

# 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

$\alpha$	Theta values initialized for curves
$\rho$	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/s <sup>2</sup> 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.

---

\*SID: 105272937

†SID: 105599913

‡SID: 109364560:

§SID:107500341

# 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 curvilinear, 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.

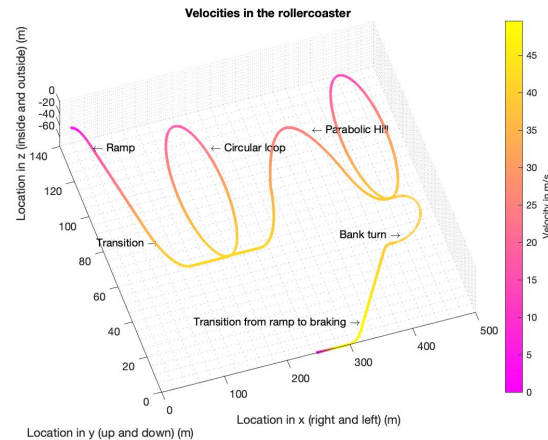


Figure 1: A 3D-plot of the roller coaster track with velocity, color coded. It is important to notice that the loops are perfectly rounded 3D, except that the scale of the x, y and z-axis are not consistent.

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 sitting 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.

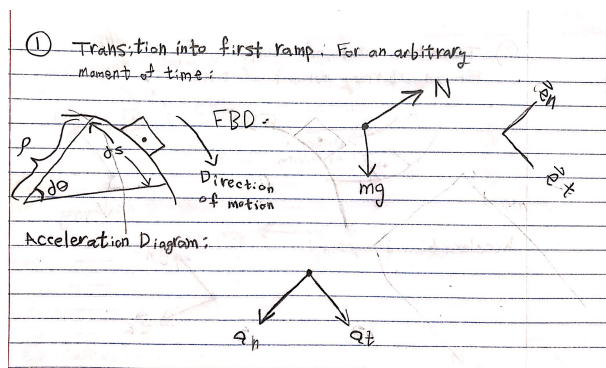


Figure 2: Free Body Diagram of Transition Section

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.

$$X = x_{trans} - R * \sin(\alpha) \quad (1)$$

$$Y = y_{trans} - R - R * \cos(\alpha) \quad (2)$$

$$Z = z_{trans} * \cos(\alpha) \quad (3)$$

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.

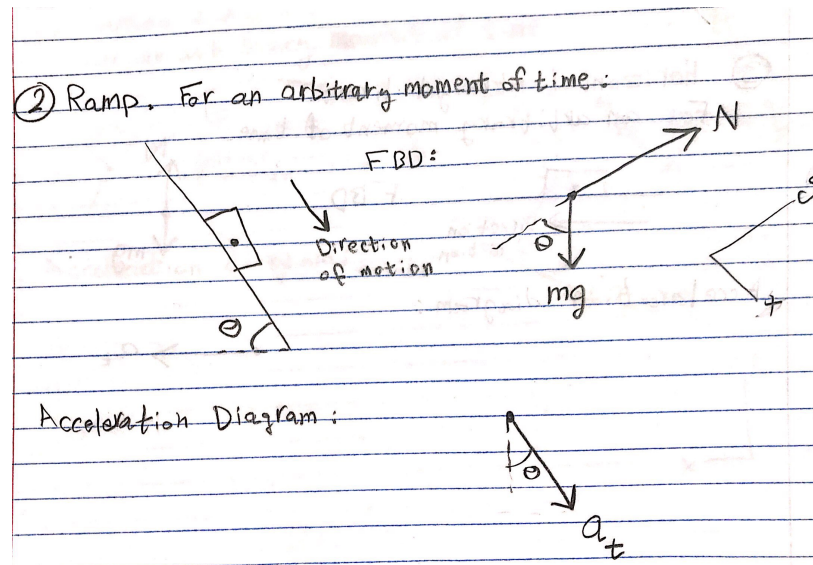


Figure 3: Free Body Diagram of Ramps Section

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:

$$y_{ramp} = L_{ramp} * \sin(\alpha) \quad (4)$$

$$x_{ramp} = L_{ramp} * \cos(\alpha) \quad (5)$$

$$X = x_0 - x_{ramp} \quad (6)$$

$$Y = y_0 - y_{ramp} \quad (7)$$

### 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.

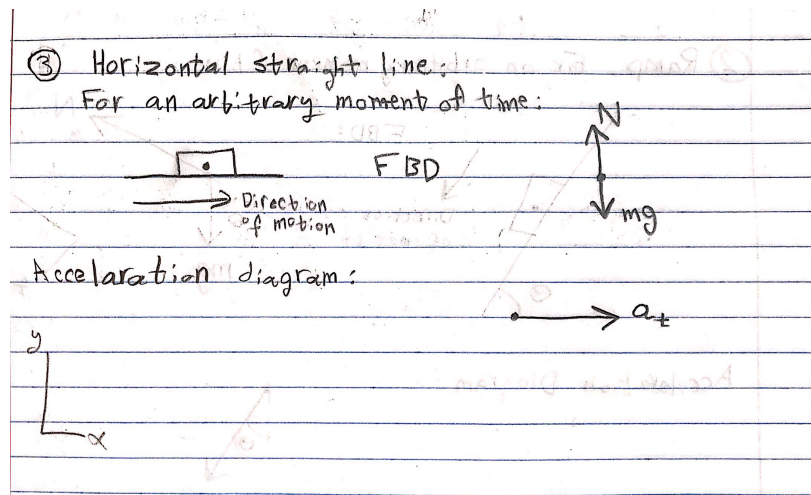


Figure 4: Free Body Diagram of Horizontal Straight Lines

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

$$X = x_0 + x_{line} + D \quad (8)$$

$$Y = y_0 + y_{line} \quad (9)$$

$$Z = z_0 + z_{line} \quad (10)$$

#### 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.

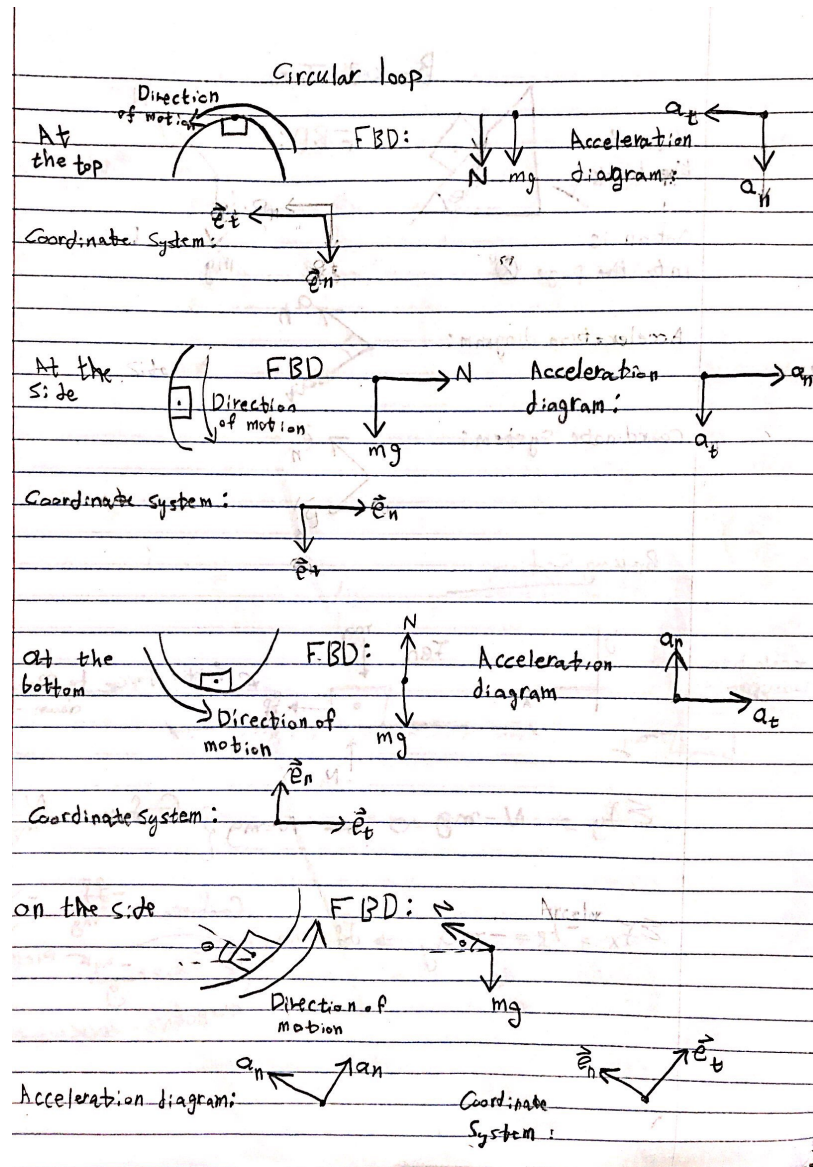


Figure 5: Free Body Diagrams of Loops Section

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:

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

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

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

## 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.

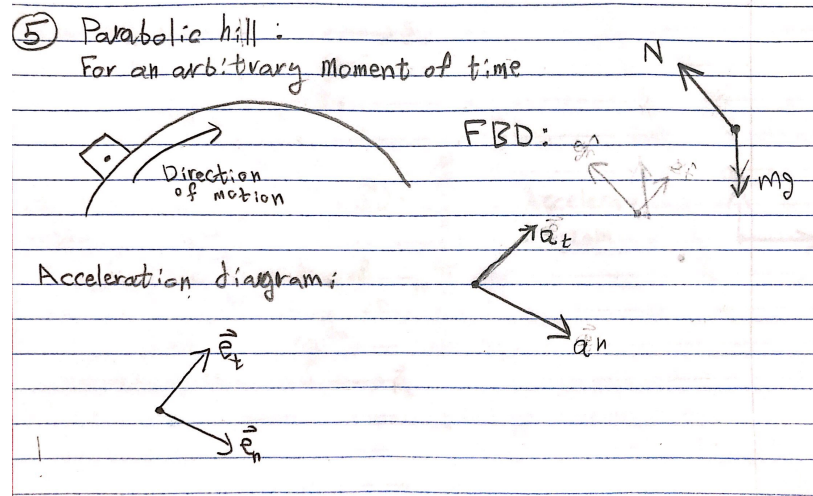


Figure 6: Free Body Diagram of Parabolic Hill

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:

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

$$x_{parabola} = x_0 + v_x * t \quad (15)$$

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

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

$$a = y_{parabola,f} - y_{parabola,i} \quad (17)$$

$$b = x_{parabola,f} - x_{parabola,i} \quad (18)$$

$$s = \sqrt{b^2 + 16a^2} \quad (19)$$

## 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.



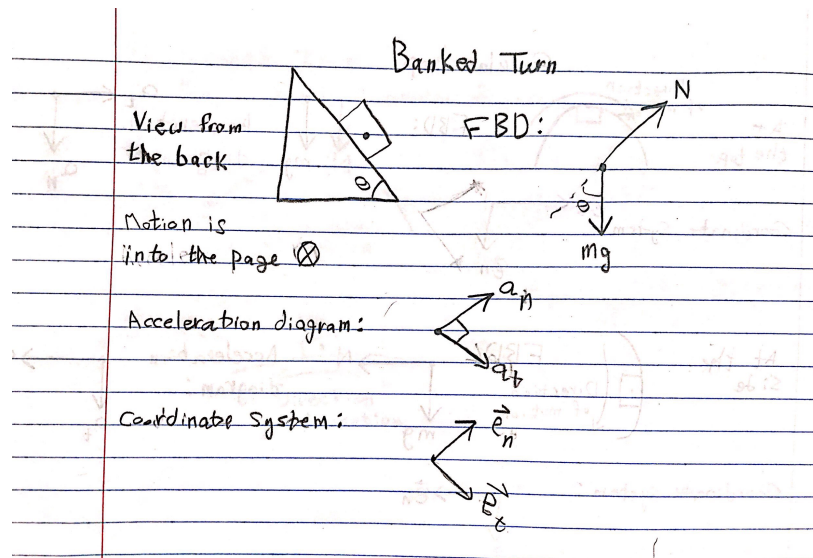


Figure 7: Free Body Diagram of Banked Turn

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:

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

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

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.

$$Arclength = r * \theta \quad (22)$$

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  $\theta$  is from 0 degrees to 180°.

$$z(\theta) = z0 - r + r * \cos(\theta) \quad (23)$$

$$x(\theta) = x0 + r * \sin(\theta) \quad (24)$$



## G. Breaking Segment

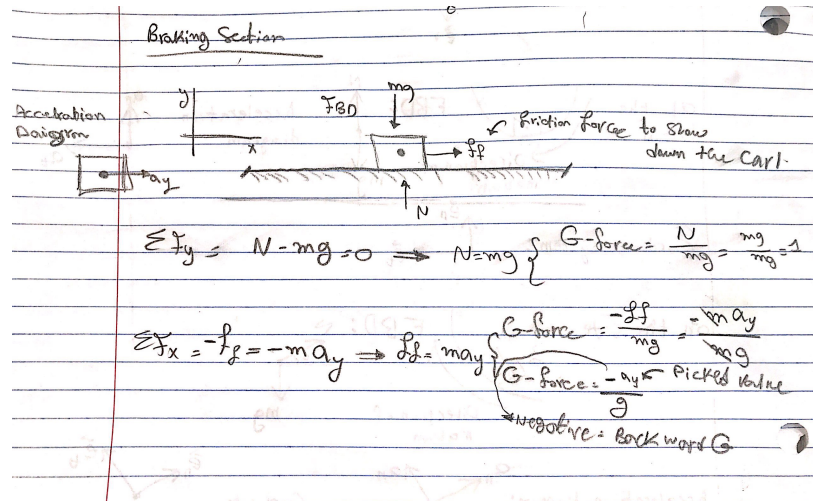


Figure 8: Free Body Diagram of Ramps Section

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

$$G - \text{force} = \frac{a_{\text{picked}}}{g} \quad (25)$$

To maximize the G experienced, G was chosen to be 3.9, which in return would require a constant deceleration 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 coaster 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 that our acceleration is constant, which makes the kinematic equations for constant acceleration useful. The following equation:

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

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

## H. G Force

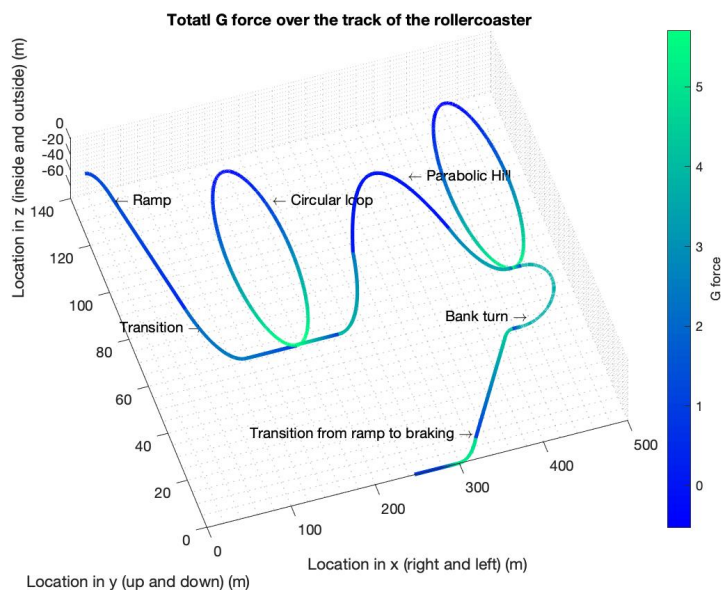


Figure 9: G force modeled in the roller-coaster

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	$ G_{max} $
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

<sup>1</sup>James, Stewart *Essential Calculus, Second Edition*, James Stewart, 2007

<sup>2</sup>Axelrad, P., ASEN 2003 Lab 1, Canvas, Aerospace Engineering Sciences Department, ASEN 2003, University of Colorado at Boulder, Spring 2019

<sup>3</sup>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 Aiciouene	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

```
1 %% info
2 % comments needs to be added.
3 %% housekeeping
4
5 clear;
6 clc;
7 close all;
8
9 %% define constants
10
11 % define all initial conditions for the first segment
12 g = 9.81;
13 h0 = 125 ; %initial height in meters
14
15 % velocity function
16 syms h
17 v(h) = sqrt ( 2 * g * (h0 - h)) ;
18
19 Vi = double(v(h0)); %initial velocity
20 Ti = 0; %time in s
21 Xi = 0; % in m
22 Yi = h0; % in m
23 Zi = 0; % in m
24 a0x = 0;
25 angle = 60;
26 r = 20;
27 t0 = 0;
28
```

```

29 [TimeNew GNew LocaNew VelocNew DistanceCovered] = TransitionToDownRamp(r,angle,Xi,Yi,Zi);
30
31 % concatenate:
32 TotalDistanceCovered = DistanceCovered;
33 G = cat(1,1,GNew');
34 xPosit = cat(1,0,LocaNew(1,:));
35 yPosit = cat(1,h0,LocaNew(2,:));
36 zPosit = cat(1,0,LocaNew(3,:));
37 xVeloc = cat(1,Vi,VelocNew');
38 %t = cat(1,0,TimeNew(1));
39
40 % get coordinate of a point in the middle of the segment so it can be
41 % named on the plot at that point.
42
43 Coordinate_i = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
44 /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
45
46 %% ramp down
47
48 Ramptheta = 60 ;
49 Length2 = 60;
50 x0 = xPosit(length(xPosit));
51 y0 = yPosit(length(yPosit));
52 z0 = zPosit(length(zPosit));
53 t0 = 0;
54
55 [ TimeNew GNew LocaNew VelocNew] = RampDown(Vi,0, Ramptheta,Length2, y0, x0, z0, h0 );
56 G = cat(1,G,GNew');
57 TotalDistanceCovered = cat(1,TotalDistanceCovered,Length2);
58 xPosit = cat(1,xPosit,LocaNew(1,:));
59 yPosit = cat(1,yPosit,LocaNew(2,:));
60 zPosit = cat(1,zPosit,LocaNew(3,:));
61 xVeloc = cat(1,xVeloc,VelocNew');
62
63 % get coordinate of a point in the middle of the segment so it can be
64 % named on the plot at that point.
65
66 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
67 /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
68 Coordinate = cat(3,Coordinate_i,CoordinateNew);
69
70 %t = cat(1,t,TimeNew);
71 %% transition off rampCurvature
72
73 Curvature = 50; % Chosen arbitrary.
74
75 x0 = xPosit(length(xPosit));
76 y0 = yPosit(length(yPosit));
77 z0 = zPosit(length(zPosit));
78 t0 = 0;
79
80 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_fromRampDown(t0,x0, y0, z0,
Ramptheta,Curvature);

```

```

81
82     G = cat(1,G,GNew');
83     TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
84     xPosit = cat(1,xPosit,LocaNew(1,:));
85     yPosit = cat(1,yPosit,LocaNew(2,:));
86     zPosit = cat(1,zPosit,LocaNew(3,:));
87     xVeloc = cat(1,xVeloc,VelocNew');
88
89     % get coordinate of a point in the middle of the segment so it can be
90     % named on the plot at that point.
91
92     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
93         /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
94     Coordinate = cat(3,Coordinate,CoordinateNew);
95
96     % t = cat(1,t,TimeNew);
97
98     %% Linear section: transition, For 5 meters
99
100     distance = 60; %Linear distance
101
102     x0 = xPosit(length(xPosit));
103     y0 = yPosit(length(yPosit));
104     z0 = zPosit(length(zPosit));
105     t0 = 0;
106
107     GNew = 1;
108     G = cat(1,G,GNew');
109     TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
110     xPosit = cat(1,xPosit,x0+distance);
111     yPosit = cat(1,yPosit,y0);
112     zPosit = cat(1,zPosit,z0);
113     xVeloc = cat(1,xVeloc,VelocNew');
114
115     % get coordinate of a point in the middle of the segment so it can be
116     % named on the plot at that point.
117
118     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
119         /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
120     Coordinate = cat(3,Coordinate,CoordinateNew);
121
122     % t = cat(1,t,TimeNew);
123
124     %% circular loop:
125
126     % initial info.
127
128     x0 = xPosit(length(xPosit));
129     y0 = yPosit(length(yPosit));
130     z0 = zPosit(length(zPosit));
131     r = 37; %loop Raduis
132     t=0;
133     [ TimeNew GNew LocaNew VelocNew DistanceCovered ] = CircularLoop(xVeloc, t, r, x0, y0, z0,
134         h0);
135
136     G = cat(1,G,GNew');

```

```

133 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
134 xPosit = cat(1,xPosit,LocaNew(1,:));
135 yPosit = cat(1,yPosit,LocaNew(2,:));
136 zPosit = cat(1,zPosit,LocaNew(3,:));
137 xVeloc = cat(1,xVeloc,VelocNew');
138
139 % get coordinate of a point in the middle of the segment so it can be
140 % named on the plot at that point.
141
142 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
143 Coordinate = cat(3,Coordinate,CoordinateNew);
144
145 % t = cat(1,t,TimeNew);
146
147 %% Linear section: transition, For 5 meters
148
149 distance = 50; %Linear distance
150
151 x0 = xPosit(length(xPosit));
152 y0 = yPosit(length(yPosit));
153 z0 = zPosit(length(zPosit));
154 t0 = 0;
155
156 GNew = 1;
157 G = cat(1,G,GNew');
158 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
159 xPosit = cat(1,xPosit,x0+distance);
160 yPosit = cat(1,yPosit,y0);
161 zPosit = cat(1,zPosit,z0);
162 xVeloc = cat(1,xVeloc,VelocNew');
163
164 % get coordinate of a point in the middle of the segment so it can be
165 % named on the plot at that point.
166
167 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
168 Coordinate = cat(3,Coordinate,CoordinateNew);
169
170 % t = cat(1,t,TimeNew);
171
172 %% transition into hill
173
174 Curvture = 60; % Chooosen arbitrary.
175
176 x0 = xPosit(length(xPosit));
177 y0 = yPosit(length(yPosit));
178 z0 = zPosit(length(zPosit));
179 t0 = 0;
180
181 [ TimeNew GNew LocaNew VelocNew DistanceCovered ] = Transition_into(t0,x0, y0, z0,Rampheta
    ,Curvture);
182
183
184 G = cat(1,G,GNew');

```



```

185     TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
186     xPosit = cat(1,xPosit,LocaNew(1,:));
187     yPosit = cat(1,yPosit,LocaNew(2,:));
188     zPosit = cat(1,zPosit,LocaNew(3,:));
189     xVeloc = cat(1,xVeloc,VelocNew');
190
191     % get coordinate of a point in the middle of the segment so it can be
192     % named on the plot at that point.
193
194     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
195         /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
196     Coordinate = cat(3,Coordinate,CoordinateNew);
197
198     %t = cat(1,t,TimeNew);
199
200     %% parabolic hill:
201
202
203     x0 = xPosit(length(xPosit));
204     y0 = yPosit(length(yPosit));
205     z0 = zPosit(length(zPosit));
206     theta = 45;
207     a0 = 0;
208     v = double(v(y0));
209     t0 = 0;
210
211     [ TimeNew GNew LocaNew VelocNew DistanceCovered] = ParabolaicHill(t0, y0, x0, z0, theta, a0
212         , v);
213
214     G = cat(1,G,GNew');
215     TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
216     xPosit = cat(1,xPosit,LocaNew(1,:));
217     yPosit = cat(1,yPosit,LocaNew(2,:));
218     zPosit = cat(1,zPosit,LocaNew(3,:));
219     xVeloc = cat(1,xVeloc,VelocNew(1,:));
220
221     % get coordinate of a point in the middle of the segment so it can be
222     % named on the plot at that point.
223
224     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
225         /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
226     Coordinate = cat(3,Coordinate,CoordinateNew);
227
228     %t = cat(1,t,TimeNew);
229
230     %% transition off the hill.
231
232
233     x0 = xPosit(length(xPosit));
234     y0 = yPosit(length(yPosit));
235     z0 = zPosit(length(zPosit));
236     t0 = 0;
237     Ramptheta = 50;
238
239     [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_out(t0,x0, y0, z0,Ramptheta,

```

```

    Curvture);
237
238
239     G = cat(1,G,GNew');
240     TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
241     xPosit = cat(1,xPosit,LocaNew(1,:));
242     yPosit = cat(1,yPosit,LocaNew(2,:));
243     zPosit = cat(1,zPosit,LocaNew(3,:));
244     xVeloc = cat(1,xVeloc,VelocNew');
245
246     % get coordinate of a point in the middle of the segment so it can be
247     % named on the plot at that point.
248
249     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
        /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
250     Coordinate = cat(3,Coordinate,CoordinateNew);
251
252     % t = cat(1,t,TimeNew);
253
254
255     %% Linear section: transition, For 5 meters
256
257     distance = 5; %Linear distance
258
259     x0 = xPosit(length(xPosit));
260     y0 = yPosit(length(yPosit));
261     z0 = zPosit(length(zPosit));
262     t0 = 0;
263
264     GNew = 1;
265     G = cat(1,G,GNew');
266     TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
267     xPosit = cat(1,xPosit,x0+distance);
268     yPosit = cat(1,yPosit,y0);
269     zPosit = cat(1,zPosit,z0);
270     xVeloc = cat(1,xVeloc,VelocNew');
271
272     % get coordinate of a point in the middle of the segment so it can be
273     % named on the plot at that point.
274
275     CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
        /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
276     Coordinate = cat(3,Coordinate,CoordinateNew);
277
278     % t = cat(1,t,TimeNew);
279
280     %% circular loop:
281
282     % initial info.
283
284     x0 = xPosit(length(xPosit));
285     y0 = yPosit(length(yPosit));
286     z0 = zPosit(length(zPosit));
287     r = 35; %loop Raduis
288

```

```

289 % get coordinate of a point in the middle of the segment so it can be
290 % named on the plot at that point.
291
292 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
293 Coordinate = cat(3,Coordinate,CoordinateNew);
294
295
296 %%
297 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = CircularLoop(xVeloc, t, r, x0, y0, z0,h0
    );
298
299
300 G = cat(1,G,GNew');
301 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
302 xPosit = cat(1,xPosit,LocaNew(1,:));
303 yPosit = cat(1,yPosit,LocaNew(2,:));
304 zPosit = cat(1,zPosit,LocaNew(3,:));
305 xVeloc = cat(1,xVeloc,VelocNew');
306
307 % get coordinate of a point in the middle of the segment so it can be
308 % named on the plot at that point.
309
310 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
311 Coordinate = cat(3,Coordinate,CoordinateNew);
312
313 %t = cat(1,Time,TimeNew);
314 %% Linear section: transition, For 5 meters
315
316 distance = 10; %Linear distance
317
318 x0 = xPosit(length(xPosit));
319 y0 = yPosit(length(yPosit));
320 z0 = zPosit(length(zPosit));
321 t0 = 0;
322
323 GNew = 1;
324 G = cat(1,G,GNew');
325 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
326 xPosit = cat(1,xPosit,x0+distance);
327 yPosit = cat(1,yPosit,y0);
328 zPosit = cat(1,zPosit,z0);
329 xVeloc = cat(1,xVeloc,VelocNew');
330
331 % get coordinate of a point in the middle of the segment so it can be
332 % named on the plot at that point.
333
334 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
335 Coordinate = cat(3,Coordinate,CoordinateNew);
336
337 % t = cat(1,t,TimeNew);
338
339

```

```

340
341 %% Banked Turn
342
343 % initial info.
344 x0 = xPosit(length(xPosit));
345 y0 = yPosit(length(yPosit));
346 z0 = zPosit(length(zPosit));
347 r = 39; %loop Raduis
348 BankAngle = 50;
349
350 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = BankTurn(BankAngle, t, r, x0, y0, z0,h0)
351 ;
352 G = cat(1,G,GNew(1,:));
353 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
354 xPosit = cat(1,xPosit,double(LocaNew(1,:)));
355 yPosit = cat(1,yPosit,double(LocaNew(2,:)));
356 zPosit = cat(1,zPosit,double(LocaNew(3,:)));
357 xVeloc = cat(1,xVeloc,VelocNew);
358
359 % get coordinate of a point in the middle of the segment so it can be
360 % named on the plot at that point.
361
362 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
363 /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
364
365 %% Linear section: transition, For 5 meters
366
367 distance = 10; %Linear distance
368
369 x0 = xPosit(length(xPosit));
370 y0 = yPosit(length(yPosit));
371 z0 = zPosit(length(zPosit));
372 t0 = 0;
373
374 GNew = 1;
375 G = cat(1,G,GNew);
376 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
377 xPosit = cat(1,xPosit,x0-distance);
378 yPosit = cat(1,yPosit,y0);
379 zPosit = cat(1,zPosit,z0);
380 xVeloc = cat(1,xVeloc,VelocNew);
381
382 % get coordinate of a point in the middle of the segment so it can be
383 % named on the plot at that point.
384
385 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
386 /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
387
388
389 % t = cat(1,t,TimeNew);
390
391 %% ramp down but in opposite direction to the other ramp

```

```

392 %% ramp down
393
394 Curvature = 30; % Chosen arbitrary.
395 Ramptheta = 20;
396 x0 = xPosit(length(xPosit));
397 y0 = yPosit(length(yPosit));
398 z0 = zPosit(length(zPosit));
399 t0 = 0;
400
401 [ TimeNew GNew LocaNew VelocNew DistanceCovered ] = Transition_fromBankedTurn(t0,x0, y0, z0
    ,Ramptheta,Curvature);
402
403 G = cat(1,G,GNew');
404 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
405 xPosit = cat(1,xPosit,LocaNew(1,:));
406 yPosit = cat(1,yPosit,LocaNew(2,:));
407 zPosit = cat(1,zPosit,LocaNew(3,:));
408 xVeloc = cat(1,xVeloc,VelocNew');
409 % t = cat(1,t,TimeNew);
410
411
412 x0 = xPosit(length(xPosit));
413 y0 = yPosit(length(yPosit));
414 z0 = zPosit(length(zPosit));
415 t0 = 0;
416 Ramptheta = 25.1 ;
417 length1 = 86.6;
418 [ TimeNew GNew LocaNew VelocNew] = RampDownOpposite(t0,Ramptheta,length1,y0, x0, z0 );
419
420 G = cat(1,G,GNew');
421 TotalDistanceCovered = cat(1,TotalDistanceCovered,length1);
422 xPosit = cat(1,xPosit,double(LocaNew(1,:)));
423 yPosit = cat(1,yPosit,double(LocaNew(2,:)));
424 zPosit = cat(1,zPosit,double(LocaNew(3,:)));
425 xVeloc = cat(1,xVeloc,VelocNew');
426
427 % get coordinate of a point in the middle of the segment so it can be
428 % named on the plot at that point.
429
430 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
    /2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
431 Coordinate = cat(3,Coordinate,CoordinateNew);
432
433
434
435 %% Transition to ground
436
437 x0 = xPosit(length(xPosit));
438 y0 = yPosit(length(yPosit));
439 z0 = zPosit(length(zPosit));
440 t0 = 0;
441 r = 60;
442 curvature = 30;
443 [ TimeNew GNew LocaNew VelocNew DistanceCovered] = Transition_toGround(x0, y0, z0,curvature
    , r);

```

```

444
445 G = cat(1,G,GNew');
446 TotalDistanceCovered = cat(1,TotalDistanceCovered,DistanceCovered);
447 xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
448 yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
449 zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
450 xVeloc = cat(1,xVeloc,VelocNew');
451
452
453 % get coordinate of a point in the middle of the segment so it can be
454 % named on the plot at that point.
455
456 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];
457 Coordinate = cat(3,Coordinate,CoordinateNew);
458
459 %% Linear section: transition, For 5 meters
460
461 distance = 20; %Linear distance
462
463 x0 = xPosit(length(xPosit));
464 y0 = yPosit(length(yPosit));
465 z0 = zPosit(length(zPosit));
466 t0 = 0;
467
468 GNew = 1;
469 G = cat(1,G,GNew');
470 TotalDistanceCovered = cat(1,TotalDistanceCovered,distance);
471 xPosit = cat(1,xPosit,x0-distance);
472 yPosit = cat(1,yPosit,y0);
473 zPosit = cat(1,zPosit,z0);
474 xVeloc = cat(1,xVeloc,VelocNew');
475 % t = cat(1,t,TimeNew);
476
477
478 %% Braking and stopping
479
480
481 x0 = xPosit(length(xPosit));
482 y0 = yPosit(length(yPosit));
483 z0 = zPosit(length(zPosit));
484 t0 = 0;
485 [TimeNew GNew LocaNew VelocNew] = BreakAndStop(x0,y0,z0);
486
487 G = cat(1,G,GNew');
488 xPosit = cat(1,xPosit,double(LocaNew(1,:)'));
489 yPosit = cat(1,yPosit,double(LocaNew(2,:)'));
490 zPosit = cat(1,zPosit,double(LocaNew(3,:)'));
491 xVeloc = cat(1,xVeloc,VelocNew');
492
493 % get coordinate of a point in the middle of the segment so it can be
494 % named on the plot at that point.
495
496 CoordinateNew = [ LocaNew(1,(floor(length(LocaNew)/2))); LocaNew(2,(floor(length(LocaNew)
/2))) ; LocaNew(3,(floor(length(LocaNew)/2))) ];

```

```

497 Coordinate = cat(3,Coordinate,CoordinateNew);
498
499
500 %% All velocities =
501 syms h
502 v = @(h) sqrt ( 2 * g * (h0 - h)) ;
503
504 AllV = v(yPosit);
505 AllV(end) = 0;
506
507 %% print total distance
508
509 fprintf('\n ');
510 fprintf('Total distance covered: %6.2f %12.8f \n ',sum(TotalDistanceCovered))
511 fprintf(' meters. \n ');
512
513 %% plot
514
515 figure(1);
516
517 color_line3(xPosit,yPosit,zPosit,AllV,'LineWidth',2.5);
518 colormap spring;
519 c = colorbar;
520 c.Label.String = 'Velocity in m/s';
521 grid minor;
522 title('Velocities in the rollercoaster');
523 xlabel('Location in x (right and left) (m)');
524 ylabel('Location in y (up and down) (m)');
525 zlabel('Location in z (inside and outside) (m)');
526 view([-18 80])
527
528 %put texts on the graph to label segments.
529 text(Coordinate(1,1,1)-45,Coordinate(2,1,2),Coordinate(3,1,3),' Transition \rightarrow');
530
531 text(Coordinate(1,1,4)-45,Coordinate(2,1,5),Coordinate(3,1,6),'\leftarrow Ramp');
532
533 text(Coordinate(1,1,7)-40,Coordinate(2,1,8),Coordinate(3,1,9),'\leftarrow Circular loop');
534
535 text(Coordinate(1,1,10)-40,Coordinate(2,1,11)+40,Coordinate(3,1,12),'\leftarrow Parabolic
Hill');
536
537
538 text(Coordinate(1,1,13)-80,Coordinate(2,1,14),Coordinate(3,1,15),'Bank turn \rightarrow');
539
540 text(Coordinate(1,1,16)-240,Coordinate(2,1,17)+30,Coordinate(3,1,18),'Transition from ramp
to braking \rightarrow');
541
542
543 saveas(gcf,'Gforce.jpg')
544
545
546 % name the sections
547
548 % --- !!
549

```



```

550     figure(2);
551
552     colormap winter;
553     color_line3(xPosit,yPosit,zPosit,G,'LineWidth',2.5);
554     c = colorbar;
555     c.Label.String = 'G force';
556     grid minor;
557     title('Totatl G force over the track of the rollercoaster');
558     xlabel('Location in x (right and left) (m)');
559     ylabel('Location in y (up and down) (m)');
560     zlabel('Location in z (inside and outside) (m)');
561     view([-18 80])
562
563     %put texts on the graph to label segments.
564     text(Coordinate(1,1,1)-45,Coordinate(2,1,2),Coordinate(3,1,3),' Transition \rightarrow');
565
566     text(Coordinate(1,1,4)-45,Coordinate(2,1,5),Coordinate(3,1,6),'\leftarrow Ramp');
567
568     text(Coordinate(1,1,7)-40,Coordinate(2,1,8),Coordinate(3,1,9),'\leftarrow Circular loop');
569
570     text(Coordinate(1,1,10)-40,Coordinate(2,1,11)+40,Coordinate(3,1,12),'\leftarrow Parabolic
        Hill');
571
572
573     text(Coordinate(1,1,13)-80,Coordinate(2,1,14),Coordinate(3,1,15),'Bank turn \rightarrow');
574
575     text(Coordinate(1,1,16)-240,Coordinate(2,1,17)+30,Coordinate(3,1,18),'Transition from ramp
        to braking \rightarrow');
576
577     saveas(gcf,'Velocity.jpg')

```

#### BankTurn.m

```

1  function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] = BankTurn(
    Banktheta,t,raduis,x0,y0,z0, RollerCoasterHeight )
2
3  %
4  % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5  %
6  %{
7  Done by:
8  - Abdullah AlMugirun
9  - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to module a
14 a bank turn.
15
16 % -----
17 Inputs:
18
19 1- Banktheta : the angel which the bank it turned at.
20
21 2- t: initial time relative to the whole roller coaster.
22

```

```

23 3- raduis: Raduis of curvature, which is constant here.
24
25 4- y0: initial y position.
26
27 5- x0: initial x position.
28
29 6- z0: initial z position.
30
31 7- RollerHeight: Roller coaster maximum height.
32
33 % -----
34
35
36 Outputs:
37
38 1- TimeElapsed : time just spent on this segment.
39
40 2- Outputs_G: G's at each (x,y,z) coordinate.
41
42 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
43
44 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
45
46 5- ArcLength: distance covered in this segment.
47 % -----
48
49
50 %}
51
52 g = 9.81;
53
54 %% code
55 % Arc length
56 ArcLength = Banktheta * pi / 180 * raduis;
57 % height is fixed, thus velocity is.
58 v = sqrt ( 2 * g * (RollerCoasterHeight - y0)) ;
59
60 % this normal force is divided by m.
61 Normal = (((v^2 )/raduis ) * sind(Banktheta)) + g*cosd(Banktheta) ;
62
63 % Lateral force
64 Lateral = -sind(Banktheta )*( g - (((v^2 )/raduis )*cosd(Banktheta ))) ;
65
66 % Gs felt in the direction normal to ramp
67 Normal_G = Normal / 9.81 ;
68
69 % Gs felt in the direction lateral to ramp
70 Lateral_G = Lateral / 9.81 ;
71
72 %substitute from theta = 0 degrees to 180, the whole bank turn.
73 alphaRange = 0:.01:180 ;
74
75
76 % use the range of the angles in the equations for the position in the x, y
77 % and z

```

```

78 CurrentZ = z0 - raduis + raduis*cosd(alphaRange);
79 CurrentX = x0 + raduis*sind(alphaRange);
80 CurrentY = y0 * ones(length(alphaRange),1) ;
81
82 % write outputs
83 Outputs_G = [ Normal_G*ones(1,length(CurrentX)) ; Lateral_G*ones(1,length(CurrentX)) ];
84 ArcLength = [ArcLength];
85 Outputs_Loc = [ CurrentX ; CurrentY' ; CurrentZ ] ;
86 Outputs_Velocity = [ ones(1,length(alphaRange))'*v ] ;
87 TimeElapsed = [];
88 fprintf('The banked turn generates a maximum magnitude of: %6.2f %12.8f and %6.2f %12.8f\n',
      Normal_G,Lateral_G )
89 fprintf(' G, normal and lateral, respectively. \n ' );
90
91
92
93
94 end

```

#### BreakAndStop.m

```

1 function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity] = BreakAndStop(x0,y0,z0)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
4 %
5 %{
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to module the
14 breaking and stop section.
15
16 % -----
17 Inputs:
18
19 1- x0: initial x position.
20
21 2- y0: initial y position.
22
23 3- z0: initial z position.
24
25 % -----
26
27
28 Outputs:
29
30 1- TimeElapsed : time just spent on this segment.
31
32 2- Outputs_G: G's at each (x,y,z) coordinate.
33
34 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
35

```

```

36 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
37
38
39 % -----
40
41
42 %}
43 %% Define constants
44
45
46 g = 9.81;
47
48
49
50
51
52
53 %% g's
54 upGs = 1;
55 backGs = 3.9;
56 N = sqrt(upGs^2 + backGs^2);
57
58 %% initial velocity
59 VelocFinal = sqrt(2 * g * (125 - y0));
60
61 %% give final position based on velocity
62 PositionFinalX = x0 - VelocFinal^2/(2 * backGs * 9.81);
63 PositionFinalY = y0;
64 PositionFinalZ = z0;
65
66 %track length
67 LengthOfTrack = VelocFinal^2/(2 * backGs * 9.81);
68
69 %% write outputs
70
71 Outputs = [ PositionFinalX ; PositionFinalY ; z0] ;
72 TimeElapsed = [];
73 Outputs_G = [upGs];
74 Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
75 Outputs_Velocity = [ VelocFinal ] ;
76
77 end

```

CircularLoop.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity Circumference] = CircularLoop(
   V0, t0, r, x0, y0, z0,RollerHeight)
2
3 %
4 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5 %
6 %
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah

```

```

11  — Abdulla AlAmeri
12
13  This function is one segment of a roller coaster, it attempts to module the
14  Circular loop with a fixed raduis.
15
16  % -----
17  Inputs:
18
19  Inputs:
20  1- v0 : Initial velocity (Total magnitude).
21
22  2- t0: initial time relative to the whole roller coaster when
23  the cart started going on the ramp.
24
25  3- r: raduis of circle, which is equivalent to raduis of curvature in this case.
26
27  4- x0: initial x position.
28
29  5- y0: initial y position.
30
31  6- z0: initial z position.
32
33  7- RollerHeight: Roller coaster maximum height.
34
35  % -----
36
37
38  Outputs:
39
40  1- TimeElapsed : time just spent on this segment.
41
42  2- Outputs_G: G's at eaxh (x,y,z) coordinate.
43
44  3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
45
46  4- Outputs_Velocity: Velocity at each point on the rollercoaster.
47
48  5- Circumference : which is equivalent to the total distance covered.
49
50
51  % -----
52
53
54  %}
55
56
57
58  %% using Normal tangent coordinate system.
59
60  % we can know our height based on our simple geometry.
61  Circumference = 2 * pi * r;
62  thetaStep = 0:0.1:360;
63  g = 9.81;
64
65

```

```

66 %syms theta_current
67
68 currentHeight = y0 + (r - r*cosd(thetaStep));
69 currentX = x0 + r*sind(thetaStep);
70
71 %
72 %syms currentHeight
73 v = sqrt ( 2 * g * (RollerHeight - (y0 + r - r*cosd(thetaStep)))) ;
74
75 % get value for accelration
76
77 an = (v.^2)/r; %normal accelration
78 at = g*sind(thetaStep); %tangential accelration
79
80 Normal = (an) + g*cosd(thetaStep) ; % normal force (N divided by m, so later you can just
    multiply by 1/g!) to get g's.
81
82 % Normal force will make the
83
84
85 %% prepare outputs
86
87 zf = 1; %final z position, will assume it's one because cart needs to go into the page to
    finish the loop.
88
89 % sub in thetas to get current points:
90
91 OutputY = currentHeight;
92 OutputX = currentX;
93 OutputZ = linspace(z0,zf,length(OutputX));
94
95
96 G = double(Normal./9.81);
97
98 fprintf('The second loop generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
99 fprintf(' G, forward and upward. \n ' );
100
101 %% write outputs
102 Circumference = [Circumference];
103 Outputs_G = [ G ] ;
104 Outputs_Loc = [ OutputX ; OutputY ; OutputZ ] ;
105 Outputs_Velocity = [ v ] ;
106
107 TimeElapsed = 0 ;
108
109 end

```

#### RampDown.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity] = RampDown(v0, t0, RampAngle,
    Length, y0, x0, z0, RollerHeight )
2
3 %
4 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5 %
6 %

```

```

7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to module a
14 ramp down, where the user will determine the ramp specifications, and
15 initial conditions, and the function will return the end status.
16 % -----
17
18 note: here all x and y is tilted, so that +x is pointing with the heading
19 vector, i.e where the rider is looking.
20
21 % -----
22 Inputs:
23 1- v0 : Initial velocity (Total magnitude).
24
25 2- t0: initial time relative to the whole roller coaster when
26 the cart started going on the ramp.
27
28 3- RampAngle: relative to horizon, in degrees.
29
30 4- Length: hypotenuse of the ramp.
31
32 5- y0: initial y position.
33
34 6- x0: initial x position.
35
36 7- z0: initial z position.
37
38 8- RollerHeight: Roller coaster maximum height.
39
40 % -----
41
42
43 Outputs:
44
45 1- TimeElapsed : time just spent on this segment.
46
47 2- Outputs_G: G's at each (x,y,z) coordinate.
48
49 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
50
51 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
52
53
54 % -----
55
56
57 %}
58
59 %% Define constants
60
61

```



```

62 g = 9.81;
63
64 %% accelerations
65
66 ax = g * sind ( RampAngle ) ;
67 ay = 0 ;
68
69 AccelerationFinal = ax;
70
71 %% velocity: since acceleration is constant, we can go ahead and use kinematic equations.
72
73 % get final velocity based on height.
74 syms h
75 v(h) = sqrt ( 2 * g * (RollerHeight - h)) ;
76
77
78
79 %% position
80
81 %{ we can also get it from the height, which is inputted by the user.
82
83 RampHeight = Length * sind ( RampAngle );
84 RampWidth = Length * cosd ( RampAngle );
85
86 %evaluate veloc function defined above.
87 VelocFinal = double(v(y0 - RampHeight));
88
89 % store as output
90 PositionFinalX = x0 + RampWidth;
91 PositionFinalY = y0 - RampHeight;
92
93
94 %% time spent: since acceleration is constant, we can go ahead and use kinematic equations.
95
96 % here all velocity will be in the x direction, thus we can get time from
97 % kinematic equations that uses velocity and acceller will be used.
98
99 % ( ( v - v0 ) / a0 ) + t0 = t , where all of these in x since ay is 0.
100 % Again the coordinate system is tilted, but the time should be the same.
101
102
103 TimeElapsed = ( (VelocFinal - v0) / ax ) + t0 ;
104
105
106
107 %% G's felt.
108
109 G = cosd(RampAngle) ;
110 fprintf('The ramp down generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
111 fprintf(' G, forward and upward. \n ' );
112
113 %% write outputs
114
115 % All G's here are forward.
116 Outputs = [ PositionFinalX ; PositionFinalY ; z0 ; G ] ;

```

```

117
118 Outputs_G = [G];
119 Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
120 Outputs_Velocity = [ VelocFinal ] ;

```

#### ParabolaiHill.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity Arclength ] = ParabolaicHill(
    t0, y0, x0, z0, theta, a0, v)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
4 %
5 %{
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to module the
14 Parabolaic Hill. This segment follows the free-fall conditions, and hence
15 the G's will be == 0 ;
16
17 % -----
18 Inputs:
19
20 Inputs:
21
22 1- t0: initial time relative to the whole roller coaster when
23 the cart started going on the ramp.
24
25 2- y0: initial y position.
26
27 3- x0: initial x position.
28
29 4- z0: initial y position.
30
31 5- theta: angel of initial object @ the beginning, just like lunching an
32 object from a cannon.
33
34 6- a0: inital accelration.
35
36 7- v: initial velocity.
37
38 % -----
39
40
41 Outputs:
42
43 1- TimeElapsed : time just spent on this segment.
44
45 2- Outputs_G: G's at eaxh (x,y,z) coordinate.
46
47 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
48

```

```

49 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
50
51 5- Arclength : which is equivalent to the total distance covered.
52
53
54 % -----
55
56
57 %}
58
59
60 g = 9.81;
61
62 vx = v*cosd(theta);
63
64 % time, all equations will be depending on it.
65 vy = @(t) -g*t + v*sind(theta);
66 y = @(t) -(1/2)* g * t^2 + v*sind(theta) * t + y0 ;
67 x = @(t) vx*t + x0;
68
69
70 % place holders
71
72 CurrentX = [];
73 CurrentY = [];
74 CurrentZ = [];
75 CurrentVy = [];
76 CurrentVx = [];
77
78
79 % there's no way we will start from height == 0, so we will make it as
80 % initial condition just to start the loop.
81
82 y_loop = y0+1 ;
83 i = 0 ; % time counter
84 TimeElapsed = i;
85 Outputs = [ CurrentX ; CurrentY ; CurrentVy ; CurrentVx ] ;
86
87 while y0<=y_loop
88
89     CurrentX = [ CurrentX ; x(i) ] ;
90     CurrentY = [ CurrentY ; y(i) ] ;
91     CurrentZ = [ CurrentZ ; z0 ] ;
92     CurrentVy = [ CurrentVy ; vy(i) ] ;
93     CurrentVx = [ CurrentVx ; vx ] ;
94     TimeElapsed = [ TimeElapsed ; i ] ;
95     y_loop = double(subs(y,i));
96     i = i + 0.01;
97
98
99
100
101 end
102
103 %Calculate the arc length of the parabola

```

```

104 b = CurrentX(end) - x0;
105 a = max(CurrentY) - y0;
106 s = sqrt(b^2+(16*a^2));
107 Arclength = (s/2) + (b^2/(8*a)) * log(((4*a)+s)/b);
108 %% write outputs
109 Arclength = [Arclength];
110 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ] ;
111 Outputs_Velocity = [ CurrentVy' ; CurrentVx' ] ;
112 Outputs_G = [ zeros(1,length(CurrentX)) ] ;
113 G = zeros(1,length(CurrentX));
114 fprintf('The parabolic hill generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G))
)
115 fprintf(' G, forward and upward. \n ' );
116
117
118 end

```

#### RampDownOpposite.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ] = RampDownOpposite(t0,
    RampAngle, Length, y0, x0, z0)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
4 %
5 %{
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to module a
14 ramp down, where the user will determine the ramp specifications, and
15 initial conditions, and the function will return the end status. This is a
16 version of the ramp the goes down in a positive slope rather than negative.
17 % -----
18
19 note: here all x and y is tilted, so that +x is pointing with the heading
20 vector, i.e where the rider is looking.
21
22 % -----
23 Inputs:
24
25 1- t0: initial time relative to the whole roller coaster when
26 the cart started going on the ramp.
27
28 2- RampAngle: relative to horizon, in degrees.
29
30 3- Length: hypotenuse of the ramp.
31
32 4- y0: initial y position.
33
34 5- x0: initial x position.
35
36 6- z0: initial z position.

```

```

37
38 % -----
39
40
41 Outputs:
42
43 1- TimeElapsed : time just spent on this segment.
44
45 2- Outputs_G: G's at each (x,y,z) coordinate.
46
47 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
48
49 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
50
51
52 % -----
53
54
55 %}
56
57 %% Define constants
58
59
60 g = 9.81;
61
62 %% accelerations
63
64 ax = g * sind ( RampAngle ) ;
65 ay = 0 ;
66
67 AccelerationFinal = ax;
68
69 %% velocity: since acceleration is constant, we can go ahead and use kinematic equations.
70
71 % get final velocity based on height.
72 syms h
73 v(h) = sqrt ( 2 * g * (125 - h)) ;
74
75
76
77 %% position
78
79 %{ we can also get it from the height, which is inputted by the user.
80
81 RampHeight = Length * sind ( RampAngle );
82 RampWidth = Length * cosd ( RampAngle );
83
84 %evaluate veloc function defined above.
85 VelocFinal = double(v(y0 - RampHeight));
86
87 % store as output
88 PositionFinalX = x0 - RampWidth;
89 PositionFinalY = y0 - RampHeight;
90
91

```

```

92 %% time spent: since acceleration is constant, we can go ahead and use kinematic equations.
93
94 % here all velocity will be in the x direction, thus we can get time from
95 % kinematic equations that uses velocity and acceler will be used.
96
97 % ( ( v - v0 ) / a0 ) + t0 = t , where all of these in x since ay is 0.
98 % Again the coordinate system is tilted, but the time should be the same.
99
100
101 TimeElapsed = [];
102
103
104
105 %% G's felt.
106
107 G = cosd(RampAngle) ;
108 fprintf('The second ramp generates a maximum magnitude of: %6.2f %12.8f \n ', abs(max(G)))
109 fprintf(' G, forward and backward. \n ' );
110
111 %% write outputs
112
113 % All G's here are forward.
114 Outputs = [ PositionFinalX ; PositionFinalY ; z0 ; G ] ;
115
116 Outputs_G = [G];
117 Outputs_Loc = [ PositionFinalX ; PositionFinalY ; z0 ];
118 Outputs_Velocity = [ VelocFinal ] ;

```

TransitionfromBankedTurn.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
   Transition_fromBankedTurn(t0,x0, y0, z0,theta, r)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
4 %
5 %{
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta.
17
18 % -----
19 Inputs:
20
21 2- t0: initial time relative to the whole roller coaster when
22 the cart started going on the ramp.
23
24
25 6- x0: initial x position.

```

```

26
27 5- y0: initial y position.
28
29 7- z0: initial z position.
30
31 3- theta: imagine the transition is a circle, starting from 0, how many degrees
32 should it run too? That's your theta.
33
34 4- r: Radius of the circle.
35
36
37 8- RollerHeight: Roller coaster maximum height.
38
39 % -----
40
41
42 Outputs:
43
44 1- TimeElapsed : time just spent on this segment.
45
46 2- Outputs_G: G's at each (x,y,z) coordinate.
47
48 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
49
50 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
51
52 5- ArcLength: which is equivalent to the total distance covered
53
54 % -----
55
56
57 %}
58 g = 9.81;
59
60 %% begin calculations of position
61
62 ArcLength = theta * pi / 180 * r;
63
64 ThetaRange = (180 : (180 + theta))';
65
66 % get location
67 CurrentY = y0 - r - r * cosd(ThetaRange); %height as a function of theta
68
69 CurrentX = x0 + r * sind(ThetaRange);
70
71 CurrentZ = z0 * ones(length(ThetaRange),1);
72
73 CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height
74
75 Normal = g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m
76
77 G = Normal/g;
78
79 fprintf('The transition out of the banked turn generates a maximum magnitude of: %6.2f %12.8
      f \n ', abs(max(G)))

```



```

80 fprintf(' G, forward and backward. \n ' );
81 %% write output
82
83 TimeElapsed = [];
84 ArcLength =[ArcLength];
85 Outputs_G = [ G' ] ;
86 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
87 Outputs_Velocity = [ CurrentV' ] ;
88
89 end

```

#### TransitionfromRampDown.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength ] =
   Transition_fromRampDown(t0,x0, y0, z0,theta, r)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
4 %
5 %{
6
7 Done by:
8 — Abdullah AlMugirun
9 — Mohamed Aichiouene
10 — Aufa Amirullah
11 — Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta.
17
18 % -----
19 Inputs:
20
21 1— t0: initial time relative to the whole roller coaster when
22 the cart started going on the ramp.
23
24
25 2— x0: initial x position.
26
27 3— y0: initial y position.
28
29 4— z0: initial z position.
30
31 5— theta: imagine the transition is a circle, starting from 0, how many degrees
32 should it run too? That's your theta.
33
34 6— r: Raduis of the circle.
35
36
37 % -----
38
39
40 Outputs:
41
42 1— TimeElapsed : time just spent on this segment.

```

```

43
44 2- Outputs_G: G's at each (x,y,z) coordinate.
45
46 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
47
48 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
49
50 5- ArcLength: which is equivalent to the total distance covered
51
52 % -----
53
54
55 %}
56
57 g = 9.81;
58
59 %% begin calculations of position
60
61 ArcLength = theta * pi / 180 * r;
62
63 ThetaRange = (360 - theta : 360)';
64
65 CurrentY = y0 + r * cosd(360 - theta) - r * cosd(ThetaRange); %height as a function of theta
66
67 CurrentX = x0 - r * sind(360 - theta) + r * sind(ThetaRange);
68
69 CurrentZ = z0 * ones(length(ThetaRange),1);
70
71 CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height
72
73 Normal = - g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m
74
75 G = Normal/g;
76
77 fprintf('The transition out of the ramp generates a maximum magnitude of: %6.2f %12.8f \n ',
78         abs(max(G)))
79 fprintf(' G, forward and upward. \n ');
80
81 %% write output
82
83 TimeElapsed = [];
84 ArcLength = [ArcLength];
85 Outputs_G = [ G' ];
86 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
87 Outputs_Velocity = [ CurrentV' ];
88 end

```

Transitioninto.m

```

1 function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] = Transition_into(t0
2     ,x0, y0, z0,theta, Radius)
3
4 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5 %
6 %
7 %
8 %
9 %

```

```

6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta. This's a transition into something, something above you.
17
18 % -----
19 Inputs:
20
21 1- t0: initial time relative to the whole roller coaster when
22 the cart started going on the ramp.
23
24
25 2- x0: initial x position.
26
27 3- y0: initial y position.
28
29 4- z0: initial z position.
30
31 5- theta: imagine the transition is a circle, starting from 0, how many degrees
32 should it run too? That's your theta.
33
34 6- Raduis: Raduis of the circle.
35
36
37
38 % -----
39
40
41 Outputs:
42
43 1- TimeElapsed : time just spent on this segment.
44
45 2- Outputs_G: G's at eaxh (x,y,z) coordinate.
46
47 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
48
49 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
50
51 5- ArcLength: which is equivalent to the total distance covered
52
53 % -----
54
55
56 %}
57
58 %% Gravity
59 g = 9.81;
60 %% Tangent normal coordinate CurrentYstem

```

```

61
62 ArcLength = theta * pi / 180 * Raduis;
63
64 ThetaRange = (0:(theta))';
65
66 % Location
67 CurrentY = y0 + (Raduis - Raduis * cosd(ThetaRange));
68
69 CurrentX = x0 + Raduis * sind(ThetaRange);
70
71 CurrentZ = z0 * ones(length(ThetaRange),1);
72
73
74 CurrentV = sqrt(2 * g * (125 - CurrentY));
75
76 Normal = g * cosd(ThetaRange) + CurrentV.^2/Raduis; % Normal force / m
77
78 G = Normal/g; % G force
79
80 fprintf('The transition into the parabolic hill generates a maximum magnitude of: %6.2f
      %12.8f \n ', abs(max(G)))
81 fprintf(' G, forward and upward. \n ' );
82
83 %% write output
84 ArcLength = [ArcLength];
85 TimeElapsed = [];
86 Outputs_G = [ G' ] ;
87 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
88 Outputs_Velocity = [ CurrentV' ] ;
89
90
91 end

```

Transitionout.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength ] = Transition_out(
   t0,x0, y0, z0,theta, r)
2
3 %
4 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5 %
6 %{
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta. This's transition out of something, to something below you.
17
18 % -----
19 Inputs:
20

```

```

21 1- t0: initial time relative to the whole roller coaster when
22 the cart started going on the ramp.
23
24
25 2- x0: initial x position.
26
27 3- y0: initial y position.
28
29 4- z0: initial z position.
30
31 5- theta: imagine the transition is a circle, starting from 0, how many degrees
32 should it run too? That's your theta.
33
34 6- r: Radius of the circle.
35
36
37 % -----
38
39
40 Outputs:
41
42 1- TimeElapsed : time just spent on this segment.
43
44 2- Outputs_G: G's at each (x,y,z) coordinate.
45
46 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
47
48 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
49
50 5- ArcLength: which is equivalent to the total distance covered
51
52 % -----
53
54
55 %}
56 g = 9.81;
57
58 %% begin calculations of position
59
60 ArcLength = theta * pi / 180 * r;
61
62 ThetaRange = (360 - theta : 360)';
63
64 % get location
65 CurrentY = y0 + r * cosd(360 - theta) - r * cosd(ThetaRange); %height as a function of theta
66
67 CurrentX = x0 - r * sind(360 - theta) + r * sind(ThetaRange);
68
69 CurrentZ = z0 * ones(length(ThetaRange),1);
70
71 CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height
72
73 Normal = g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m
74
75 G = Normal/g;

```

```

76
77 fprintf('The transition out of the parabolic hill generates a maximum magnitude of: %6.2f
    %12.8f \n ', abs(max(G)))
78 fprintf(' G, forward and backward. \n ' );
79
80 %% write output
81 ArcLength = [ArcLength];
82 TimeElapsed = [];
83 Outputs_G = [ G' ] ;
84 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
85 Outputs_Velocity = [ CurrentV' ] ;
86
87 end

```

#### TransitiontoGround.m

```

1 function [ TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
    Transition_toGround(x0, y0, z0,theta, radius)
2
3 %
4 % ASEN 2003: Dynamics, Lab 1, Roller Coaster
5 %
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta.
17
18 % -----
19 Inputs:
20
21
22 1- x0: initial x position.
23
24 2- y0: initial y position.
25
26 3- z0: initial z position.
27
28 4- theta: imagine the transition is a circle, starting from 0, how many degrees
29 should it run too? That's your theta.
30
31 5- r: Raduis of the circle.
32
33
34 % -----
35
36
37 Outputs:
38
39 1- TimeElapsed : time just spent on this segment.

```

```

40
41 2- Outputs_G: G's at each (x,y,z) coordinate.
42
43 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
44
45 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
46
47 5- ArcLength: which is equivalent to the total distance covered
48
49 % -----
50
51
52 %}
53
54 g = 9.81;
55
56 %% begin calculations of position
57
58 ArcLength = theta * pi / 180 * radius;
59
60 ThetaRange = (theta:-1:0)';
61
62 % get location
63 CurrentY = y0 + radius * cosd(theta) - radius * cosd(ThetaRange); %height as a function of
    theta
64
65 CurrentX = x0 - radius * sind(theta) + radius * sind(ThetaRange);
66
67 CurrentZ = z0 * ones(length(ThetaRange),1);
68
69 CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height
70
71 Normal = g * cosd(ThetaRange) + CurrentV.^2/radius; % Normal force / m
72
73 G = Normal/g;
74
75 fprintf('The transition to ground generates a maximum magnitude of: %6.2f %12.8f \n ', abs(
    max(G)))
76 fprintf(' G, forward and backward. \n ');
77
78 %% write output
79
80 TimeElapsed = [];
81 ArcLength = [ArcLength];
82 Outputs_G = [ G' ];
83 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
84 Outputs_Velocity = [ CurrentV' ];
85
86 end

```

TransitionToDownRamp.m

```

1 function [TimeElapsed Outputs_G Outputs_Loc Outputs_Velocity ArcLength] =
    TransitionToDownRamp(r,theta,x0,y0,z0)
2 %
3 % ASEN 2003: Dynamics, Lab 1, Roller Coaster

```

```

4 %
5 %{
6
7 Done by:
8 - Abdullah AlMugirun
9 - Mohamed Aichiouene
10 - Aufa Amirullah
11 - Abdulla AlAmeri
12
13 This function is one segment of a roller coaster, it attempts to
14 module a transition from a segment. The segment is in the name of the
15 function. Most transitions are assumed to be circles that run to a certain
16 theta.
17
18 % -----
19 Inputs:
20
21 1- r: Raduis of the circle.
22
23 2- theta: imagine the transition is a circle, starting from 0, how many degrees
24 should it run too? That's your theta.
25
26 3- x0: initial x position.
27
28 4- y0: initial y position.
29
30 5- z0: initial z position.
31
32
33 % -----
34
35
36 Outputs:
37
38 1- TimeElapsed : time just spent on this segment.
39
40 2- Outputs_G: G's at eaxh (x,y,z) coordinate.
41
42 3- Outputs_Loc: [ 3 x n ], where each column is one point in (x,y,z).
43
44 4- Outputs_Velocity: Velocity at each point on the rollercoaster.
45
46 5- ArcLength: which is equivalent to the total distance covered
47
48 % -----
49
50
51 %}
52
53 g = 9.81;
54
55 %% begin calculations of position
56
57 ArcLength = [theta * pi / 180 * r];
58

```



```

59 ThetaRange = (180 : (theta + 180))';
60
61 CurrentY = y0 - r - r * cosd(ThetaRange); %height as a function of theta
62
63 CurrentX = x0 - r * sind(ThetaRange);
64
65 CurrentZ = z0 * ones(length(ThetaRange),1);
66
67 CurrentV = sqrt(2 * g * (125 - CurrentY)); %velocity due to change in height
68
69 Normal = - g * cosd(ThetaRange) + CurrentV.^2/r; % Normal force / m
70
71 G = Normal/g;
72
73 fprintf('The transition to the ramp generates a maximum magnitude of: %6.2f %12.8f \n ', abs
    (max(G)))
74 fprintf(' G, forward and upward. \n ');
75
76
77 %% write outputs
78
79 % All G's here are forward.
80 ArcLength = [ArcLength];
81 TimeElapsed = [];
82 Outputs_G = [ G' ];
83 Outputs_Loc = [ CurrentX' ; CurrentY' ; CurrentZ' ];
84 Outputs_Velocity = [ CurrentV' ];
85 end

```

colorline3.m

```

1 function h = color_line3(x, y, z, c, varargin)
2 % color_line3 plots a 3-D line with c-data as color
3 %
4 %     h = color_line(x, y, z, c)
5 %     by default: 'LineStyle','-' and 'Marker','none'
6 %
7 %     or
8 %     h = color_line(x, y, z, c, mark)
9 %     or
10 %     h = color_line(x, y, z, c, 'Property','value'...)
11 %         with valid 'Property','value' pairs for a surface object
12 %
13 % in:  x      x-data
14 %      y      y-data
15 %      z      z-data
16 %      c      4th dimension for colouring
17 %      mark   for scatter plots with no connecting line
18 %
19 % out: h      handle of the surface object
20
21
22 h = surface(...
23     'XData',[x(:) x(:)],...
24     'YData',[y(:) y(:)],...
25     'ZData',[z(:) z(:)],...

```

```

26     'CData',[c(:) c(:)],...
27     'FaceColor','interp',...
28     'EdgeColor','interp',...
29     'Marker','none');
30
31 if nargin ==5
32     switch varargin{1}
33         case {'+' 'o' '*' '.' 'x' 'square' 'diamond' 'v' '^' '>' '<' 'pentagram' 'p' '
34             hexagram' 'h'}
35             set(h,'LineStyle','none','Marker',varargin{1})
36         otherwise
37             error(['Invalid marker: ' varargin{1}])
38     end
39 elseif nargin > 5
40     set(h,varargin{:})
41 end

```