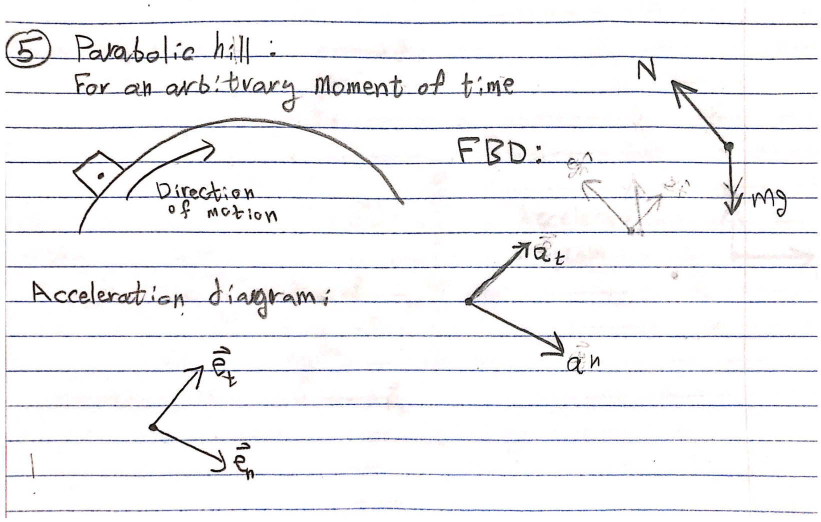

Figure 6: Free Body Diagram of Parabolic Hill

Therefore, we used the initial height as an argument for the while loop in which it keeps running until the argument is not satisfied anymore; i.e. the cart comes back to the same initial height. The significance in this segment is that the person setting inside the track would experience a zero G force because the parabolic path follows a perfect trajectory just like if the cart were launched from a cannon. Moreover, this element adheres to free fall conditions; thus the following equations to find the x and y positions were used:

$$y_{parabola} = y_{parabola,i} + v_0 * sin(\alpha) - \frac{1}{2} * g * t^2 \tag{14}$$

$$x_{parabola} = x_0 + v_x * t \tag{15}$$

For the calculation of the of the length of the track of the parabola, the following equations were used:

$$Arc - length = \frac{s}{2} + \frac{b^2}{8a} * log(\frac{4a + s}{b}) \tag{16}$$

$$a = y_{parabola,f} - y_{parabola,i} \tag{17}$$

$$b = x_{parabola,f} - x_{parabola,i} \tag{18}$$

$$s = \sqrt{b^2 + 16a^2} \tag{19}$$

F. Banked Turn

When it comes to the bank turn, it is easier to think of it as a ramp, but instead of moving in a path tangent to the ramp, you move in a path normal to the ramp.

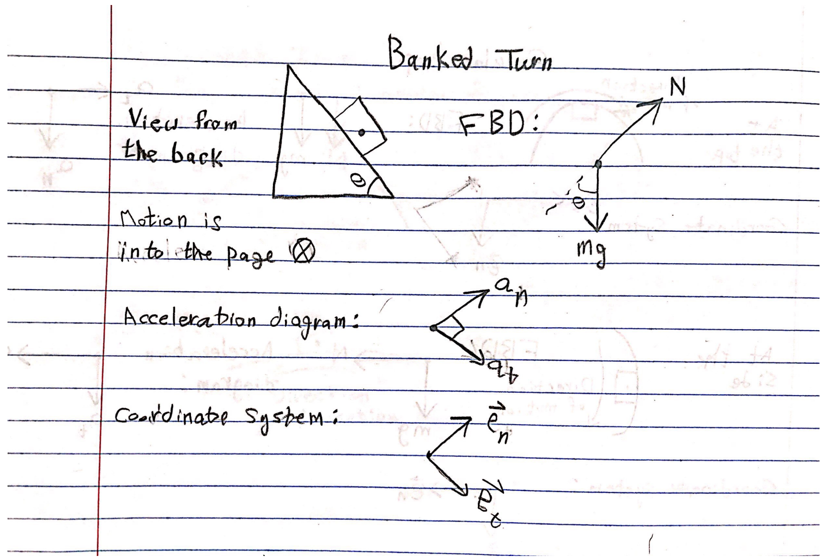

Figure 7: Free Body Diagram of Banked Turn

As a result, the force from the track itself breaks down into two type of forces, normal, and lateral, each in return will make the rider experience different G force. Using a coordinate system that is tangent and normal to the ramp that represents the bank at each turn provides a way to express the lateral and normal G forces.

The following equations were used to estimate the G force experienced in each direction:

$$G_{Normal} = \frac{\frac{v^2}{\rho} * \sin\theta + g * \cos\theta}{g} \tag{20}$$

$$G_{Lateral} = \frac{-\sin\theta + (g - (\frac{v^2}{\rho} * \cos\theta))}{g} \tag{21}$$

When it came to estimating to distance covered on the bank turn, it was convenient to assume it is half a circle, and simply the distance covered will be the arc length for a fixed chosen arbitrary radius, and an angel of 180 degrees.

$$Arclength = r * \theta \tag{22}$$

Hence, the cart will not change the height, and as a result the velocity will be constant. However, the position in both x (right and left) and z (in and out) relative to an observer from far will change. Given an initial position (x0,y0,z0) in Cartesian coordinate system, this is how the change in position was computed, where r is simply the radius of curvature, which is fixed, and θ is from 0 degrees to 180°.

$$z(\theta) = z0 - r + r * cos(\theta) \tag{23}$$

$$x(\theta) = x0 + r * sin(\theta) \tag{24}$$

G. Breaking Segment

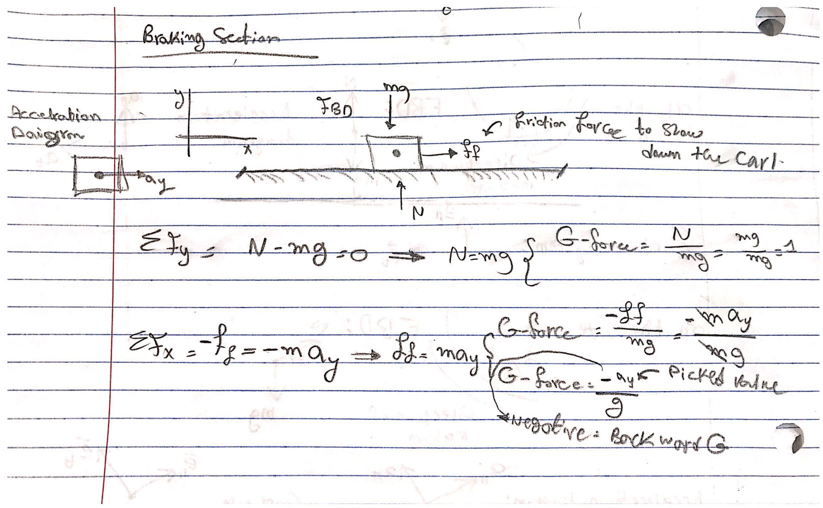

Figure 8: Free Body Diagram of Ramps Section

By the end of the ride, as the cart converts a lot of its potential energy if not all of it (if the cart goes back to 0 m) to kinetic energy, the cart needs to be brought to a full stop. This was done through applying a constant deceleration braking section. Through this braking section, the rider will feel 1 G-force normal to his body, however, the G-force felt backward/forward from the seats would depend on the constant deceleration rate. According to table 1, the maximum G-force someone can feel in the backward/forward direction should be less than 4 G's. After drawing the FBD, shown in Figure 8, it becomes clear that the picked constant deceleration rate will be given by the equation

$$G - force = \frac{a_{picked}}{g} \tag{25}$$

To maximize the G experienced, G was chosen to be 3.9, which in return would require a constant declaration rate of 38.26 $\frac{m}{s^2}$. This would also require a distance of 32.05 meters to come to a full stop, which keeps the roller caster within the total given limit of 1200 meters. As the cart comes to a stop, the total distance covered was 1149.2 meters.

The length of the track required to make the braking and the stop was calculated using the fact the our acceleration is constant, which makes the kinematic equations for constant acceleration useful. The following equation:

$$v_f{}^2 = v_0{}^2 + 2 * a_{picked} * (x_f - x_0) \tag{26}$$

was rearranged to solve for the track length required given by $x_f - x_0$ to stop the roller-coaster.

H. G Force

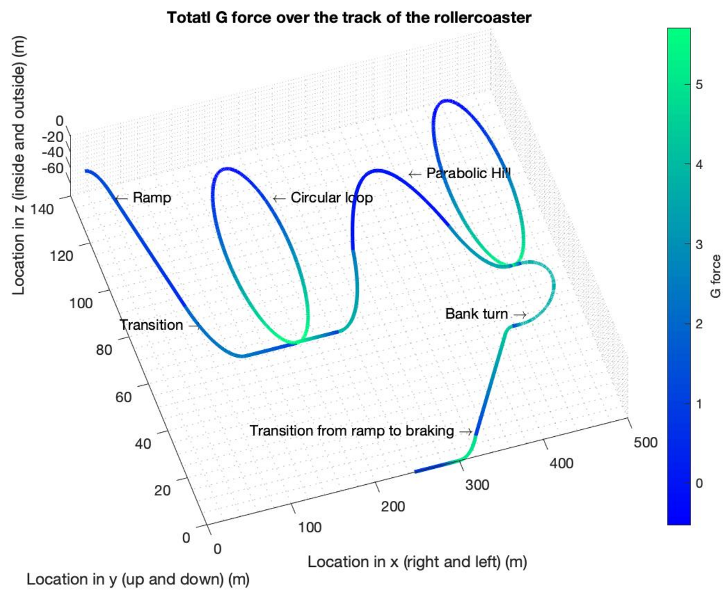

Figure 9: G force modeled in the roller-coaster

The following table summarizes the total magnitude of the G force experienced by the rider at each of the seven unique segment of the track.

Segment	$\lvert G_{max}\rvert$
Transition to Ramp	1.50
Transition out of Ramp	1.50
Transition into Parabolic Hill	3.90
Transition out of Parabolic Hill	3.61
Transition out of Banked Turn	4.41
Transition to ground	5.17
Ramp Down	0.50
Horizontal straight lines	5
Loop	5.48
Parabolic hill	0.00
Banked turns (normal)	3.72
Banked turns (lateral)	1.21

Table 2: G force for each unique segment in the design

The color-coding in figure 1, as well as figure 9 was done using customized function made available to the public via Math-Works, and done by Stillfried, G

. As it can be seem, figure 9 models the G force accurately. The G force changes along each track element, and sometimes it has unique values, such as at the parabolic hill section where the value is 0 G. The Performance reflection section has more analysis about figure 9.

I. Performance reflection

The goal of this lab was to design a fun and practical roller coaster, and after modeling the G forces felt on the coaster the team can safely say that the requirements are met. The design has many sections and elements that have the rider experience a wide range of G forces and large drops in three dimensions. The highest magnitude of G's felt by the rider was 5.85 G's upward, which is slightly below the safety limits. The practicality of the design isn't as promising however. The design only works under a friction-less track assumption which allows total conservation of energy. Real coasters utilize mechanical energy to achieve this goal, and this design would require a lot of mechanical support to function properly. The most difficult part of this lab was modeling the smooth transitions between sections. Almost all our transitions are smooth except for two areas. These sections are still slightly sharp since they go from a line to ramps. We discovered that this results in a sharp uptick in G's felt by the rider; however these G's still never exceed safety limits and can be considered negligible. Our group is confident that the design will be safe and remain fun.

III. Conclusions and Recommendations

Throughout this lab, the team learned that a roller coaster experiences a wide range of G-forces in each section. From all the knowledge has gained, we are convinced that the coaster is safe to run since the requirements were all met including safety limits.

Although the team managed to meet all the requirements to design the coaster,there is still room for improvement. The ride could be more fun and include a helix. However, this improvement idea would require more length to make that happen. We also could go back and fix some of the transitions to make them smoother.

IV. References

[1]James, Stewart *Essential Calculus, Second Edition*, James Steward, 2007

[2]Axelrad, P., `ASEN 2003 Lab 1`, Canvas, Aerospace Engineering Sciences Department, ASEN 2003, University of Colorado at Boulder, Spring 2019

[3]Stillfried, G., "MathWorks," 3D colored line plot Available: https://www.mathworks.com/matlabcentral/fileexchange/23566-3d-colored-line-plot?focused=5139154tab=function.

V. Acknowledgments

Thank you to Prof. Axelrad and the TA's who were really helpful when it came to modeling G's and parametric equations. Collaboration with student Adam Elsayed as well.

VI. Appendix

A. A: Team member contributions

	plan	model	Experiment	Results	Report	code
Name						
Mohamed Aicioune	2	1	1	1	1	1
Abdullah Amirullah	1	2	2	1	1	1
Abdulla AlAmeri	1	1	1	2	2	2
Aufa Amirullah	1	1	1	1	1	1

B. B: Matlab code

RollerCoaster.m

```
%% info
% comments needs to be added.
%% housekeeping

clear;
clc;
close all;

%% define constants

% define all inital conditions for the first segment
g = 9.81;
h0 = 125 ; %initial height in meters

% velocity function
syms h
v(h) = sqrt ( 2 * g * (h0 - h)) ;

Vi = double(v(h0)); %initial velocity
Ti = 0; %time in s
Xi = 0; % in m
Yi = h0; % in m
Zi = 0; % in m
a0x = 0;
angle = 60;
r = 20;
t0 = 0;

```

```
[TimeNew GNew LocaNew VelocNew DistanceCovered] = TransitionToDownRamp(r,angle,Xi,Yi,Zi);

 % concatate:
 TotalDistanceCovered = DistanceCovered;
 G = cat(1,1,GNew');
 xPosit = cat(1,0,LocaNew(1,:)');
 yPosit = cat(1,h0,LocaNew(2,:)');
 zPosit = cat(1,0,LocaNew(3,:)');
 xVeloc = cat(1,Vi,VelocNew');
 %t = cat(1,0,TimeNew(1));

 % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 Coordinate_i = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
     /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];


%% ramp down

Ramptheta = 60 ;
Length2 = 60;
 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;

 [ TimeNew GNew LocaNew VelocNew] = RampDown(Vi,0, Ramptheta,Length2, y0, x0, z0, h0 );
   G = cat(1,G,GNew');
   TotalDistanceCovered = cat(1,TotalDistanceCovered,Length2);
 xPosit = cat(1,xPosit,LocaNew(1,:)');
 yPosit = cat(1,yPosit,LocaNew(2,:)');
 zPosit = cat(1,zPosit,LocaNew(3,:)');
 xVeloc = cat(1,xVeloc,VelocNew');

 % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
     /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate_i,CoordinateNew);

 %t = cat(1,t,TimeNew);
 %% transition off rampCurvture

 Curvture = 50; % Choosen arbitrary.

 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;

 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_fromRampDown(t0,x0, y0, z0,
     Ramptheta,Curvture);

```

```
 G = cat(1,G,GNew');
 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
xPosit = cat(1,xPosit,LocaNew(1,:)');
yPosit = cat(1,yPosit,LocaNew(2,:)');
zPosit = cat(1,zPosit,LocaNew(3,:)');
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);

%% Linear section: transition, For 5 meters

distance = 60; %Linear distance

x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

GNew = 1;
 G = cat(1,G,GNew');
 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
xPosit = cat(1,xPosit,x0+distance);
yPosit = cat(1,yPosit,y0);
zPosit = cat(1,zPosit,z0);
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);
%% circular loop:

% initial info.

x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
r = 37; %loop Raduis
t=0;
[ TimeNew GNew LocaNew VelocNew DistanceCovered ] = CircularLoop(xVeloc, t, r, x0, y0, z0,
    h0);

 G = cat(1,G,GNew');
```

```
TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
xPosit = cat(1,xPosit,LocaNew(1,:)');
yPosit = cat(1,yPosit,LocaNew(2,:)');
zPosit = cat(1,zPosit,LocaNew(3,:)');
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);

%% Linear section: transition, For 5 meters

distance = 50; %Linear distance

 x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

 GNew = 1;
 G = cat(1,G,GNew');
  TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
xPosit = cat(1,xPosit,x0+distance);
yPosit = cat(1,yPosit,y0);
zPosit = cat(1,zPosit,z0);
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);

 %% transition into hill

Curvture = 60; % Choosen arbitrary.

x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

[ TimeNew GNew LocaNew VelocNew DistanceCovered ] = Transition_into(t0,x0, y0, z0,Ramptheta
    ,Curvture);


 G = cat(1,G,GNew');
```

```
     TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
 xPosit = cat(1,xPosit,LocaNew(1,:)');
 yPosit = cat(1,yPosit,LocaNew(2,:)');
 zPosit = cat(1,zPosit,LocaNew(3,:)');
 xVeloc = cat(1,xVeloc,VelocNew');

 % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate,CoordinateNew);

 %t = cat(1,t,TimeNew);


 %% parabolic hill:


x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
theta = 45;
a0 = 0;
v = double(v(y0));
t0 = 0;

 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = ParabolaicHill(t0, y0, x0, z0, theta, a0, v);

  G = cat(1,G,GNew');
  TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
 xPosit = cat(1,xPosit,LocaNew(1,:)');
 yPosit = cat(1,yPosit,LocaNew(2,:)');
 zPosit = cat(1,zPosit,LocaNew(3,:)');
 xVeloc = cat(1,xVeloc,VelocNew(1,:)');

 % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate,CoordinateNew);

 %t = cat(1,t,TimeNew);

 %% transition off the hill.

 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;
Ramptheta = 50;

 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_out(t0,x0, y0, z0,Ramptheta,
```

```
    Curvture);


 G = cat(1,G,GNew');
 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
xPosit = cat(1,xPosit,LocaNew(1,:)');
yPosit = cat(1,yPosit,LocaNew(2,:)');
zPosit = cat(1,zPosit,LocaNew(3,:)');
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);


%% Linear section: transition, For 5 meters

distance = 5; %Linear distance

 x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

GNew = 1;
 G = cat(1,G,GNew');
 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
xPosit = cat(1,xPosit,x0+distance);
yPosit = cat(1,yPosit,y0);
zPosit = cat(1,zPosit,z0);
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);

 %% circular loop:

% initial info.

x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
r = 35; %loop Raduis

```

```matlab
% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);


%%
[ TimeNew GNew LocaNew VelocNew DistanceCovered] = CircularLoop(xVeloc, t, r, x0, y0, z0,h0);


G = cat(1,G,GNew');
TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
xPosit = cat(1,xPosit,LocaNew(1,:)');
yPosit = cat(1,yPosit,LocaNew(2,:)');
zPosit = cat(1,zPosit,LocaNew(3,:)');
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

%t = cat(1,Time,TimeNew);
%% Linear section: transition, For 5 meters

distance = 10; %Linear distance

 x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

GNew = 1;
 G = cat(1,G,GNew');
  TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
xPosit = cat(1,xPosit,x0+distance);
yPosit = cat(1,yPosit,y0);
zPosit = cat(1,zPosit,z0);
xVeloc = cat(1,xVeloc,VelocNew');

% get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

% t = cat(1,t,TimeNew);
```

```
%% Banked Turn

 % initial info.
 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 r = 39; %loop Raduis
 BankAngle = 50;

 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = BankTurn(BankAngle, t, r, x0, y0, z0,h0);
  G = cat(1,G,GNew(1,:)');
  TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
 xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
 yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
 zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
 xVeloc = cat(1,xVeloc,VelocNew);

 % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate,CoordinateNew);


  %% Linear section: transition, For 5 meters

 distance = 10; %Linear distance

 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;

 GNew = 1;
  G = cat(1,G,GNew);
    TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
 xPosit = cat(1,xPosit,x0-distance);
 yPosit = cat(1,yPosit,y0);
 zPosit = cat(1,zPosit,z0);
 xVeloc = cat(1,xVeloc,VelocNew);

  % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate,CoordinateNew);


% t = cat(1,t,TimeNew);

 %% ramp down but in opposite direction to the other ramp
```

```
%% ramp down

  Curvture = 30; % Choosen arbitrary.
  Ramptheta = 20;
 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;

 [ TimeNew GNew LocaNew VelocNew DistanceCovered ] = Transition_fromBankedTurn(t0,x0, y0, z0,Ramptheta,Curvture);

  G = cat(1,G,GNew');
  TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
 xPosit = cat(1,xPosit,LocaNew(1,:)');
 yPosit = cat(1,yPosit,LocaNew(2,:)');
 zPosit = cat(1,zPosit,LocaNew(3,:)');
 xVeloc = cat(1,xVeloc,VelocNew');
% t = cat(1,t,TimeNew);


 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;
Ramptheta = 25.1 ;
length1 = 86.6;
[ TimeNew GNew LocaNew VelocNew] = RampDownOpposite(t0,Ramptheta,length1,y0, x0, z0 );

  G = cat(1,G,GNew');
   TotalDistanceCovered = cat(1,TotalDistanceCovered,length1);
 xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
 yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
 zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
 xVeloc = cat(1,xVeloc,VelocNew');

   % get coordinate of a point in the middle of the segment so it can be
 % named on the plot at that point.

  CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
 Coordinate = cat(3,Coordinate,CoordinateNew);



 %% Transition to ground

 x0 = xPosit(length(xPosit));
 y0 = yPosit(length(yPosit));
 z0 = zPosit(length(zPosit));
 t0 = 0;
 r = 60;
 curvature = 30;
 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_toGround(x0, y0, z0,curvature, r);
```

```
G = cat(1,G,GNew');
TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
xVeloc = cat(1,xVeloc,VelocNew');


   % get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
     /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
Coordinate = cat(3,Coordinate,CoordinateNew);

  %% Linear section: transition, For 5 meters

distance = 20; %Linear distance

x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;

GNew = 1;
G = cat(1,G,GNew');
TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
xPosit = cat(1,xPosit,x0-distance);
yPosit = cat(1,yPosit,y0);
zPosit = cat(1,zPosit,z0);
xVeloc = cat(1,xVeloc,VelocNew');
% t = cat(1,t,TimeNew);


%% Braking and stopping


x0 = xPosit(length(xPosit));
y0 = yPosit(length(yPosit));
z0 = zPosit(length(zPosit));
t0 = 0;
[TimeNew GNew LocaNew VelocNew] = BreakAndStop(x0,y0,z0);

 G = cat(1,G,GNew');
xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
xVeloc = cat(1,xVeloc,VelocNew');

   % get coordinate of a point in the middle of the segment so it can be
% named on the plot at that point.

 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2)));  LocaNew(2,(floor(length(LocaNew)
     /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
```

```
 Coordinate = cat(3,Coordinate,CoordinateNew);


 %% All velocities =
 syms h
v = @(h) sqrt ( 2 * g * (h0 - h)) ;

AllV = v(yPosit);
AllV(end) = 0;

%% print total distance

fprintf('\n ' );
fprintf('Total distance covered: %6.2f %12.8f \n ',sum(TotalDistanceCovered))
fprintf(' meters. \n ' );

 %% plot

 figure(1);

 color_line3(xPosit,yPosit,zPosit,AllV,'LineWidth',2.5);
 colormap spring;
 c = colorbar;
 c.Label.String = 'Velocity in m/s';
 grid minor;
 title('Velocities in the rollercoaster');
 xlabel('Location in x (right and left) (m)');
 ylabel('Location in y (up and down) (m)');
 zlabel('Location in z (inside and outside) (m)');
view([-18 80])

%put texts on the graph to label segments.
text(Coordinate(1,1,1)-45,Coordinate(2,1,2),Coordinate(3,1,3),' Transition \rightarrow');

text(Coordinate(1,1,4)-45,Coordinate(2,1,5),Coordinate(3,1,6),'\leftarrow Ramp');

text(Coordinate(1,1,7)-40,Coordinate(2,1,8),Coordinate(3,1,9),'\leftarrow Circular loop');

text(Coordinate(1,1,10)-40,Coordinate(2,1,11)+40,Coordinate(3,1,12),'\leftarrow Parabolic
    Hill');


text(Coordinate(1,1,13)-80,Coordinate(2,1,14),Coordinate(3,1,15),'Bank turn \rightarrow');

text(Coordinate(1,1,16)-240,Coordinate(2,1,17)+30,Coordinate(3,1,18),'Transition from ramp
    to braking \rightarrow');


saveas(gcf,'Gforce.jpg')


% name the sections

% -__- !!

```

```
 figure(2);

colormap winter;
color_line3(xPosit,yPosit,zPosit,G,'LineWidth',2.5);
c = colorbar;
c.Label.String = 'G force';
grid minor;
title('Totatl G force over the track of the rollercoaster');
xlabel('Location in x (right and left) (m)');
ylabel('Location in y (up and down) (m)');
zlabel('Location in z (inside and outside) (m)');
view([-18 80])

%put texts on the graph to label segments.
text(Coordinate(1,1,1)-45,Coordinate(2,1,2),Coordinate(3,1,3),' Transition \rightarrow');

text(Coordinate(1,1,4)-45,Coordinate(2,1,5),Coordinate(3,1,6),'\leftarrow Ramp');

text(Coordinate(1,1,7)-40,Coordinate(2,1,8),Coordinate(3,1,9),'\leftarrow Circular loop');

text(Coordinate(1,1,10)-40,Coordinate(2,1,11)+40,Coordinate(3,1,12),'\leftarrow Parabolic Hill');


text(Coordinate(1,1,13)-80,Coordinate(2,1,14),Coordinate(3,1,15),'Bank turn \rightarrow');

text(Coordinate(1,1,16)-240,Coordinate(2,1,17)+30,Coordinate(3,1,18),'Transition from ramp to braking \rightarrow');

saveas(gcf,'Velocity.jpg')
```

BankTurn.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] = BankTurn( Banktheta,t,raduis,x0,y0,z0, RollerCoasterHeight )
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module a
a bank turn.

% -------------------------------------
Inputs:

1- Banktheta : the angel which the bank it turned at.

2-  t: initial time relative to the whole roller coaster.

```

```
3— raduis: Raduis of curvature, which is constant here.

4— y0: initial y position.

5— x0: initial x position.

6— z0: initial z position.

7— RollerHeight: Roller coaster maximum height.

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


Outputs:

1— TimeElapsed : time just spent on this segment.

2— Outputs_G: G's at eaxh (x,y,z) coordinate.

3— Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4— Outputs_Velocity: Velocity at each point on the rollercoaster.

5— ArcLength: distance covered in this segment.
% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


%}

g = 9.81;

%% code
% Arc length
ArcLength = Banktheta * pi / 180 * raduis;
% height is fixed, thus velocity is.
v = sqrt ( 2 * g * (RollerCoasterHeight — y0)) ;

% this normal force is devided by m.
Normal = (((v^2 )/raduis ) * sind(Banktheta)) + g*cosd(Banktheta) ;

% Lateral force
Lateral = —sind(Banktheta )*( g — (((v^2 )/raduis )*cosd(Banktheta ))) ;

% Gs felt in the direction normal to ramp
Normal_G = Normal / 9.81 ;

% Gs felt in the direction lateral to ramp
Lateral_G = Lateral / 9.81 ;

%substuite from theta = 0 degrees to 180, the whole bank turn.
alphaRange = 0:.01:180 ;


% use the range of the angles in the equations for the postion in the x, y
% and z
```

```
CurrentZ = z0 - raduis + raduis*cosd(alphaRange);
CurrentX = x0 + raduis*sind(alphaRange);
CurrentY = y0 * ones(length(alphaRange),1) ;

% write outputs
Outputs_G = [ Normal_G*ones(1,length(CurrentX)) ; Lateral_G*ones(1,length(CurrentX)) ];
ArcLength = [ArcLength];
Outputs_Loc = [ CurrentX  ; CurrentY' ; CurrentZ ] ;
Outputs_Velocity = [ ones(1,length(alphaRange))'*v ] ;
TimeElapsed = [];
fprintf('The banked turn generates a maximum magnitude of: %6.2f %12.8f and %6.2f %12.8f\n',
    Normal_G,Lateral_G )
fprintf(' G, normal and lateral, respectively. \n ' );




end
```

BreakAndStop.m

```
function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity] = BreakAndStop(x0,y0,z0)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module the
breaking and stop section.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:

1- x0: initial x position.

2- y0: initial y position.

3- z0: initial z position.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Outputs:

1- TimeElapsed : time just spent on this segment.

2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

```

```
4- Outputs_Velocity: Velocity at each point on the rollercoaster.


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -



%}
%% Define constants


g = 9.81;






%% g's
upGs = 1;
backGs = 3.9;
N = sqrt(upGs^2 + backGs^2);

%% initial velocity
VelocFinal = sqrt(2 * g * (125 - y0));

%% give final position based on velocity
PositionFinalX = x0 - VelocFinal^2/(2 * backGs * 9.81);
PositionFinalY = y0;
PositionFinalZ = z0;

%track length
LengthOfTrack = VelocFinal^2/(2 * backGs * 9.81);

%% write outputs

Outputs = [ PositionFinalX ; PositionFinalY ; z0] ;
TimeElapsed = [];
Outputs_G = [upGs];
Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
Outputs_Velocity = [ VelocFinal ] ;

end
```

CircularLoop.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity Circumference] = CircularLoop(
    V0, t0, r, x0, y0, z0,RollerHeight)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
```

```
— Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module the
Circular loop with a fixed raduis.

% ——————————————————————————————————————
Inputs:

Inputs:
1— v0 : Initial velocity (Total magnitude).

2—  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.

3— r: raduis of circle, which is equivalent to raduis of curvature in this case.

4— x0: initial x position.

5— y0: initial y position.

6— z0: initial z position.

7— RollerHeight: Roller coaster maximum height.

% ——————————————————————————————————————


Outputs:

1— TimeElapsed : time just spent on this segment.

2— Outputs_G: G's at eaxh (x,y,z) coordinate.

3— Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4— Outputs_Velocity: Velocity at each point on the rollercoaster.

5— Circumference : which is equivalent to the total distance covered.


% ——————————————————————————————————————


%}



%% using Normal tangent coordinate system.

% we can know our height based on our simple geometry.
Circumference = 2 * pi * r;
thetaStep = 0:0.1:360;
g = 9.81;
```

```
%syms theta_current

currentHeight = y0 + (r - r*cosd(thetaStep));
currentX = x0 + r*sind(thetaStep);

%
%syms currentHeight
v = sqrt ( 2 * g * (RollerHeight - (y0 + r - r*cosd(thetaStep)))) ;

% get value for  accelration

an = (v.^2)/r; %normal accelration
at = g*sind(thetaStep); %tangential accelration

Normal = (an) + g*cosd(thetaStep) ; % normal force (N divided by m, so later you can just
    multiply by 1/g!) to get g's.

% Normal force will make the


%% prepare outputs

zf = 1; %final z position, will assume it's one because cart needs to go into the page to
    finish the loop.

% sub in thetas to get current points:

OutputY = currentHeight;
OutputX = currentX;
OutputZ = linspace(z0,zf,length(OutputX));


G = double(Normal./9.81);

fprintf('The second loop generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
fprintf(' G, forward and upward. \n ' );

%% write outputs
Circumference = [Circumference];
Outputs_G = [ G ] ;
Outputs_Loc = [ OutputX ; OutputY ; OutputZ ] ;
Outputs_Velocity = [ v ] ;

TimeElapsed = 0 ;

end
```

RampDown.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity] = RampDown(v0, t0, RampAngle,
     Length, y0, x0, z0, RollerHeight )
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

```

```
Done by:
— Abdullah AlMugirun
— Mohamed Aichiouene
— Aufa Amirullah
— Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module a
ramp down, where the user will determine the ramp specifications, and
initial conditions, and the function will return the end status.
% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

note: here all x and y is tilted, so that +x is pointing with the heading
vector, i.e where the rider is looking.

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
Inputs:
1— v0 : Initial velocity (Total magnitude).

2—  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.

3— RampAngle: relative to horizon, in degrees.

4— Length: hypotenuse of the ramp.

5— y0: initial y position.

6— x0: initial x position.

7— z0: initial z position.

8— RollerHeight: Roller coaster maximum height.

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


Outputs:

1— TimeElapsed : time just spent on this segment.

2— Outputs_G: G's at eaxh (x,y,z) coordinate.

3— Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4— Outputs_Velocity: Velocity at each point on the rollercoaster.


% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


%}

%% Define constants


```

```
g = 9.81;

%% accelerations

ax = g * sind ( RampAngle ) ;
ay = 0 ;

AccelerationFinal = ax;

%% velocity: since acceleration is constant, we can go ahead and use kinematick equations.

% get final velocity based on height.
syms h
v(h) = sqrt ( 2 * g * (RollerHeight - h)) ;



%% position

%{ we can also get it from the height, which is inputted by the user.

RampHeight = Length * sind ( RampAngle );
RampWidth = Length * cosd ( RampAngle );

%evaluate veloc function defined above.
VelocFinal = double(v(y0 - RampHeight));

% store as output
PositionFinalX = x0 + RampWidth;
PositionFinalY = y0 - RampHeight;


%% time spent: since acceleration is constant, we can go ahead and use kinematic equations.

% here all velocity will be in the x direction, thus we can get time from
% kinematic equations that uses velocity and acceler will be used.

% ( ( v - v0 ) / a0 ) + t0 = t , where all of these in x since ay is 0.
% Again the coordinate system is tilted, but the time should be the same.


TimeElapsed =  ( (VelocFinal - v0) / ax ) + t0 ;



%% G's felt.

G =  cosd(RampAngle) ;
fprintf('The ramp down generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
fprintf(' G, forward and upward. \n ' );

%% write outputs

% All G's here are forward.
Outputs = [ PositionFinalX ; PositionFinalY ; z0 ; G  ] ;
```

```
Outputs_G = [G];
Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
Outputs_Velocity = [ VelocFinal ] ;
```

ParabolaiHill.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity Arclength ] = ParabolaicHill(t0, y0, x0, z0, theta, a0, v)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module the
Parabolaic Hill. This segment follows the free-fall conditions, and hence
the G's will be == 0 ;

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:

Inputs:

1-  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.

2- y0: initial y position.

3- x0: initial x position.

4- z0: initial y position.

5- theta: angel of initial object @ the beginning, just like lunching an
object from a cannon.

6- a0: inital accelration.

7- v: initial velocity.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Outputs:

1- TimeElapsed : time just spent on this segment.

2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

```

```
4— Outputs_Velocity: Velocity at each point on the rollercoaster.

5— Arclength : which is equivalent to the total distance covered.


% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


%}


g = 9.81;

vx = v*cosd(theta);

% time, all equations will be depnding on it.
vy = @(t) -g*t + v*sind(theta);
y = @(t) -(1/2)* g * t^2 + v*sind(theta) * t + y0 ;
x = @(t) vx*t + x0;


% place holders

CurrentX = [];
CurrentY = [];
CurrentZ = [];
CurrentVy = [];
CurrentVx = [];


% there's no way we will start from height == 0, so we will make it as
% inital condition just to start the loop.

y_loop = y0+1 ;
i = 0 ; % time counter
TimeElapsed = i;
Outputs = [ CurrentX ; CurrentY ; CurrentVy ; CurrentVx ] ;

while y0<=y_loop

    CurrentX = [ CurrentX ; x(i) ] ;
    CurrentY = [ CurrentY ; y(i) ] ;
    CurrentZ = [ CurrentZ ; z0 ];
    CurrentVy = [ CurrentVy ; vy(i) ] ;
    CurrentVx = [ CurrentVx ; vx ] ;
    TimeElapsed = [ TimeElapsed ; i ] ;
    y_loop = double(subs(y,i));
        i = i + 0.01;




end

%Calculate the arc length of the parabola
```

```
b = CurrentX(end) - x0;
a = max(CurrentY) - y0;
s = sqrt(b^2+(16*a^2));
Arclength = (s/2) + (b^2/(8*a)) * log(((4*a)+s)/b);
%% write outputs
Arclength = [Arclength];
    Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ] ;
    Outputs_Velocity = [ CurrentVy' ; CurrentVx' ] ;
    Outputs_G = [ zeros(1,length(CurrentX)) ] ;
    G = zeros(1,length(CurrentX));
fprintf('The parabolic hill generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
fprintf(' G, forward and upward. \n ' );


end
```

RampDownOpposite.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ] = RampDownOpposite(t0, RampAngle, Length, y0, x0, z0)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to module a
ramp down, where the user will determine the ramp specifications, and
initial conditions, and the function will return the end status. This is a
version of the ramp the goes down in a positive slope rather than negative.
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

note: here all x and y is tilted, so that +x is pointing with the heading
vector, i.e where the rider is looking.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:

1-  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.

2- RampAngle: relative to horizon, in degrees.

3- Length: hypotenuse of the ramp.

4- y0: initial y position.

5- x0: initial x position.

6- z0: initial z position.
```

```
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Outputs:

1- TimeElapsed : time just spent on this segment.

2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4- Outputs_Velocity: Velocity at each point on the rollercoaster.


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


%}

%% Define constants


g = 9.81;

%% accelerations

ax = g * sind ( RampAngle ) ;
ay = 0 ;

AccelerationFinal = ax;

%% velocity: since acceleration is constant, we can go ahead and use kinematick equations.

% get final velocity based on height.
syms h
v(h) = sqrt ( 2 * g * (125 - h)) ;



%% position

%{ we can also get it from the height, which is inputted by the user.

RampHeight = Length * sind ( RampAngle );
RampWidth = Length * cosd ( RampAngle );

%evaluate veloc function defined above.
VelocFinal = double(v(y0 - RampHeight));

% store as output
PositionFinalX = x0 - RampWidth;
PositionFinalY = y0 - RampHeight;
```

```
%% time spent: since acceleration is constant, we can go ahead and use kinematic equations.

% here all velocity will be in the x direction, thus we can get time from
% kinematic equations that uses velocity and acceler will be used.

% ( ( v - v0 ) / a0 ) + t0 = t , where all of these in x since ay is 0.
% Again the coordinate system is tilted, but the time should be the same.


TimeElapsed = [];



%% G's felt.

G =  cosd(RampAngle) ;
fprintf('The second ramp generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
fprintf(' G, forward and backward. \n ' );

%% write outputs

% All G's here are forward.
Outputs = [ PositionFinalX ; PositionFinalY ; z0 ; G  ] ;

Outputs_G = [G];
Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
Outputs_Velocity = [ VelocFinal ] ;
```

TransitionfromBankedTurn.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
    Transition_fromBankedTurn(t0,x0, y0, z0,theta, r)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:

2-  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.


6- x0: initial x position.
```

```
5- y0: initial y position.

7- z0: initial z position.

3- theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

4- r: Raduis of the circle.


8- RollerHeight: Roller coaster maximum height.

% --------------------------------------


Outputs:

1- TimeElapsed : time just spent on this segment.

2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4- Outputs_Velocity: Velocity at each point on the rollercoaster.

5- ArcLength: which is equivalent to the total distance covered

% --------------------------------------


%}
g = 9.81;

%% begin calculations of position

ArcLength = theta * pi / 180 * r;

ThetaRange = (180 : (180 + theta))';

% get location
CurrentY = y0 - r - r * cosd(ThetaRange); %height as a function of theta

CurrentX = x0 + r * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);

CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height

Normal = g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m

G = Normal/g;

fprintf('The transition out of the banked turn generates a maximum magnitude of: %6.2f %12.8
    f \n ', abs(max(G)))
```

```
fprintf(' G, forward and backward. \n ' );
%% write output

TimeElapsed = [];
ArcLength =[ArcLength];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;

end
```

TransitionfromRampDown.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength ] =
    Transition_fromRampDown(t0,x0, y0, z0,theta, r)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:

1-  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.


2- x0: initial x position.

3- y0: initial y position.

4- z0: initial z position.

5- theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

6- r: Raduis of the circle.


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Outputs:

1- TimeElapsed : time just spent on this segment.
```

```
2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4- Outputs_Velocity: Velocity at each point on the rollercoaster.

5- ArcLength: which is equivalent to the total distance covered

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


%}

g = 9.81;

%% begin calculations of position

ArcLength = theta * pi / 180 * r;

ThetaRange = (360 - theta : 360)';

CurrentY = y0 + r * cosd(360 - theta) - r * cosd(ThetaRange); %height as a function of theta

CurrentX = x0 - r * sind(360 - theta) + r * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);

CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height

Normal = - g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m

G = Normal/g;

fprintf('The transition out of the ramp generates a maximum magnitude of: %6.2f %12.8f \n ',
    abs(max(G)))
fprintf(' G, forward and upward. \n ' );

%% write output

TimeElapsed = [];
ArcLength = [ArcLength];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;

end
```

Transitioninto.m

```
function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] = Transition_into(t0
    ,x0, y0, z0,theta, Raduis)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{
```

```
Done by:
— Abdullah AlMugirun
— Mohamed Aichiouene
— Aufa Amirullah
— Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta. This's a transition into something, something above you.

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
Inputs:

1—  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.


2— x0: initial x position.

3— y0: initial y position.

4— z0: initial z position.

5— theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

6— Raduis: Raduis of the circle.



% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


Outputs:

1— TimeElapsed : time just spent on this segment.

2— Outputs_G: G's at eaxh (x,y,z) coordinate.

3— Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4— Outputs_Velocity: Velocity at each point on the rollercoaster.

5— ArcLength: which is equivalent to the total distance covered

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


%}

%% Gravity
g = 9.81;
%% Tangent normal coordinate CurrentYstem
```

```matlab

ArcLength = theta * pi / 180 * Raduis;

ThetaRange = (0:(theta))';

% Location
CurrentY = y0 + (Raduis - Raduis * cosd(ThetaRange));

CurrentX = x0 + Raduis * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);


CurrentV = sqrt(2 * g * (125 - CurrentY));

Normal = g * cosd(ThetaRange) + CurrentV.^2/Raduis; % Normal force / m

G = Normal/g; % G force

fprintf('The transition into the parabolic hill generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
fprintf(' G, forward and upward. \n ' );

%% write output
ArcLength = [ArcLength];
TimeElapsed = [];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;


end
```

Transitionout.m

```matlab
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength ] = Transition_out(t0,x0, y0, z0,theta, r)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta. This's transition out of something, to something below you.

% -------------------------------------
Inputs:

```

```
1-  t0: initial time relative to the whole roller coaster when
the cart started going on the ramp.


2- x0: initial x position.

3- y0: initial y position.

4- z0: initial z position.

5- theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

6- r: Raduis of the circle.


% -------------------------------------


Outputs:

1- TimeElapsed : time just spent on this segment.

2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4- Outputs_Velocity: Velocity at each point on the rollercoaster.

5- ArcLength: which is equivalent to the total distance covered

% -------------------------------------


%}
g = 9.81;

%% begin calculations of position

ArcLength = theta * pi / 180 * r;

ThetaRange = (360 - theta : 360)';

% get location
CurrentY = y0 + r * cosd(360 - theta) - r * cosd(ThetaRange); %height as a function of theta

CurrentX = x0 - r * sind(360 - theta) + r * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);

CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height

Normal = g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m

G = Normal/g;
```

```
fprintf('The transition out of the parabolic hill generates a maximum magnitude of: %6.2f
    %12.8f \n ', abs(max(G)))
fprintf(' G, forward and backward. \n ' );

%% write output
ArcLength = [ArcLength];
TimeElapsed = [];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;

end
```

TransitiontoGround.m

```
function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
    Transition_toGround(x0, y0, z0,theta, radius)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
%
%{

Done by:
- Abdullah AlMugirun
- Mohamed Aichiouene
- Aufa Amirullah
- Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta.

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Inputs:


1- x0: initial x position.

2- y0: initial y position.

3- z0: initial z position.

4- theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

5- r: Raduis of the circle.


% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Outputs:

1- TimeElapsed : time just spent on this segment.
```

```
2- Outputs_G: G's at eaxh (x,y,z) coordinate.

3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4- Outputs_Velocity: Velocity at each point on the rollercoaster.

5- ArcLength: which is equivalent to the total distance covered

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


%}

g = 9.81;

%% begin calculations of position

ArcLength = theta * pi / 180 * radius;

ThetaRange = (theta:-1:0)';

% get location
CurrentY = y0 + radius * cosd(theta) - radius * cosd(ThetaRange); %height as a function of
    theta

CurrentX = x0 - radius * sind(theta) + radius * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);

CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height

Normal = g * cosd(ThetaRange) + CurrentV.^2/radius; % Normal force / m

G = Normal/g;

fprintf('The transition to ground generates a maximum magnitude of: %6.2f %12.8f \n ', abs(
    max(G)))
fprintf(' G, forward and backward. \n ' );

%% write output

TimeElapsed = [];
ArcLength = [ArcLength];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;

end
```

TransitionToDownRamp.m

```
function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
    TransitionToDownRamp(r,theta,x0,y0,z0)
%
% ASEN 2003: Dynamics, Lab 1, Roller Coaster
```

```
%
%{

Done by:
— Abdullah AlMugirun
— Mohamed Aichiouene
— Aufa Amirullah
— Abdulla AlAmeri

This function is one segment of a roller coaster, it attempts to
module a transition from a segment. The segment is in the name of the
function. Most transitions are assumed to be circles that run to a certain
theta.

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —
Inputs:

1— r: Raduis of the circle.

2— theta: imagine the transition is a circle, starting from 0, how many degrees
should it run too? That's your theta.

3— x0: initial x position.

4— y0: initial y position.

5— z0: initial z position.


% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


Outputs:

1— TimeElapsed : time just spent on this segment.

2— Outputs_G: G's at eaxh (x,y,z) coordinate.

3— Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).

4— Outputs_Velocity: Velocity at each point on the rollercoaster.

5— ArcLength: which is equivalent to the total distance covered

% — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —


%}

g = 9.81;

%% begin calculations of position

ArcLength = [theta * pi / 180 * r];

```

```
ThetaRange = (180 : (theta + 180))';

CurrentY = y0 - r - r * cosd(ThetaRange); %height as a function of theta

CurrentX = x0 - r * sind(ThetaRange);

CurrentZ = z0 * ones(length(ThetaRange),1);

CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height

Normal = - g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m

G = Normal/g;

fprintf('The transition to the ramp generates a maximum magnitude of: %6.2f %12.8f \n ', abs
    (max(G)))
fprintf(' G, forward and upward. \n ' );


%% write outputs

% All G's here are forward.
ArcLength = [ArcLength];
TimeElapsed = [];
Outputs_G = [ G' ] ;
Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
Outputs_Velocity = [ CurrentV' ] ;
end
```

colorline3.m

```
function h = color_line3(x, y, z, c, varargin)
% color_line3 plots a 3-D line with c-data as color
%
%       h = color_line(x, y, z, c)
%       by default: 'LineStyle','-' and 'Marker','none'
%
%          or
%       h = color_line(x, y, z, c, mark)
%          or
%       h = color_line(x, y, z, c, 'Property','value'...)
%             with valid 'Property','value' pairs for a surface object
%
%  in:  x      x-data
%       y      y-data
%       z      z-data
%       c      4th dimension for colouring
%       mark   for scatter plots with no connecting line
%
% out:  h   handle of the surface object


h = surface(...
  'XData',[x(:) x(:)],...
  'YData',[y(:) y(:)],...
  'ZData',[z(:) z(:)],...
```

```
  'CData',[c(:) c(:)],...
  'FaceColor','interp',...
  'EdgeColor','interp',...
  'Marker','none');

if nargin ==5
    switch varargin{1}
        case {'+' 'o' '*' '.' 'x' 'square' 'diamond' 'v' '^' '>' '<' 'pentagram' 'p' '
            hexagram' 'h'}
            set(h,'LineStyle','none','Marker',varargin{1})
        otherwise
            error(['Invalid marker: ' varargin{1}])
    end

elseif nargin > 5
    set(h,varargin{:})
end
```