```
clear all;
clc;

%% Constants
global g R mu fr airDen gearRatio Cd L eff_trans;
g = 9.81;               % Acceleration due to gravity (m/s^2)
R = 0.6681;             % Wheel Radius (m)
mu = 1;                 % Friction Coefficient
fr = 0.009;             % Rolling friction coefficient
airDen = 1.26;          % Air Density (Kg/m^3)
gearRatio = 8;          % Gear Ratio (including final drive ratio)
Cd = 0.27;              % Coefficient of Drag
L = 2.6;                % Wheel Base (m)
eff_trans = 0.95;       % Efficiency of transmission
eff_batt = 0.9;         % Efficiency of battery
Kp = 1;                 % Parameter Kp for PI controller
Ki = 0;                 % Parameter Ki for PI controller

%% Variables

% Battery
global drivingBattCap;
capCell = 17 * 0.001;       % Cell capacity in kWh
numCellSeries = 192;        % Number of cells in series
numParallelStrings = 16;    % Number of parallel strings of the cells in series (TBD)
netBattCap = capCell * numCellSeries * numParallelStrings;   % Net battery capacity (kWh)
usableBattCap = 0.9 * netBattCap;   % Useful battery energy capacity (kWh)
drivingBattCap = 0.9 * usableBattCap;   % Battery capacity available for driving (after
cooling) (kWh)

% Vehicle Mass (Kg)
global mVehicle mAxle mGear;
mPass = 2*101;              % Mass of two AM95 passengers
mCargo = 2*7;               % Mass of cargo for two passengers
mInfo = 15;                 % Mass of infotainment system
mSun = 10;                  % Mass of sunroof
mUnsprung = 4*60;           % Unsprung mass
mProp = 0;                  % Mass of propeller shaft (not used)
mDiff = 10;                 % Mass of Differential
mGear = 45;                 % Mass of Gearbox (to be decided)
mAxle = 2*5;                % Mass of Axle (Add another for AWD)
mBody = 250;                % Mass of body (TBD)
mFrame = 250;               % Mass of Frame Rail (TBD)
mBattery = netBattCap * 1000/160;       % Mass of battery in Kg (TBD)
mMotor = 0;                 % Mass of Motor (TBD)
mVehicle = mPass + mCargo + mInfo + mSun + mUnsprung + mProp + mDiff + mGear + mAxle + mBody
+ mFrame + mBattery;   % Mass of Vehicle (Additions to be made)

% Vehicle Cost ($)
global cVehicle cAxle cGear;
```

```
 cElec = 3000;       % Cost of Misc. Electronics
 cWheel = 4*450;     % Cost of Wheel Assemblies
 cProp = 0;          % Cost of propeller shaft (Not used)
 cDiff = 600;        % Cost of Differential
 cGear = 7500;       % Cost of Gear Box (TBD. Also, double for 2AWD and consider for 4AWD)
 cAxle = 2*350;      % Cost of Axles (Add another for AWD)
 cMotor = 0;         % Cost of Motor (TBD)
 cBody = 9400;       % Cost of body (TBD)
 cFrame = 2*mFrame;  % Cost of frame (TBD)
 cBattery = netBattCap * 145;       % Cost of battery (TBD)
 cVehicle = cElec + cWheel + cProp + cDiff + cAxle + cBody + cFrame + cBattery;  % Cost of
vehicle (Additions to be made)

 % Vehicle Dimensions
 global fArea WD;
 w103Dim = 2.02;         % Dimension W103 (Vehicle Width)
 h101Dim = 1.35;         % Dimension H101 (Vehicle Height)
 fArea = w103Dim*h101Dim;% Frontal Area (TBD)
 WD = 0.48;              % Weight Distribution from the front (TBD)
 a = (1-WD)*L;           % Distance of CG from the front (m) (TBD)
 b = WD*L;               % Distance of CG from the rear (m) (TBD)
 h = 0.65;               % Height of CG (m) (TBD)
 isRWD = 1;              % 1 for RWD and 0 for AWD

 % Motor
 global EM_Efficiency EM_Torque_Map EM_Omega_Map EM_Torque_Max EM_Omega_Max contPowerRating;
 contPowerRating = 0;    % Continuous power rating of motor (W)

 % Drive Cycle
 global sim_time followDriveCycle grade;
 load US06_Drive_Cycle.mat;
 sim_time = t_cyc(end);
 followDriveCycle = 1;   % Whether to follow drive cycle or not (=0 for Wide Open Throttle)
 grade = 0;              % Road grade (changes for testing 130kmph ability at 12% slope)

 %% Drive Cycle Tests
 % We'll run tests for different possible configurations and accumulate the
 % results to determine the best possible setup for us

 global profit mass cost range toHundredTime topSpeed index;
 profit = zeros(1,12);   % Variable to store profit values from various configurations
 mass = zeros(1,12);   % Variable to store mass values for various configurations
 cost = zeros(1,12);   % Variable to store cost values for various configurations
 range = zeros(1,12);   % Variable to store vehicle range values for battery size
 toHundredTime = zeros(1,12);   % Variable to store 0-100 kmph times in s
 topSpeed = zeros(1,12);   % Variable to store top speed values from various configurations
 index = 1;

 %% Motor A
 mSel = 'A';
```

```
% 1 motor RWD
isRWD = 1;
numMotors = 1;
performProfitAnalysis(mSel, isRWD, numMotors);

% 2 motor RWD
isRWD = 1;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 2 motor AWD
isRWD = 0;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 4 motor AWD
isRWD = 0;
numMotors = 4;
performProfitAnalysis(mSel, isRWD, numMotors);

%% Motor B
mSel = 'B';

% 1 motor RWD
isRWD = 1;
numMotors = 1;
performProfitAnalysis(mSel, isRWD, numMotors);
```

Can use low speed gearbox for Motor B RWD=1 numM=1

```
% 2 motor RWD
isRWD = 1;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 2 motor AWD
isRWD = 0;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 4 motor AWD
isRWD = 0;
numMotors = 4;
performProfitAnalysis(mSel, isRWD, numMotors);

%% Motor C
mSel = 'C';

% 1 motor RWD
isRWD = 1;
numMotors = 1;
```

```
performProfitAnalysis(mSel, isRWD, numMotors);

% 2 motor RWD
isRWD = 1;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 2 motor AWD
isRWD = 0;
numMotors = 2;
performProfitAnalysis(mSel, isRWD, numMotors);

% 4 motor AWD
isRWD = 0;
numMotors = 4;
performProfitAnalysis(mSel, isRWD, numMotors);

%% Summarizing results
motor = ['A'; 'A'; 'A'; 'A'; 'B'; 'B'; 'B'; 'B'; 'C'; 'C'; 'C'; 'C'];
conf = ['RWD'; 'RWD'; 'AWD'; 'AWD'; 'RWD'; 'RWD'; 'AWD'; 'AWD'; 'RWD'; 'RWD'; 'AWD'; 'AWD'];
numM = [1; 2; 2; 4; 1; 2; 2; 4; 1; 2; 2; 4];

T = table(motor, conf, numM, profit' * 1e-6, mass', cost', range', toHundredTime',
topSpeed');
T.Properties.VariableNames = {'Motor', 'Config', 'Number of Motors', 'Profit ($Million)',
'Mass (Kg)', 'Cost $', 'Range (Km)', '0-100kmph Time (s)', 'Top Speed (kmph)'};

T
```

T = 12×9 table

	Motor	Config	Number of Motors	Profit ($Million)	Mass (Kg)
1	A	RWD	1	1.2955e+03	1.4924e+03
2	A	RWD	2	1.6295e+03	1.6124e+03
3	A	AWD	2	524.4191	1.6674e+03
4	A	AWD	4	8.1270e-06	1.9074e+03
5	B	RWD	1	0	1.4294e+03
6	B	RWD	2	308.9733	1.4864e+03
7	B	AWD	2	0.6772	1.5414e+03
8	B	AWD	4	0.0068	1.6554e+03
9	C	RWD	1	2.2834	1.4394e+03
10	C	RWD	2	487.9930	1.5064e+03
11	C	AWD	2	8.7334	1.5614e+03
12	C	AWD	4	9.8936e-05	1.6954e+03

```
[maxProf, maxProfConfig] = max(profit);
maxProf * 1e-6
```

```
ans = 1.6295e+03
```

```
maxProfConfig
```

```
maxProfConfig = 2
```

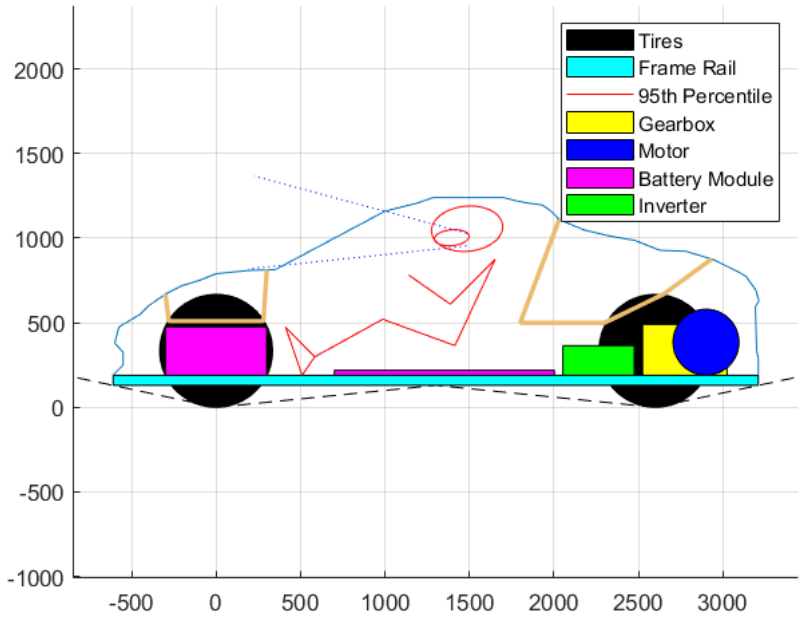
Group_9_Packaging

```
%% Setting all the variable values -- per category
%% Delete the below 2 lines if running in the complete code
% clear
% close all
% requirements
tireRadius = 668.1/2; % in [mm] as per requirement
wheelBase = 2600; % in [mm] as per requirement

% design choices -- variables that can be chosen by the designer
frameRailLength = 3820; % in [mm]
frameRailHeight = 60; % in [mm]
H156 = 130; % Ground Clearance [mm]
A47 = 71;       % Driver shoe plane angle
foot = 300;     % Foot length
L99 = 971;
L53 = L99-203*cosd(A47);        % SGRP to Heel Reference Point
H30 = 175;      % Seat Height
A = 459.1;      % Lower leg segment
B = 456;        % Thigh segment
UA = 375.4;     % Upper Arm segment
FA = 301.5;     % Forearm segment
A44 = 131;      % Knee Angle
A57 = 20;       % Thigh Angle
A40 = 25;       % Torso Angle
Torso = 563;    % Torso length
UA_ang = 45;    % Upper arm Angle
elb_ang = 100;      % Elbow Angle
UVC_ang = 15;       % Min. Upward Vision Cone Angle
DVC_ang = 6;        % Min. Downward Vision Cone Angle
Ang_A = 12;     % Angle of Approach
Ang_D = 12;     % Angle of Departure
Ramp_ang = 10;  % Ramp Breakover angle
Motor = 1;      % Selected motor (A=1, B=2, C=3)
Motor_F = 0;        % Is the vehicle AWD (0=No, 1=Yes)
Diff = 0;       % Differential (0=Integrated, 1=Seperate)
GB = 2;         % Gearbox (1=low torque, 2=high torque)
Parralel_count = 16;
Battery_Cap = 17*192*Parralel_count/1000;  % kWh

% intermediate variable -- computed from design variables or requirements
frontAxle = [0,tireRadius]; % in [mm]
rearAxle = frontAxle + [wheelBase,0]; % follows from definition of wheelBase
middle = (wheelBase)/2;
front = middle - frameRailLength/2;
rear = front +  frameRailLength;
```

```
%% Plot the vehicle packaging
figure(1)
grid on
hold on
axis equal
plotCircle(frontAxle,tireRadius,'k');
plotCircle(rearAxle,tireRadius,'k');

% plot the frame rails
plotRectangle([front,H156+frameRailHeight],[rear,H156],'c')

% plot 95th percentile (points)
AHP = [tireRadius+175,H156+frameRailHeight];
f_point = [AHP(1)-foot*cosd(A47),AHP(2)+foot*sind(A47)];
BOFRP = [AHP(1)-203*cosd(A47),AHP(2)+203*cosd(A47)];
SGRP = [AHP(1)+L53,AHP(2)+H30];
Knee = [SGRP(1)-B*cosd(A57),SGRP(2)+B*sind(A57)];
Shin = [Knee(1)-A*sind(A44-(90-A57)),Knee(2)-A*cosd(A44-(90-A57))];
Shoulder = [SGRP(1)+Torso*sind(A40),SGRP(2)+Torso*cosd(A40)];
Elbow = [Shoulder(1)-UA*sind(UA_ang),Shoulder(2)-UA*cosd(UA_ang)];
Wrist = [Elbow(1)-FA*sind(elb_ang-UA_ang),Elbow(2)+FA*cosd(elb_ang-UA_ang)];
percy = [Shin;AHP;f_point;Shin;Knee;SGRP;Shoulder;Elbow;Wrist];
plot(percy(:,1),percy(:,2),'r')

% plot head ellipse
a=211.25;       % Major Axis
b=133.5;       % Minor Axis
x0 = BOFRP(1)+1043;     % Ellipse centre (X-coordinates) (BOF to Eye)
y0 = AHP(2)+866;     % Ellipse centre (Y-coordinates) (AHP to Eye)
t = linspace(0,2*pi,100);
ang = deg2rad(5.4);      % Head Tilt Angle
x = x0 + a*cos(t)*cos(ang) - b*sin(t)*sin(ang);
y = y0 + b*sin(t)*cos(ang) + a*cos(t)*sin(ang);
plot(x,y,'r');

% plot eye ellipse
a=206.4/2;       % Major Axis
b=93.4/2;       % Minor Axis
x0 = BOFRP(1)+952;     % Ellipse centre (X-coordinates) (BOF to Eye)
y0 = AHP(2)+813;     % Ellipse centre (Y-coordinates) (AHP to Eye)
t = linspace(0,2*pi,100);
ang = deg2rad(5.4);      % Head Tilt Angle
x = x0 + a*cos(t)*cos(ang) - b*sin(t)*sin(ang);
y = y0 + b*sin(t)*cos(ang) + a*cos(t)*sin(ang);
plot(x,y,'r');

% plot Upward and Downward Vision Cone
l = 1300;
UVG_eye = [SGRP(1)+68,SGRP(2)+665];
UVG_2 = [UVG_eye(1)-l*cosd(UVC_ang),UVG_eye(2)+l*sind(UVC_ang)];
UVG = [UVG_eye;UVG_2];
```

```
 plot(UVG(:,1),UVG(:,2),':b')
 DVG_eye = [SGRP(1)+68,SGRP(2)+589];
 DVG_2 = [DVG_eye(1)-l*cosd(DVC_ang),DVG_eye(2)-l*sind(DVC_ang)];
 DVG = [DVG_eye;DVG_2];
 plot(DVG(:,1),DVG(:,2),':b')

 % plot Ang of Approach, Departure, Ramp
 plot([0;0-(l/1.5)*cosd(Ang_A)],[0;(l/1.5)*sind(Ang_A)],'--k')       % Approach line
 plot([wheelBase;wheelBase+(l/1.5)*cosd(Ang_A)],[0;(l/1.5)*sind(Ang_D)],'--k')       %
Departure line
 plot([0;wheelBase/2;wheelBase],[0;H156;0],'--k')

 % plot motor & inverter
 M_rad = [390/2,276/2,280/2];
 M_pos_r = [wheelBase+300,AHP(2)+M_rad(Motor)];
 I_dim = [531,422,175;418,393,180;434,200,80];
 c = 100;        % Clearance from Axle

 % plot gear box & diff
 gb_dim = [250,400;300,500];
 plotRectangle([M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-
250)/2),AHP(2)+gb_dim(GB,1)],[M_pos_r(1)+((gb_dim(GB,2)-250)/2),AHP(2)],'y')
 plotCircle(M_pos_r,M_rad(Motor),'b')        % Plot Rear Motor

 % Plot for front motor & gear box
 if Motor_F == 1
     M_pos_f = [-300,AHP(2)+M_rad(Motor)];
     plotRectangle([M_pos_f(1)-((gb_dim(GB,2)-
250)/2),AHP(2)+gb_dim(GB,1)],[M_pos_f(1)+gb_dim(GB,2)-((gb_dim(GB,2)-250)/2),AHP(2)],'y')
     plotCircle(M_pos_f,M_rad(Motor),'b')    % Plot Front Motor
 end

 % Plot for Battery
 Bat_V = Battery_Cap*10^3/200;        % Battery_Volume (liter)
 Area = Bat_V*10^6/1250;
 if Motor_F ==0
     h_1 = 285;      % Battery height of 1st section
     l_1 = 600;      % Battery length of 1st section
     A_1 = h_1*l_1;  % Area of 1st battery section
     l_2 = M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-250)/2)-50-I_dim(Motor,2)-50-700;      %
Length of 2nd battery
     A_2 = Area-A_1;     % Area of the 2nd battery section
     h_2 = A_2/l_2;      % Height of the remaining battery section
     plotRectangle([-300,AHP(2)+h_1],[300,AHP(2)],'m')
     plotRectangle([700,AHP(2)+h_2],[700+l_2,AHP(2)],'m')
     plotRectangle([M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-250)/2)-50-
I_dim(Motor,2),AHP(2)+I_dim(Motor,3)],[M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-250)/2)-
50,AHP(2)],'g')     % Plot Inverter
 else
     l_1 = M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-250)/2)-50-700;
     h_1 = 70;
```

```
    A_1 = h_1*l_1;
    l_2 = M_pos_r(1)-gb_dim(GB,2)+((gb_dim(GB,2)-250)/2)-50-1650;
    h_2 = (Area-A_1)/l_2;
    plotRectangle([700,AHP(2)+h_1],[700+l_1,AHP(2)],'m')
    plotRectangle([1650,AHP(2)+h_1+h_2],[700+l_1,AHP(2)+h_1],'m')
    plotRectangle([300-I_dim(Motor,3),AHP(2)+I_dim(Motor,2)],[300,AHP(2)],'g')     % Plot
Inverter
 end

 %% Plot Vehicle boundary
 XX = [-610,190;-550,250;-550,325;-600,380;-575,475;-450,550;-400,600;-300,670;-200,720;-
100,750;0,790;200,810;350,814;1000.2,1160;1080,1180;1200,1210;1281.9,1240;1701.5,1240;1780,12
20;...

1850,1207;1931.9,1196.1;1980.9,1161;2030.9,1110.4;2180.9,1048;2329.9,1012.7;2479,987;2628,928
;2777.2,923.1;2935.7,876.5;3095,799.8;3145.95,767.9;3143.3,765;3195.3,692.5;3213.4,630;3192.2
,593.78;3200.4,329;3210,293.65;3210,190];
 plot(XX(:,1),XX(:,2))

 % Front trunk specs
 f_trunk = [-300,670;-280,510;280,510;300,813];
 f_trunk_vol = -(f_trunk(4,2)-(f_trunk(4,2)-f_trunk(1,2))/2-
f_trunk(3,2))*2*f_trunk(2,1)*1250*10^-9;     % Front trunk volume (m^3)
 plot(f_trunk(:,1),f_trunk(:,2),"Color",'#ebbb6e',"LineWidth",2)

 % Rear Trunk Specs
 trunk = [2030.9,1110.4;1800,500;2300,500;2650,670;2935.7,878.5];
 trunk_vol = 2*f_trunk_vol;          % Rear Trunk Volume (m^3)
 plot(trunk(:,1),trunk(:,2),"Color",'#ebbb6e',"LineWidth",2)
 legend({'','Tires','Frame Rail','95th
Percentile','','','','','','','','','Gearbox','Motor','','Battery Module','Inverter'})
```



```
 Ang_of_Ap = -rad2deg(atan(H156/front));     % Angle of Approach
 Ang_of_Dep = rad2deg(atan(H156/(rear-wheelBase-25)));       % Angle of Departure
 Ramp_Ang = 2*rad2deg(atan(H156/(wheelBase/2)));         % Ramp Breakover Angle
 UVC_Ang = -rad2deg(atan((1180-(SGRP(2)+665))/(1080-(SGRP(1)+68))));     % Upward Vision Cone
 DVC_Ang = rad2deg(atan((810-(SGRP(2)+589))/(200-(SGRP(1)+68))));        % Downward Vision
Cone
 veh_param =
[Ang_of_Ap;Ang_of_Dep;Ramp_Ang;UVC_Ang;DVC_Ang;f_trunk_vol;trunk_vol;f_trunk_vol+trunk_vol];
 disp(array2table(veh_param,"RowNames",{'Angle of Approach (deg)','Angle of Departure
(deg)','Ramp Breakover Angle (deg)','Upward Vision Cone (deg)',...
     'Downward Vision Cone (deg)','Front Trunk Volume (m^3)','Rear Trunk Volume (m^3)','Total
Trunk Capacity (m^3)'},"VariableNames",{'Parameter Value'}))
```

	Parameter Value
Angle of Approach (deg)	12.031
Angle of Departure (deg)	12.529
Ramp Breakover Angle (deg)	11.421
Upward Vision Cone (deg)	20.464
Downward Vision Cone (deg)	6.4091
Front Trunk Volume (m^3)	0.16205
Rear Trunk Volume (m^3)	0.3241
Total Trunk Capacity (m^3)	0.48615

Summary of Packaging Model:

This is a dynamic model which can create the side layout of the vehicle provided with the following values:

1. Motor Selection
2. Configuration of Drive
3. Battery Capacity

Since the Drivetrain Sector has finalized to run 2 Motor A's in a RWD configuration, the model is further developed for any RWD configuration. This was done to reduce the compilation time of the code and eliminate the code for AWD.

During the packaging model, driver comfort was considered as the main priority while keeping the vehicle compact at the same time. Taking the reference of a 95th percentile male as the driver and passenger, the following values are constrained to ensure a comfortable seating position in the roadster (sports car reference)

- Driver shoe plane angle (A47) = 71 deg
- BOFRP to SGRP (L99) = 971 mm
- SGRP to Heel Reference Point (L53) = L99-203*cosd(A47)
- Seat Height (H30) = 175 mm
- Lower leg segment (A) = 459.1 mm
- Thigh segment (B) = 456 mm
- Upper Arm segment = 375.4 mm
- Forearm segment = 301.5 mm
- Knee Angle (A44) = 131 deg
- Thigh Angle (A57) = 20 deg
- Torso Angle (A40) = 25 deg
- Torso length Torso = 563 mm
- Upper Arm Inclination = 45 deg
- Elbow Angle = 100 deg

- Upward Vision Cone Angle (UVC_ang) = 15 deg
- Downward Vision Cone Angle (DVC_ang) = 6 deg
- Effective Headroom = 952 mm
- Head Clearance = 110 mm

For the position of the driver and passenger in the side view, we have considered a clearance of 175 mm from the circumference of the front wheel till the AHP point.

Procedure for Plotting the 95th percentile:

- Take the AHP point and SGRP point as 2 starting points for the complete percentile.
- According to the angles for comfortable seating, get the relative coordinates of the manakin
- Once the coordinates are created, plot the figure which shows the position of the driver
- The head ellipse is plotted in the next step followed by the eye ellipse. These are taken on a 8 deg reference line from the SGRP
- Get the Upward and Downward Vision Cone angles to make a rough outline for the vehicle boundary

Procedure for determining the frame dimensions:

- Considering a roadster setup, the Ground Clearance (H156) is kept at 130mm
- To satisfy the angle of approach and departure, the length of the vehicle is kept at 3820mm
- The Frame rail height was considered as 95 for reference, further it was modified to 60 mm which is the actual design height obtained after SFBM analysis on the final packaged vehicle.
- Since all the coordinates are taken as reference to the AHP point, any changes made to a single viable would modify the complete model.

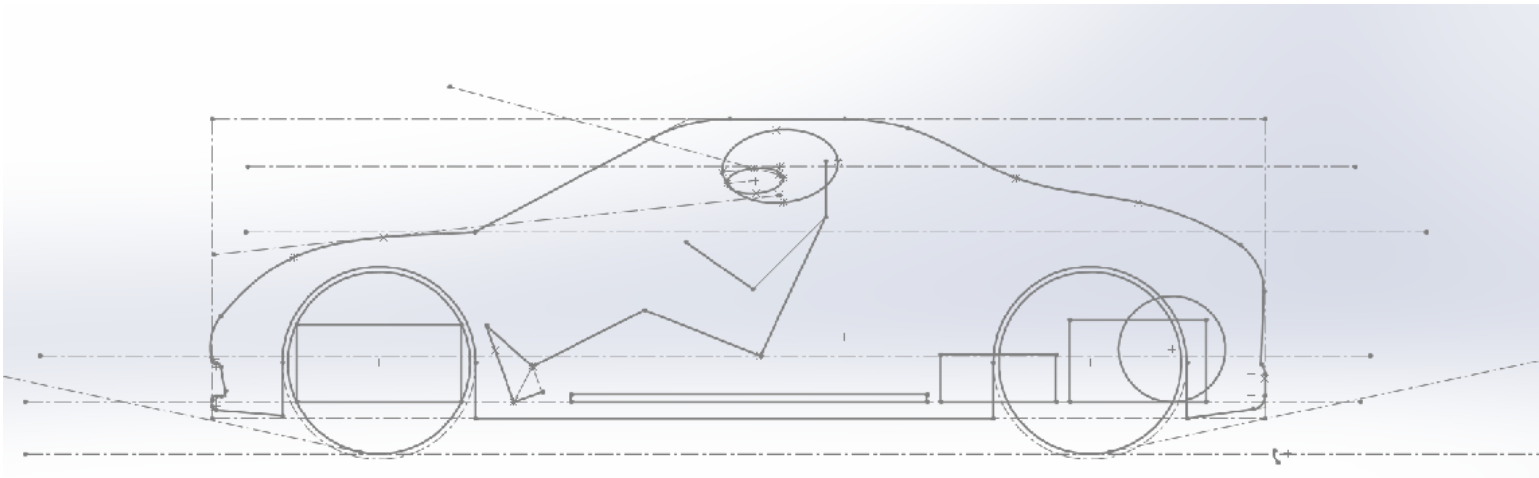
Procedure for determining the location of components:

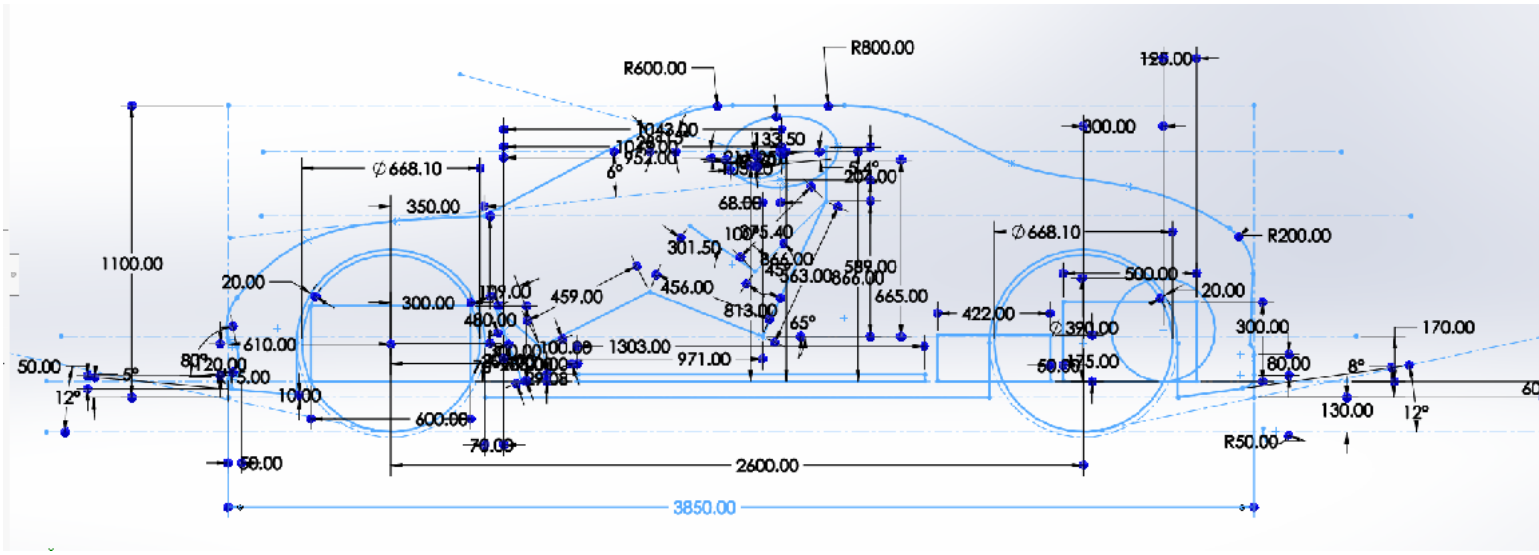
- The motor axis is constrained 300mm away from the rear tire center, in our design it is placed rearwards.
- The gearbox selected is High Torque gearbox for our situation. The C-C distance is 250 mm between the motor axis and gear box output. A single gearbox has 2 motor inputs and 1 single output to the rear axle.
- The use of a separate differential is eliminated as we have it included on our transmission placed on the axle.
- The inverter is placed in the rear section of the vehicle in case of a RWD to have weight balance in the front and rear (48:52 desirable value). For AWD setup, the inverter is placed in the front in this packaging model. Since we have finalized on a RWD setup, we can ignore the AWD setup and packaging.
- The battery is being split into 2 parts where the 1st part is placed in the front section of the vehicle.
- This is placed to ensure desirable weight distribution and is behind the crumple zone of front bumper.
- The middle battery is placed below the driver and is more than 100 mm away from the driver centerline.
- To ensure electrical insulation between the components, a minimum of 50 mm is taken between all the electrical components and battery mounting points.
- The front section of battery pack is having a fixed capacity, and the middle section varies according to the left-over battery capacity required for the desired range of the vehicle.

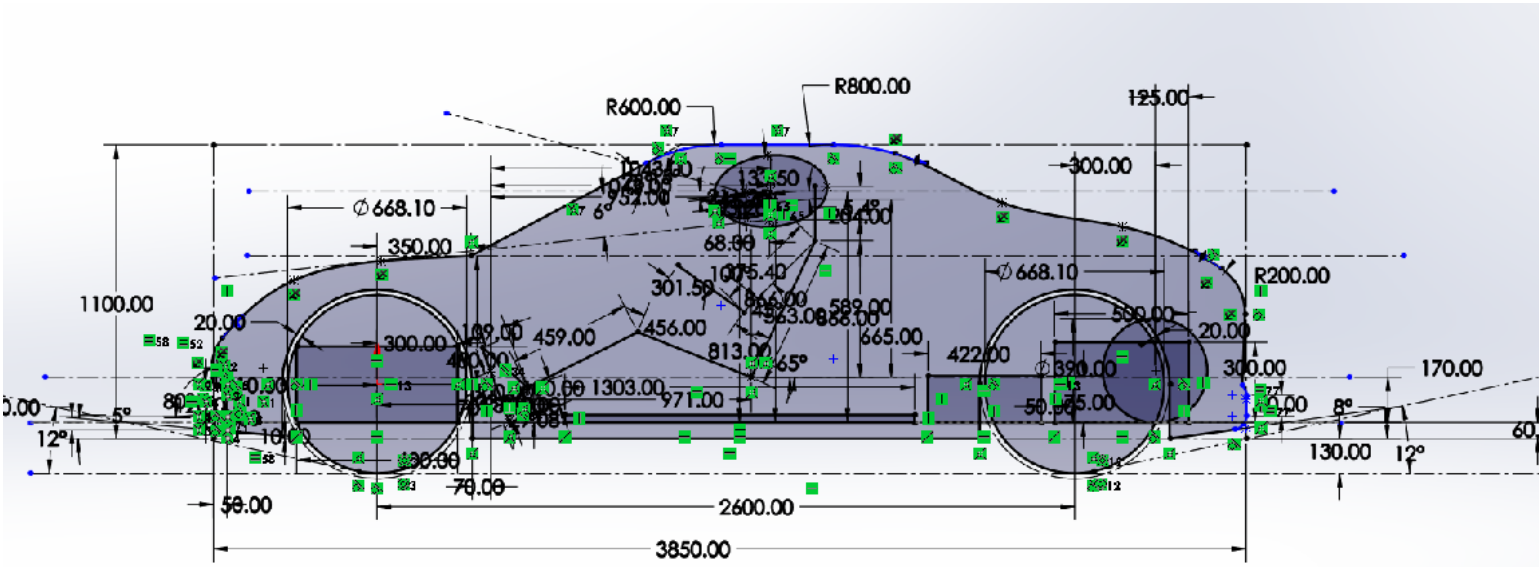
Cargo Space: This is added in the front (f_trunk) and the rear (trunk) for carrying a combined load of 100kg

The ideal place to keep the passenger cargo of 7kg (per passenger) is the front trunk.

Below is the Solid Works 2-D sketch which was used for the outline of the vehicle.




R800.00
R600.00
125.00
300.00
Ø668.10
350.00
1100.00
20.00
300.00
610.00
50.00
80.00
12°
5°
10.00
600.00
70.00
50.00
2600.00
3850.00
Ø668.10
R200.00
500.00
20.00
422.00
300.00
170.00
130.00
12°
R50.00
8°
665.00
971.00
1303.00
813.00
459.00
456.00
301.50
589.00
65°


R800.00
R600.00
125.00
300.00
Ø668.10
350.00
1100.00
20.00
300.00
50.00
80.00
12°
5°
10.00
70.00
50.00
2600.00
3850.00
Ø668.10
R200.00
500.00
20.00
422.00
300.00
170.00
130.00
12°
8°
665.00
971.00
1303.00
813.00
459.00
456.00
301.50
589.00
65°

Since we are using 2 motors, the rear has a usable space of 1250 mm, this is utilized by 2 motors (0.32 m each) and the gear box (0.3 m)

Thus, the space needed = 0.32+0.32+0.3 = 0.92 m

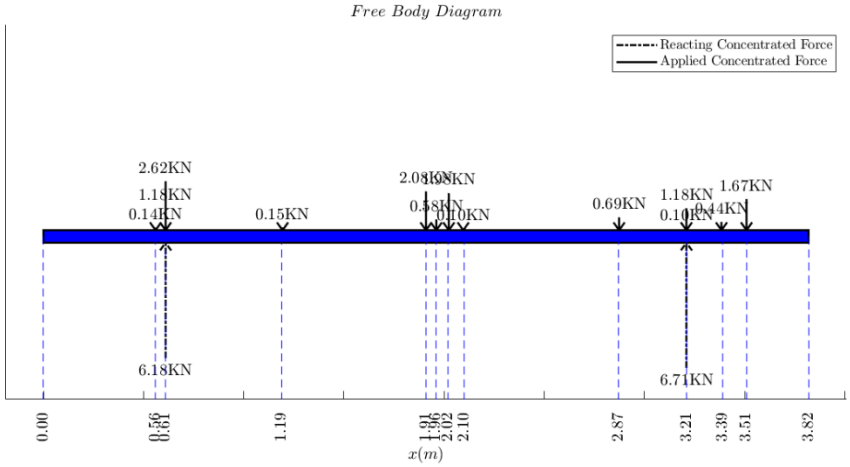
Available Space = 1.25 m

Since the Available Space is more than the required space, we can easily fit our 2 motor RWD system in this layout.

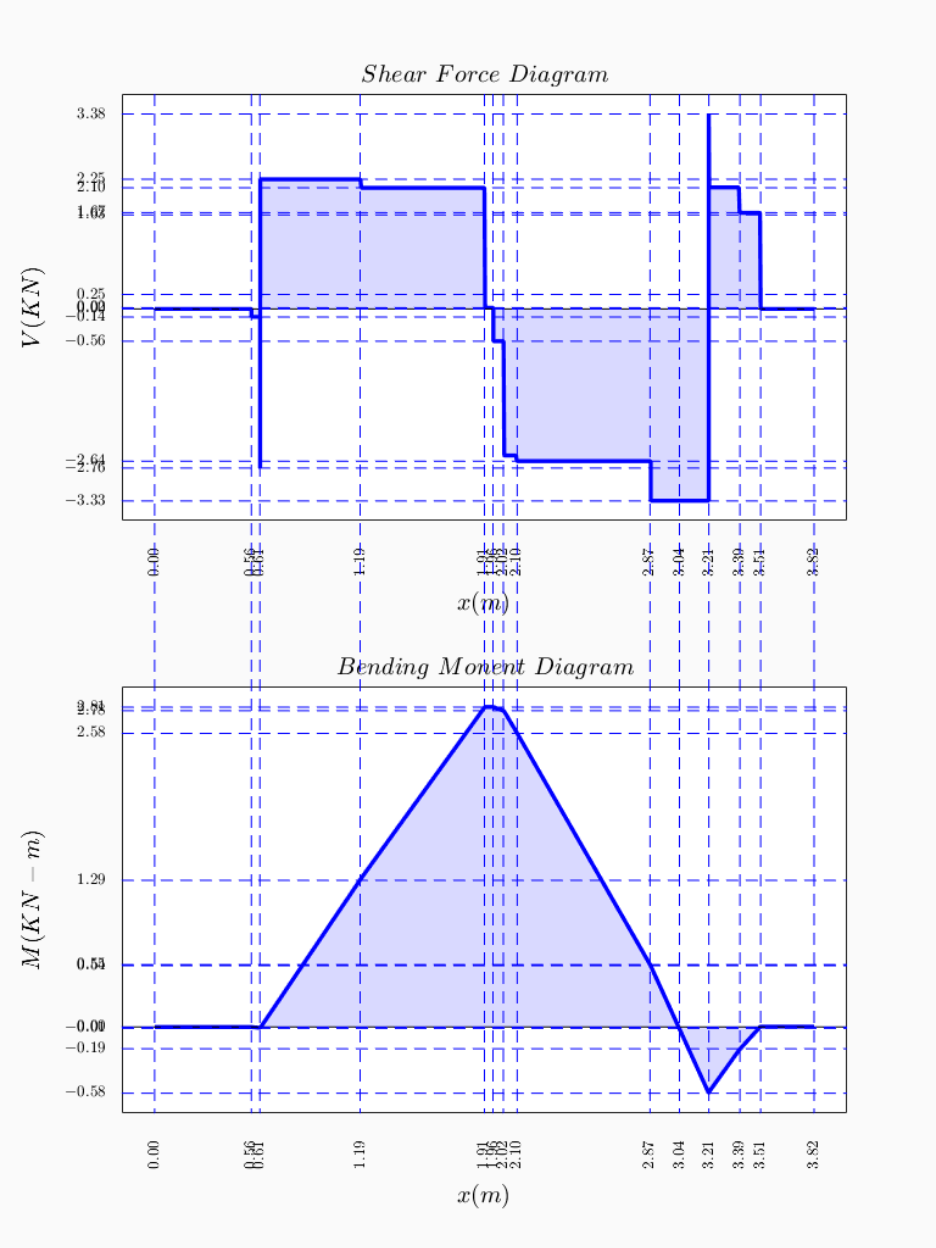
Since the Packaging model is finalized, we move on to SFBM of the frame rail to check the maximum induced stress and deformation.

```
if Motor_F==0
% Mass(kg)
g = 9.81/1000;
AM95_m = 2*101;
Cargo_m = 2*7;
Info_m = 15;
Sunroof_m = 10;
Unsprung_1m = 2*60;
Unsprung_2m = 2*60;
GB = 2;
Gearbox_m = [28,45];
Axle_m = 10;
Body_m = frameRailLength*(1240-H156)*50*10^-6;
Battery_v(1) = A_1*1250*10^-6;
Battery_v(2) = A_2*1250*10^-6;
Battery_c = Battery_v*200;
Battery_m = Battery_c/160;
Motor_m = 2*85;
Inverter_m = 2*35;

% Positions (mm)
ovr_hng = ((frameRailLength-wheelBase)/2);
AM95_p = (AHP(1)+L53+ovr_hng)/1000;
Cargo_p = (ovr_hng-50)/1000;
Info_p = (Shin(1)+ovr_hng)/1000;
Sunroof_p = (BOFRP(1)+1043+ovr_hng)/1000;
Unsprung_1p = (0+ovr_hng)/1000;
Unsprung_2p = (wheelBase+ovr_hng)/1000;
Gearbox_p = (M_pos_r(1)+((gb_dim(GB,2)-250)/2)-gb_dim(GB,2)/2+ovr_hng)/1000;
Axle_p = (wheelBase+ovr_hng)/1000;
Body_p = (frameRailLength/2)/1000;
Battery_p = [ovr_hng,700+l_2/2+ovr_hng]/1000;
Motor_p = (wheelBase+300+ovr_hng)/1000;
Inverter_p = (700+l_2+50+I_dim(Motor,2)/2+ovr_hng)/1000;

% Defining Loads
F_AM95 = {'CF',-AM95_m*g,AM95_p};
F_Cargo= {'CF',-Cargo_m*g,Cargo_p};
F_Info = {'CF',-Info_m*g,Info_p};
F_Sunroof = {'CF',-Sunroof_m*g,Sunroof_p};
```

```
 F_Unsprung_1 = {'CF',-Unsprung_1m*g,Unsprung_1p};
 F_Unsprung_2 = {'CF',-Unsprung_2m*g,Unsprung_2p};
 F_Gearbox = {'CF',-Gearbox_m(GB)*g,Gearbox_p};
 F_Axle = {'CF',-Axle_m*g,Axle_p};
 F_Body = {'CF',-Body_m*g,Body_p};
 F_Battery_1 = {'CF',-Battery_m(1)*g,Battery_p(1)};
 F_Battery_2 = {'CF',-Battery_m(2)*g,Battery_p(2)};
 F_Motor = {'CF',-Motor_m*g,Motor_p};
 F_Inverter = {'CF',-Inverter_m*g,Inverter_p};
 LengthSupp = [frameRailLength,ovr_hng,ovr_hng+wheelBase]/1000;
 [~,MaxValue] = SFBM('Group_9
Frame',LengthSupp,F_AM95,F_Cargo,F_Info,F_Sunroof,F_Unsprung_1,F_Unsprung_2,F_Gearbox,F_Axle,
F_Body,F_Battery_1,F_Battery_2,F_Motor,F_Inverter);
 end
```



Range	*Equations of Shear Force* :	*Equations of Bending Moment* :
0 *to* 0.56	0	0
0.56 *to* 0.61	-0.137	$-0.137x + 0.077$
0.61 *to* 1.1939	2.248	$2.248x - 1.382$
1.1939 *to* 1.91	2.101	$2.101x - 1.206$
1.91 *to* 1.9615	0.021	$0.021x + 2.77$
1.9615 *to* 2.024	-0.56	$-0.56x + 3.909$
2.024 *to* 2.096	-2.541	$-2.541x + 7.917$
2.096 *to* 2.874	-2.639	$-2.639x + 8.122$
2.874 *to* 3.21	-3.326	$-3.326x + 10.095$
3.21 *to* 3.385	2.109	$2.109x - 7.343$
3.385 *to* 3.51	1.668	$1.668x - 5.848$
3.51 *to* 3.82	0	0.003



From the above plots of Bending Moment and Shear Force, we are able to obtain the final dimension of our frame rail. We also get the weight distribution of the vehicle which is kept at 48% at front. This value is successfully achieved below from our packaging design.

```
 % CG calculation
 Component_mass =
[AM95_m,Cargo_m,Info_m,Sunroof_m,Unsprung_1m,Unsprung_2m,Gearbox_m(GB),Axle_m,Body_m,Battery_
m,Motor_m,Inverter_m];
 Component_dist =
[AM95_p,Cargo_p,Info_p,Sunroof_p,Unsprung_1p,Unsprung_2p,Gearbox_p,Axle_p,Body_p,Battery_p,Mo
tor_p,Inverter_p];
 Mass_Comp = sum(Component_mass);
 Fzf = 6.18;      % Front weight
 Fzr = 6.71;      % Rear Weight
 Fz = Fzf+Fzr;    % Overall Weight
 WD_front = round(Fzf/Fz,4);
```

```
fprintf('The weight distribution of Front is %.2f',WD_front)
```

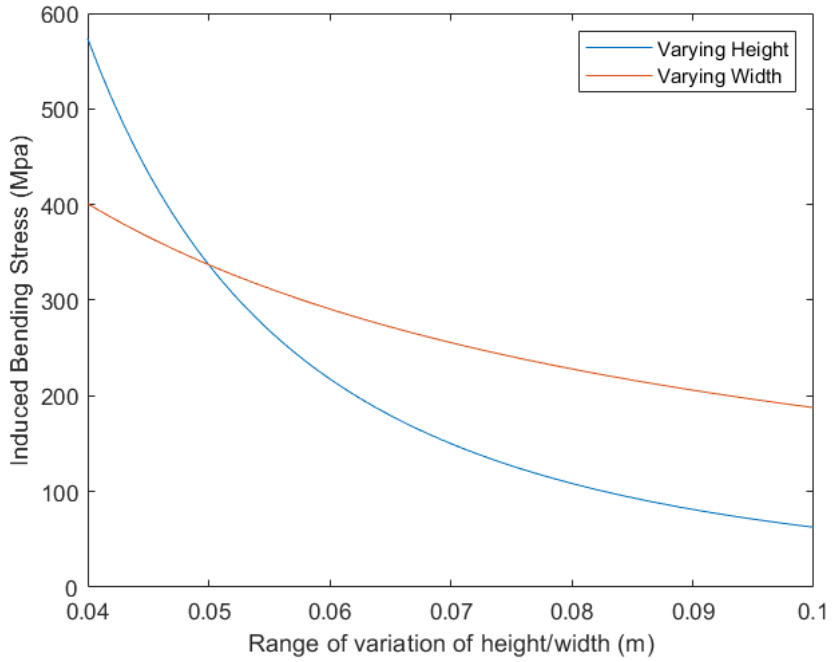
```
The weight distribution of Front is 0.48
```

Max Values of Shear Force and Bending Moment is displayed below. These are used for Stress and Deflection evaluation.

```
MaxSF = MaxValue(1);        % Max Shear Force
MaxBM = MaxValue(2);        % Max Bending Moment
fprintf('Max Shear force is %f KN, Max Bending Moment is %f kN-m',MaxSF,MaxBM)
```

```
Max Shear force is 3.384450 KN, Max Bending Moment is 2.811764 kN-m
```

```
h_sweep = .04:.001:.1;      % Sweep for height values
w_sweep = .04:.001:.1;      % Sweep for width values
% Varying height & width
t = .003;       % Initial thickness (m)
w = .05;        % Initial width (m)
h = .05;        % Initial height (m)
Iz_h = ((w*h_sweep.^3) - ((w-2*t)*(h_sweep-2*t).^3))/12;        % Iz for height change
Iz_w = ((w_sweep*h.^3) - ((w_sweep-2*t)*(h-2*t).^3))/12;        % Iz for width change
Max_Stress_h = (MaxBM*10^3*(w/2)./Iz_h)/10^6;       % Stress with varying height
Max_Stress_w = (MaxBM*10^3*(w/2)./Iz_w)/10^6;       % Stress with varying width
figure
plot(h_sweep,Max_Stress_h,w_sweep,Max_Stress_w)
xlabel('Range of variation of height/width (m)')
ylabel('Induced Bending Stress (Mpa)')
legend('Varying Height','Varying Width')
```



Inference from the graph above:

We can see that for a given constant value of thickness, changing any one parameter of the frame rail (height/width), effectively changes the induced stress in the frame. To have a general idea about the frames, the

minimum value for each iteration was 0.04 m (40 mm) up to 0.1 m (100 mm). To keep the deflection minimum, we need a higher Iz value, since deflection is inversely proportional to Iz (δ ∝ 1/Iz). Further Iz ∝ b*h^3. Thus, we need to have the maximum possible height relative to the width. On observing the graphs at the 0.06 m length mark, the height at 0.06 m gives the results closer to 215 MPa and can yield better deflection results. This thickness for this design is taken as 4mm.

```
h_fin = 60;     % Selected height of the frame rail
w_fin = 40;     % Selected width of the frame rail
t_fin = 4;      % Selected thickness of the frame rail
Iz_fin = ((w_fin*h_fin^3) - ((w_fin-2*t_fin)*(h_fin-2*t_fin)^3))/12;
SS_fin = (MaxBM*10^6*w_fin/2)/Iz_fin;
fprintf('The Bending Stress in the Frame Rail with %.1f mm height, %.1f mm width and %.1f mm thickness is %.2f MPa',h_fin,w_fin,t_fin,SS_fin)
```

```
The Bending Stress in the Frame Rail with 60.0 mm height, 40.0 mm width and 4.0 mm thickness is 162.98 MPa
```

```
syms len
Bend_eq(len) = -0.082*len + 3.101;      % Bending moment eq obtained from above
disp(len) = int(int(Bend_eq));          % Double integration of the eq
E = 210000;         % Modulus of Elasticity
deflection = double((disp(1.91)-disp(1.9695))*10^12/(E*Iz_fin));
fprintf('The deflection of this Frame Rail is %f mm (max)',deflection)
```

```
The deflection of this Frame Rail is -4.812655 mm (max)
```

```
rail_mass = (w_fin*h_fin - (w_fin-2*t_fin)*(h_fin-2*t_fin))*frameRailLength*7800*10^-9;
fprintf('The mass of each Frame Rail is %.2f kg',rail_mass)
```

```
The mass of each Frame Rail is 21.93 kg
```

```
Frame_mass = rail_mass*3;
fprintf('The mass of the Complete Frame is %.2f kg',Frame_mass)
```

```
The mass of the Complete Frame is 65.79 kg
```

```
Final_mass = Mass_Comp + Frame_mass;
Component_mass(14) = Frame_mass;
array2table(Component_mass','RowNames',{'AM95 Passenger (2 nos)','Cargo','Infotainment','Sunroof','Unsprung Mass (Front)','Unsprung Mass (Rear)','Gearbox (High Torque)',...
    'Axles (2 nos)','Body','Battery (Front)','Battery (Middle)','Motor A (2 nos)','Inverter (2 nos)','Frame (3*Rail)'},"VariableNames",{'Mass (kg)'})
```

```
ans = 14×1 table
```

	Mass (kg)
1 AM95 Passenger (2 nos)	202
2 Cargo	14
3 Infotainment	15
4 Sunroof	10
5 Unsprung Mass (Front)	120
6 Unsprung Mass (Rear)	120
7 Gearbox (High Torque)	45

	Mass (kg)
8 Axles (2 nos)	10
9 Body	212.0100
10 Battery (Front)	267.1875
11 Battery (Middle)	59.2125
12 Motor A (2 nos)	170
13 Inverter (2 nos)	70
14 Frame (3*Rail)	65.7896

```
fprintf('Final Total mass of the vehicle is %.2f kg',Final_mass)
```

```
Final Total mass of the vehicle is 1380.20 kg
```

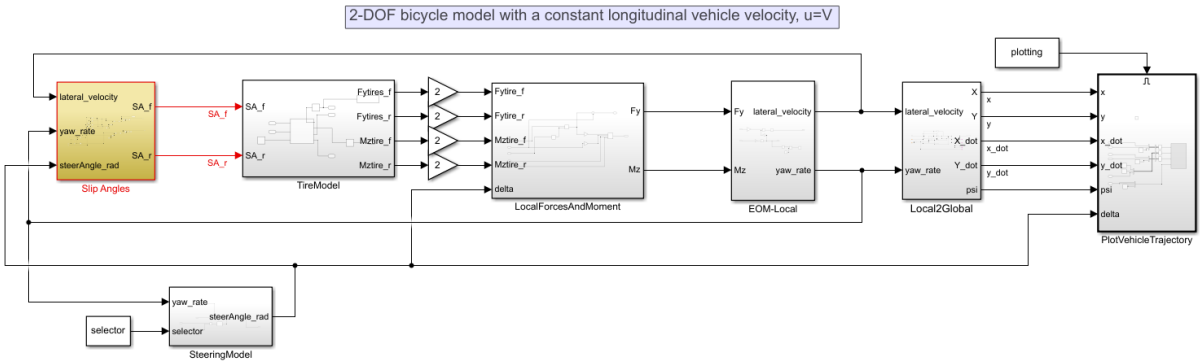
Final Specifications from packaging perspective:

- Frame Rail Length = 3820 mm
- Frame Rail Height = 60 mm
- Frame Rail Width = 40 mm
- Frame Rail Thickness = 4 mm
- Single Frame Rail Mass = 21.93 kg
- Total Frame Mass = 65.79 kg
- Body Mass = 212.01 kg
- Weight distribution = 48% Front
- CG height = 0.55 m above ground
- Angle of Approach = 12.03 deg
- Angle of Departure = 12.53 deg
- Min. Ramp Breakover Angle = 11.42 deg
- Upward Vision Cone Angle = 20.46 deg
- Downward Vision Cone Angle = 6.41 deg
- Overall Cargo Space = 0.48615 m^3 [> 100kg (0.3 m^3)]
- Front Trunk = .1621 m^3
- Rear Trunk = .3241 m^3
- Vehicle (l x w x h) = 3820 x 2020 x 1240 mm
- Total Mass of the Vehicle = 1380.20 kg

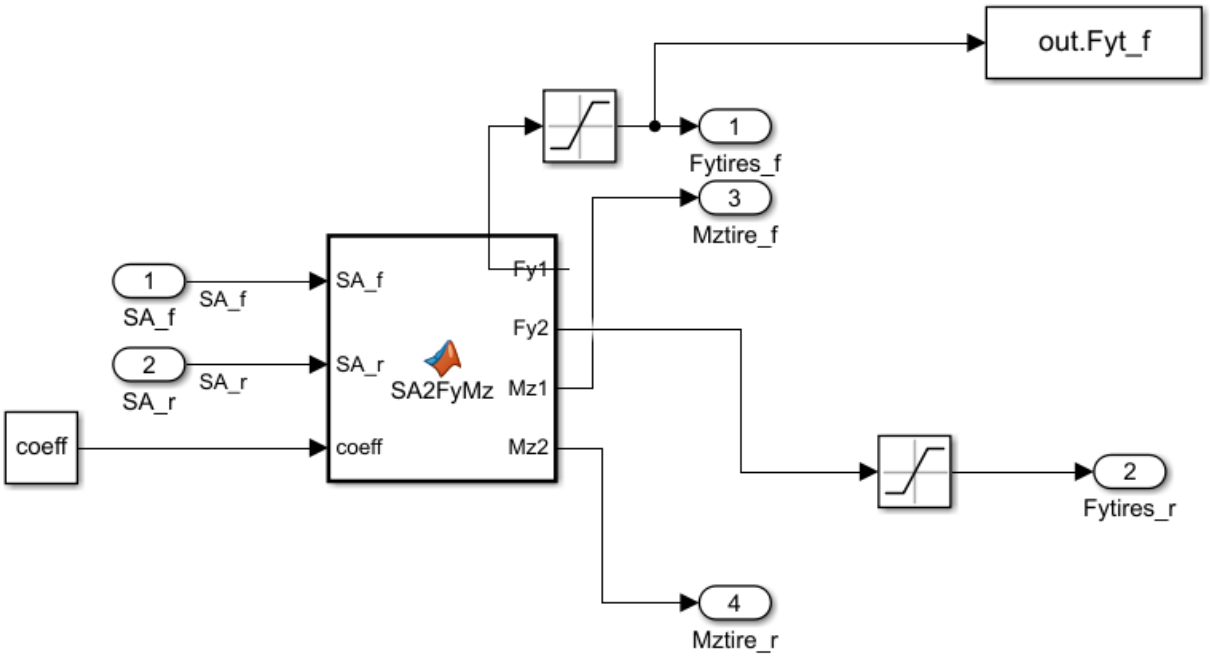
VEHICLE DYNAMICS MODEL: SKIDPAD REQUIREMENT:

Vehicle stability at high speed cornering is an really important criterion to be considered while designing a roadster. So, the plan here is to check if the vehicle is stable enough while doing a cornering at 0.8g lateral acceleration. The vehicle mass obtained from packaging of all the components is being used to test if the vehicle satisfies the required criterion. The moment of inertia about the vehicle inertial frame is assumed as 2380 Kg m^2 here. Also we have come up with a weight distribution from the design perspective which would ideally fit all the requirements of the vehicle. We have used the bicycle model given to us in the 4th assignment to verify if the requirement is satisfied. The vehicle front and rear tire lateral forces were saturated to a limit(traction limit of the vehicle) in order to get accurate results. The process is pretty much straight forward. We convert the given lateral acceleration in terms of m/s^2 and provide it as a input to the vehicle dynamics model to check if the PID controller provides corrected steering input in order to achieve the requirement.

Simulink model:

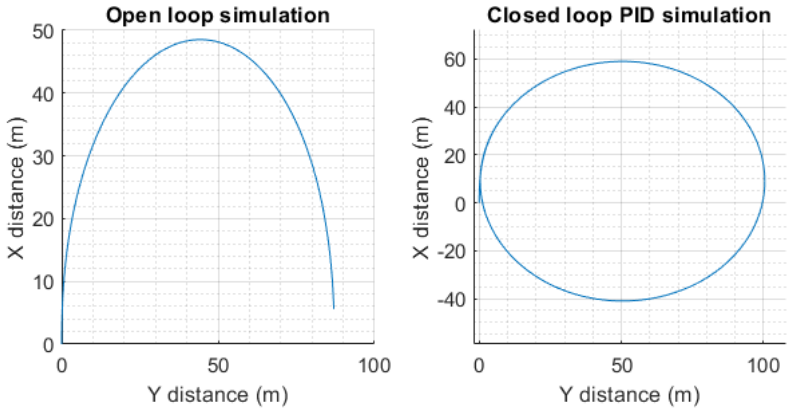


Lateral forces saturation:




```
mass = 1380.2;            % total vehicle mass [kg]
WD = 0.48;              % weight distribution [%] -- proportion on front axle
L = 2.6;              % wheelbase [m]
Jz = 2380;              % Moment of inertia of the vehicle [kg.m^2] %assumed
R = 6;                  % Turning radius
yaw_rate = 0.1;         % Yaw rate

a = (1-WD)*L;           % distance of CG from front
b = WD*L;               % distance of CG from rear
```

```
 Fzf = WD*mass*9.81;     % Weight on the front axle
 Fzr = (1-WD)*mass*9.81; % Weight on the rear axle
 [B1,C1,D1,E1,Bm1,Cm1,Dm1,Em1] = pacejka_coefficients(Fzf/(2*1000));     %Get pacejka coeff
for front single tire
 [B2,C2,D2,E2,Bm2,Cm2,Dm2,Em2] = pacejka_coefficients(Fzr/(2*1000));     %Get pacejka coeff
for front single tire
 coeff = [B1,C1,D1,E1,Bm1,Cm1,Dm1,Em1;B2,C2,D2,E2,Bm2,Cm2,Dm2,Em2];      % combine all coeff
in one matrix
 Rad = 50;                   % Radius of turn = 50 m
 ay = 0.8 *9.81 ; %lateral acceleration in m/s^2
 Vel = sqrt(ay.*Rad); %velocity in m/s
 Vkmph = Vel .* 3.6;    % velocity in kmph
 sat = mass.* 9.81 ; %saturation for lateral force
 simtime= 20;
 plotting = 0;       % disable plotting
 for(i=1)          % Run simulation in open and closed loop
     V = Vel(i);
     u = Vel(i);
     selector = 1;
     model(1,i) = sim("HW4_code_Jay_Saini.slx",simtime);
     selector = 0;
     model(2,i) = sim("HW4_code_Jay_Saini.slx",simtime);
 end
 figure
 axis square
 for(i=1)      % Plot open loop trajectory in first subplot
     subplot(1,2,1)
     hold on
     plot(model(1,i).pos_y.Data,model(1,i).pos_x.Data)
     grid on
     grid minor
     xlabel('Y distance (m)')
     ylabel('X distance (m)')
     axis square
 end
 title('Open loop simulation')
 for(i=1)      % Plot closed loop trajectory in second subplot
     subplot(1,2,2)
     hold on
     plot(model(2,i).pos_y.Data,model(2,i).pos_x.Data)
     grid on
     grid minor
     xlabel('Y distance (m)')
     ylabel('X distance (m)')
     axis square
 end
 title('Closed loop PID simulation')
 subplot(1,2,2)
 xlim([-2 108])
 ylim([-59 72])
```



Error in yaw rate = Reference yaw rate - Actual Yaw rate ; Reference Yaw rate = U/Circular path radius, where U is the longitudinal velocity. Actual Yaw is the import given in the model. From the error in the yaw rates and gains given inside the PID controller, we could find out the steering angle from the PID. Gain values are: Kp = 0.005 Ki = 0.3 Kd = 0.01 The saturation steering angle is the maximum steering angle. Here, in the open loop controller the vehicle does not fulfill the requirement and thus we have fixed with the closed loop control.

In the closed loop scenario, the vehicle tries to maintain its trajectory by following a circular path whenever the velocity increases constantly. The path along which the vehicle travels increases but the vehicle remains stable by maintaining its motion at expected trajectory without any unstable responses. The curve indicates that the vehicle tries to maintain its stability by getting the feedback from the PID controller(corrected steering angle). At higher mass this particular model fails to maintain minimum radius of turn. We had to alter the mass of the vehicle in order to fulfill this requirement.

Suspension spring constant and Suspension damping coefficient calculations:

Here, the suspension damped frequencies of the front and rear are given. We have assumed certain aspects of the parameters in order to achieve a proper value for the design variables. I have considered the motion ratio of front and rear to be 0.7 and 0.67 (from HONDAS2000). Also, we are assuming the same weight in both the front wheels and the same is applicable for the rear wheels. Since the damping ratio is 30 % of the critical damping, we use that to find ut the undamped natural frequency of the spring. We calculate the spring constant k from the formula, w = (1/2*pi) sqrt(k/m),where k is the spring constant and w is the undamped natural frequency.

Also, we have found out the wheel rate using the formula , wheel rate = spring rate * (mr^2) .

Source for this formula : http://performancetrends.com/Definitions/Wheel-Rate.htm

Source for Vehicle motion ratio data: https://robrobinette.com

```
rff = 1.2 ; %Front Ride Frequency(damped) in Hz
rfr = 1.44 ; %Rear Ride Frequency(damped) in Hz
mass = 1380.2 ; %vehicle mass in Kg
fwd = 0.48 ; %front weight distribution
rwd = 0.52 ; %rear weight distribution
motion_ratio_front = 0.7; %assumed from HondaS2000 roadster
```

```
motion_ratio_rear = 0.67; %assumed from HondaS2000 roadster
mf = (1400 * fwd)/2 - 60 ; %vehicle sprung mass front (per wheel)
mr = (1400 * rwd)/2 - 60; %vehicle sprung mass rear (per wheel)

%we know that damping cofficient is 30 % of critical damping
%thus damping ratio is 0.3
dr = 0.3;
rffud = rff./sqrt(1-dr.^2); %undamped frequency front  in Hz
rfrud = rfr./sqrt(1-dr.^2); %undamped frequency rear in Hz
%From the known values we can find out the value of spring constants for
%the front and rear
kf = rffud.^2 .* mf.* (4.*(pi.^2)); %spring rate front in N/m
fprintf('The spring constant for one front wheel is %f N/m ',kf);
```

```
The spring constant for one front wheel is 17242.090432 N/m
```

```
kr = rfrud.^2 .* mr.* (4.*(pi.^2));  %spring rate rear in N/m
fprintf('The spring constant for one rear wheel is %f N/m ',kr);
```

```
The spring constant for one rear wheel is 27347.454737 N/m
```

```
 % we assume that both the front wheels have same spring constants and
 % damping coefficients.This condition is applicable to rear wheels too.
wheel_rate_front = kf.*(motion_ratio_front.^2); %wheel rate-front in N/m
wheel_rate_rear = kr.*(motion_ratio_rear.^2); %wheel rate-rear in N/m

cdcf = 2.*sqrt(wheel_rate_front.*mf); % criticaldamping coefficient rear in Ns/m
dcf = 0.3 .* cdcf; %damping coefficient front  in Ns/m
fprintf('The suspension damping coefficient for one front wheel is %f Ns/m ',dcf);
```

```
The suspension damping coefficient for one front wheel is 916.217939 Ns/m
```

```

cdcr = 2.*sqrt(wheel_rate_rear.*mr); % criticaldamping coefficient rear in Ns/m
dcr = 0.3 .* cdcr; %damping coefficient front  in Ns/m
fprintf('The suspension damping coefficient for one rear wheel is %f Ns/m ',dcr);
```

```
The suspension damping coefficient for one rear wheel is 1159.101055 Ns/m
```

Final Profit of the vehicle:

```
T.("Mass (Kg)")(2) = Final_mass;
T(2,:)
```

```
ans = 1×9 table
```

	Motor	Config	Number of Motors	Profit ($Million)	Mass (Kg)	...
1	A	RWD	2	1.6295e+03	1.3802e+03	

%% Functions

```
function plotSimulationResults(simOut)
    close all;

    % Vehicle Speed plots
    figure;
    plot(simOut.desiredSpeed.Time, simOut.desiredSpeed.Data, 'b');
    hold on;
    plot(simOut.vehicleSpeed.Time, simOut.vehicleSpeed.Data, 'g');
    xlabel('Time (s)');
    ylabel('Speed (km/h)');
    title('Vehicle Speed vs Time');
    legend({'Drive Cycle', 'Actual Speed'});
    hold off;

    % Traction Force and traction limit plots
    figure;
    plot(simOut.tractionForce.Time, simOut.tractionForce.Data * 0.001, 'b');
    hold on;
    plot(simOut.tractionForce.Time, ones(size(simOut.tractionForce.Time)) *
simOut.accTracLimit.Data * 0.001, 'r');
    plot(simOut.tractionForce.Time, ones(size(simOut.tractionForce.Time)) *
simOut.decTracLimit.Data * 0.001, 'y');
    xlabel('Time (s)');
    ylabel('Force (kN)');
    title('Traction Force vs Time');
    legend({'Traction Force', 'Acceleration Traction Limit', 'Deceleration Traction
Limit'});
    hold off;

    % Motor Torque Plot
    figure;
    plot(simOut.motTorque.Time, simOut.motTorque.Data);
    xlabel('Time (s)');
    ylabel('Torque (Nm)');
    title('Motor Torque vs Time');

    % Power Plot
    figure;
    plot(simOut.power.Time, simOut.power.Data * 0.001);
    xlabel('Time (s)');
    ylabel('Power (kW)');
    title('Net Power vs Time');
end

function [massVehicle, cost, range, toHundredTime, topSpeed, suitability] =
calculateValueParameters(mSel, isRWD, numMotors)
    global cVehicle cAxle cGear mVehicle mAxle mGear sim_time followDriveCycle grade
drivingBattCap EM_Efficiency EM_Torque_Map EM_Omega_Map EM_Torque_Max EM_Omega_Max
contPowerRating;
```

```
% Loading motor data
if mSel == 'A'
    load MotorMaps\MotorA_Data.mat;
    cMotor = 7500;
    mMotor = 85+35;
    contPowerRating = 175*1000;
elseif mSel == 'B'
    load MotorMaps\MotorB_Data.mat;
    cMotor = 5200;
    mMotor = 41+16;
    contPowerRating = 70*1000;
elseif mSel == 'C'
    load MotorMaps\MotorC_Data.mat;
    cMotor = 5500;
    mMotor = 55+12;
    contPowerRating = 100*1000;
end

% Data as per Number of Motors
EM_Torque_Max = EM_Torque_Max * numMotors;
EM_Torque_Map = EM_Torque_Map * numMotors;
contPowerRating = contPowerRating * numMotors;
cMotor = cMotor * numMotors;
mMotor = mMotor * numMotors;

originalmVehicle = mVehicle;    % Storing value for reverting mVehicle after test
originalcVehicle = cVehicle;    % Storing value for reverting cVehicle after test
mVehicle = mVehicle + mMotor;
cVehicle = cVehicle + cMotor;
if isRWD == 0
    mVehicle = mVehicle + mAxle + mGear;
    cVehicle = cVehicle + cAxle + cGear;
end
massVehicle = mVehicle;
cost = cVehicle;

% Check if vehicle can reach 130kmph at 12% slope
followDriveCycle = 0;
grade = 0.12;
gradeSim = sim('ASO_Project_Powertrain_model.slx', sim_time);
%plotSimulationResults(gradeSim);
if max(gradeSim.vehicleSpeed.Data) >= 130
    suitability = 1;
else
    suitability = 0;
    %fprintf("Setup not suitable to reach 130 kmph at 0.12 slope\n\n");
end

% Run simulation for Drive Cycle
followDriveCycle = 1;
```

```
    grade = 0;
    simOut = sim('ASO_Project_Powertrain_model.slx', sim_time);
    %plotSimulationResults(simOut);

    % Range calculation for battery size
    energyUsed = simOut.energyConsumed.Data(end);   % in kWh
    distance = simOut.distance.Data(end) * 0.001;   % in Km
    range = drivingBattCap * distance / energyUsed;

    % Run Simulation for WOT
    followDriveCycle = 0;
    wotSim = sim('ASO_Project_Powertrain_model.slx', 60);
    %plotSimulationResults(wotSim);
    toHundredTime = wotSim.vehicleSpeed.Time(find(wotSim.vehicleSpeed.Data >= 100, 1));
    topSpeed = wotSim.vehicleSpeed.Data(end);

    maxMotorTorque = max(simOut.motTorque.Data);
    if maxMotorTorque <= 300
        fprintf('Can use low speed gearbox for Motor %c RWD=%d numM=%d', mSel, isRWD,
numMotors);
    end

    % Reverting values for next test
    mVehicle = originalmVehicle;
    cVehicle = originalcVehicle;
end

function performProfitAnalysis(mSel, isRWD, numMotors)
    global cVehicle cAxle cGear mVehicle mAxle mGear sim_time followDriveCycle grade
drivingBattCap EM_Efficiency EM_Torque_Map EM_Omega_Map EM_Torque_Max EM_Omega_Max
contPowerRating profit mass cost range toHundredTime topSpeed index;

    [massVal costVal rangeVal toHundredTimeVal topSpeedVal suitableVal] =
calculateValueParameters(mSel, isRWD, numMotors);
    if suitableVal == 1
        profitVal = profitPredict([costVal; rangeVal; toHundredTimeVal; topSpeedVal]);
    else
        profitVal = 0;
    end
    profit(index) = profitVal;
    mass(index) = massVal;
    cost(index) = costVal;
    range(index) = rangeVal;
    toHundredTime(index) = toHundredTimeVal;
    topSpeed(index) = topSpeedVal;
    index = index + 1;
end
```