ME660

Rail Road Vehicle Dynamics Combined Report

Group Members:

Rajesh Mishra (150557)

Tarun Sharma (150764)

Vaibhav Raj Singh (150788)

Pushpendra Singh (14807510)

ME660

Rail Road Vehicle Dynamics Assignment 1 Report

Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Structure of the Code:

1. **“profileR1.m”** –to get the initial point of contact between the rail and wheel profiles and finally getting the polynomial fit with origin at initial point of contact. This is done by fixing the flange gap 8mm and then moving the wheel profile to get the point of contact (reducing the minimum distance between the two profiles).

```
%rail
clc; clear all;
P1=[-33.04,-6.64];P2=[-26.05,-2.30482898]; P0=[-23.03833333,-14.95116695];
[x,y]=arc(P1,P2,P0);

P1=P2;P2=[-10.25,-0.1751553]; P0=[-7.51666667,-80.12844722];
[a,b]=arc(P1,P2,P0);
x=[x,a];y=[y,b];
P1=P2;P2=[0,0]; P0=[0,-300];
[a,b]=arc(P1,P2,P0);
rail=[[x,a];[y,b]];

%wheel
P2=[-34.52650895,-0.18119552];P1=[-40.27,-3.23]; P0=[-31.02344007,-
13.73584448];
[x,y]=arc(P1,P2,P0);
P1=P2;P2=[-22.8367359,2.10716785]; P0=[-9.50458834,-97.00011665];
[a,b]=arc(P1,P2,P0);
x=[x,a];y=[y,b];

P1=P2;P2=[-0.48400193,3.98815518]; P0=[15.99541166,-325.60011665];
[a,b]=arc(P1,P2,P0);
wheel=[[x,a];[y,b]];
global zeta_rP zeta_wP ;
nrr=rail(1,:);  zrr=rail(2,:);
f = polyfit(nrr,zrr,9);
zeta_rP=poly2sym(f);
nwr=wheel(1,:);  zwr=wheel(2,:);
f = polyfit(nwr,zwr,9);
zeta_wP = poly2sym(f);

%%Vertical Distance Minimization
func = -(zeta_rP - zeta_wP);
x = -25:0.01:-5;
distance = subs(func);
plot(x,distance);
[minimum,i] = min(distance);
x_contact = x(i)
y_contact = minimum
%%Repeat
rail(1,:)=rail(1,:)-(x_contact);rail(2,:)=rail(2,:);
wheel(1,:)=wheel(1,:)-(x_contact);wheel(2,:)=wheel(2,:);
nwr=wheel(1,:);  zwr=wheel(2,:);
f = polyfit(nwr,zwr,9);
zeta_wP = poly2sym(f)-y_contact;
nrr=rail(1,:);  zrr=rail(2,:);
f = polyfit(nrr,zrr,9);
zeta_rP=poly2sym(f);
```

ME660

```
%%Update
x=0;
V_shift = subs(zeta_wP);
zeta_rP = zeta_rP - V_shift;
zeta_wP = zeta_wP - V_shift;
%plot
fplot(zeta_rP,[-10,10]);
hold on;grid on;
fplot(zeta_wP,[-10,10]);

delta_w = atan(diff(zeta_wP));
delta_r = atan(diff(zeta_rP));
```

2. **"solve1"** – to write 14 equations.

```
function F = solve1(x)
global Uy
global rad yr
% r0=500;
% y0=0;
r0=rad; y0=yr;
L=1100;
%Wheel and rail (Right)
F(1) = x(3) - (5139597554770761*(-x(1))^9)/618970019642690137449562112 -
(978272914433037*(-x(1))^8)/2417851639229258349412352 - (541800308747879*(-
x(1))^7)/19342813113834066795298816 + (7169221819261939*(-
x(1))^6)/37778931862957161709568 + (4089400285490287*(-
x(1))^5)/4722366482869645213696 - (2387928271314221*(-
x(1))^4)/73786976294838206464 - (578189731934337*(-
x(1))^3)/9223372036854775808 - (2125149540870471*(-
x(1))^2)/73786976294838206464 + (863925489082307*(-x(1)))/9007199254740992;
F(2) = x(4) - (495229314623629*(-x(2))^9)/38685626227668133590597632 +
(7142424370555727*(-x(2))^8)/38685626227668133590597632 +
(5288588683557781*(-x(2))^7)/604462909807314587353088 - (840349606378163*(-
x(2))^6)/4722366482869645213696 - (8088071278431591*(-
x(2))^5)/9444732965739290427392 + (4912984140153015*(-
x(2))^4)/147573952589676412928 + (2790363789710575*(-
x(2))^3)/36893488147419103232 - (4315180088559653*(-
x(2))^2)/5764607523034234880 + (6911206147709207*(-x(2)))/72057594037927936 -
86020444518536962733730930817909/15474250491067253436239052800000000000000000
0;
%Wheel and rail (Left)
F(3) = x(7) - (5139597554770761*(-x(5))^9)/618970019642690137449562112 -
(978272914433037*(-x(5))^8)/2417851639229258349412352 - (541800308747879*(-
x(5))^7)/19342813113834066795298816 + (7169221819261939*(-
x(5))^6)/37778931862957161709568 + (4089400285490287*(-
x(5))^5)/4722366482869645213696 - (2387928271314221*(-
x(5))^4)/73786976294838206464 - (578189731934337*(-
x(5))^3)/9223372036854775808 - (2125149540870471*(-
x(5))^2)/73786976294838206464 + (863925489082307*(-x(5)))/9007199254740992;
F(4) = x(8) - (495229314623629*(-x(6))^9)/38685626227668133590597632 +
(7142424370555727*(-x(6))^8)/38685626227668133590597632 +
(5288588683557781*(-x(6))^7)/604462909807314587353088 - (840349606378163*(-
x(6))^6)/4722366482869645213696 - (8088071278431591*(-
x(6))^5)/9444732965739290427392 + (4912984140153015*(-
x(6))^4)/147573952589676412928 + (2790363789710575*(-
```

```
x(6))^3)/36893488147419103232 - (4315180088559653*(-
x(6))^2)/576460752303423488 + (6911206147709207*(-x(6)))/72057594037927936 -
860204445185369627337309308179O9/15474250491067253436239052800000000000000000
0;
%Delta equations
F(5) = -x(9)  + atan((46256377992936849*x(1)^8)/618970019642690137449562112 -
(978272914433037*x(1)^7)/302231454903657293676544 +
(3792602161235153*x(1)^6)/19342813113834066795298816 +
(21507665457785817*x(1)^5)/18889465931478580854784 -
(20447001427451435*x(1)^4)/4722366482869645213696 -
(2387928271314221*x(1)^3)/18446744073709551616 +
(1734569195803011*x(1)^2)/9223372036854775808 -
(2125149540870471*x(1))/36893488147419103232 -
863925489082307/9007199254740992);
F(6) = -x(10) + atan((4457063831612661*x(2)^8)/38685626227668133590597632 +
(7142424370555727*x(2)^7)/4835703278458516698824704 -
(37020120784904467*x(2)^6)/6044629098073145873530 88 -
(2521048819134489*x(2)^5)/2361183241434822606848 +
(40440356392157955*x(2)^4)/9444732965739290427392 +
(4912984140153015*x(2)^3)/36893488147419103232 -
(8371091369131725*x(2)^2)/36893488147419103232 -
(4315180088559653*x(2))/288230376151711744 -
6911206147709207/72057594037927936);
F(7) = -x(11) + atan(-(- (46256377992936849*(-
x(5))^8)/618970019642690137449562112 - (978272914433037*(-
x(5))^7)/302231454903657293676544 - (3792602161235153*(-
x(5))^6)/19342813113834066795298816 + (21507665457785817*(-
x(5))^5)/18889465931478580854784 + (20447001427451435*(-
x(5))^4)/4722366482869645213696 - (2387928271314221*(-
x(5))^3)/18446744073709551616 - (1734569195803011*(-
x(5))^2)/9223372036854775808 - (2125149540870471*(-
x(5)))/36893488147419103232 + 863925489082307/9007199254740992));
F(8) = -x(12) + atan(-(- (4457063831612661*(-
x(6))^8)/38685626227668133590597632 + (7142424370555727*(-
x(6))^7)/4835703278458516698824704 + (37020120784904467*(-
x(6))^6)/6044629098073145873530 88 - (2521048819134489*(-
x(6))^5)/2361183241434822606848 - (40440356392157955*(-
x(6))^4)/9444732965739290427392 + (4912984140153015*(-
x(6))^3)/36893488147419103232 + (8371091369131725*(-
x(6))^2)/36893488147419103232 - (4315180088559653*(-x(6)))/288230376151711744
+ 6911206147709207/72057594037927936));

%Right side equations
F(9) = Uy - y0 - r0*x(13) - x(1) + x(2);
F(10) = x(14) + L*x(13) + x(3) - x(4);
F(11) = -x(13) + x(9) - x(10);

%Left side equations
F(12) = Uy - y0 - r0*x(13) + x(5) - x(6);
F(13) = x(14) - L*x(13) + x(7) - x(8);
F(14) = -x(13) - x(11) + x(12);

F = double(F);

End
```

ME660

3. **"Plots.m"** – GUI for calculating the solution and showing the plots.

```
function varargout = Plots(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Plots_OpeningFcn, ...
                   'gui_OutputFcn',  @Plots_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Plots_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = Plots_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function pushbutton1_Callback(hObject, eventdata, handles)
global rad; global yr;

rad=str2double(get(handles.edit1,'string'));
yr=str2double(get(handles.edit2,'string'));
Solution;
function pushbutton2_Callback(hObject, eventdata, handles)
global sol; global ax;
axes(handles.axes1)
cla reset
plot(ax,sol(:,13));
grid on
axes(handles.axes2)
cla reset
plot(ax,sol(:,14));
grid on
axes(handles.axes3)
```

```
cla reset
plot(ax,sol(:,1)); hold on; plot(ax,sol(:,5))
grid on
axes(handles.axes4)
cla reset
plot(ax,sol(:,2)); hold on; plot(ax,sol(:,6))
grid on
axes(handles.axes5)
cla reset
plot(ax,sol(:,3)); hold on; plot(ax,sol(:,7))
grid on
axes(handles.axes6)
cla reset
plot(ax,sol(:,4)); hold on; plot(ax,sol(:,8))
grid on
```

4. **“newton.m”** is the code for multivariable newton-rephson method to solve 14 equations

```
function x=newton(fname,x)
f0=feval(fname,x);
n=length(x);
count=0;

while (norm(f0) > 1e-12*max(1,norm(x)))*(count<60000)
    epsil=1e-2;
    E=eye(n)*epsil;

    D=E; % initialization; will be overwritten
    for k=1:n
        temp=feval(fname,plus(x',E(:,k)));
        D(:,k)=(temp-f0)/epsil;
    end

    x=(x'-D\f0')';
    f0=feval(fname,x);
    count=count+1;
end

if count >=60000, x=inf; end
```

5. **“Solution.m”** – to solve the 14 equations using “fsolve” function of MATLAB. Conversion of equation to “f(x) =0” form is required.

```
clear; clc;
global sol; global ax
sol = zeros(101,14);
j=0;
global Uy;

for i=-5:0.1:5
j=j+1
Uy = i;
uy(j,1)=i;
fun = 'solve1';
```

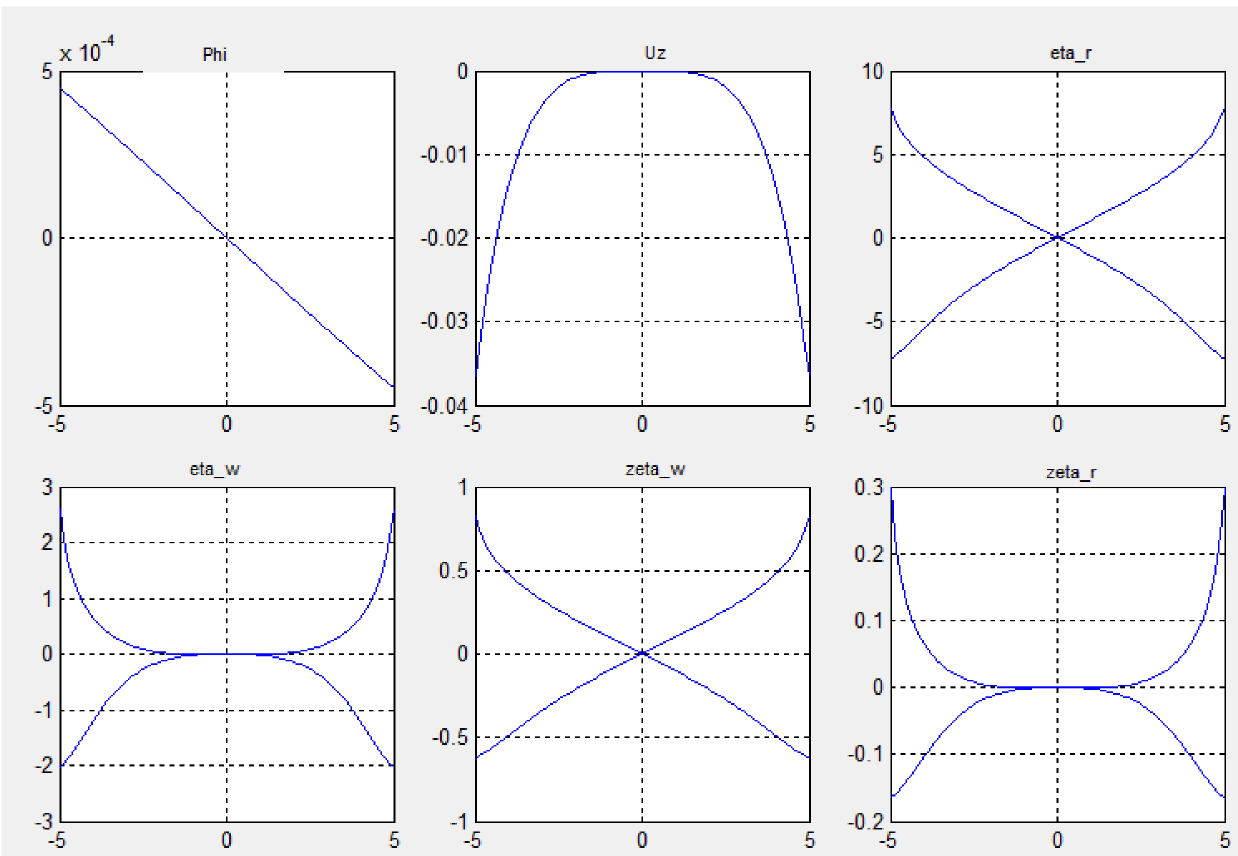
ME660

```
x0=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
x = fsolve(fun,x0);
sol(j,:) = x;

end
ax=uy;
```

How to run the code:

1. Run “Plots.m”
Here, r0=500mm and y0=0mm is taken as default value (can be changed using GUI)
2. Click on “solve” button. Waits for code run time (approx. 20 seconds)
3. Click on “show plots” to see the plots.

Plots:



Rail Road Vehicle Dynamics

Assignment 2 Report

Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Structure of the Code, Question 2:

"Ques_2_1.m": Solving the equations of motion given in Q2 to obtain the critical speed, by solving the Eigenvalue problem

1. **Defining Constants:**

```
V = 0.1:0.1:250; %In meter/sec
m=1250;
Iz= 700;
Iy = 250;
W = 78.48*1000;
ky = 0.23*10^6;
kshi = 2.5*10^6;
cy = 0;
cshi = 0;
ro = 0.45;
l = 0.7452;
lambdao = 0.1174;
epshilao = 6.423;
deltao = 0.0493;
sigma = 0.0508;
f11 = 7.44*10^6;
f22 = 6.79*10^6;
f23 = 13.7*10^3;
%Taken from slide (not given in the ques.)
No = 39340; %normal force
k = deltao*(1 - f23/(No*ro));
```

2. **Equation for Ky and Kshi:**

```
Ky = ky + (2*No*epshilao/l)*(1 - f23/(No*ro));
Kshi = kshi + (2*No*l)*(-deltao +f23/(No*l));
```

3. **Solution for Equations of Motion:**

```
for i=1:length(V)
a2(i) = m;
a1(i) = 2*f22/V(i);
a0(i) = Ky;
ba1(i) = (2*f23/V(i)) - (Iy*k*V(i))/(ro*l);
ba0(i) = -2*f22;

b2(i) = Iz;
b1(i) = 2*f11*l*l/V(i);
b0(i) = Kshi;
ab1(i) = -( 2*f23/V(i) -  Iy*deltao*V(i)/(ro*l) );
ab0(i) = 2*f11*lambdao*l/ro;

%polynomial Constants
P4(i) = -a2(i)*b2(i);
P3(i) = -a1(i)*b2(i) -a2(i)*b1(i);
P2(i) = -a0(i)*b2(i) -a1(i)*b1(i) -a2(i)*b0(i) + ab1(i)*ba1(i);
P1(i) = -a0(i)*b1(i) -a1(i)*b0(i) +ab1(i)*ba0(i) +ab0(i)*ba1(i);
P0(i) = -a0(i)*b0(i) + ab0(i)*ba0(i);

%roots
p = [P4(i) P3(i) P2(i) P1(i) P0(i)];
```

```
s(:,i) = roots(p);
s_Real(:,i) = real(s(:,i));
s_Imag(:,i) = imag(s(:,i));
end
```

4. **Plot:** Eigen Value plot for different values of Velocity(V)

```
%% Plots
plot(s_Real(1,:),s_Imag(1,:));
hold on
plot(s_Real(2,:),s_Imag(2,:));
plot(s_Real(3,:),s_Imag(3,:));
plot(s_Real(4,:),s_Imag(4,:));
grid on
axis([-700 200 -80 80])
title('Eigen Values plot for different values of Velocity (V)')
ylabel('Imag') % x-axis label
xlabel('Real') % y-axis label
legend('s1','s2','s3','s4');
```

Maple calculation to get the coefficients of polynomial:

$$exp1 := a2 \cdot s^2 \cdot Y + a1 \cdot s \cdot Y + ba1 \cdot s \cdot \mathrm{psi} + ba0 \cdot \mathrm{psi} + a0 \cdot Y = 0$$

$$exp1 := a2\, s^2\, Y + a1\, s\, Y + ba1\, s\, \psi + a0\, Y + ba0\, \psi = 0 \tag{1}$$

$$exp2 := b2 \cdot s^2 \cdot \mathrm{psi} + b1 \cdot s \cdot \mathrm{psi} + ab1 \cdot s \cdot \mathrm{psi} + ab0 \cdot \mathrm{psi} + b0 \cdot Y = 0$$

$$exp2 := b2\, s^2\, \psi + ab1\, s\, \psi + b1\, s\, \psi + b0\, Y + ab0\, \psi = 0 \tag{2}$$

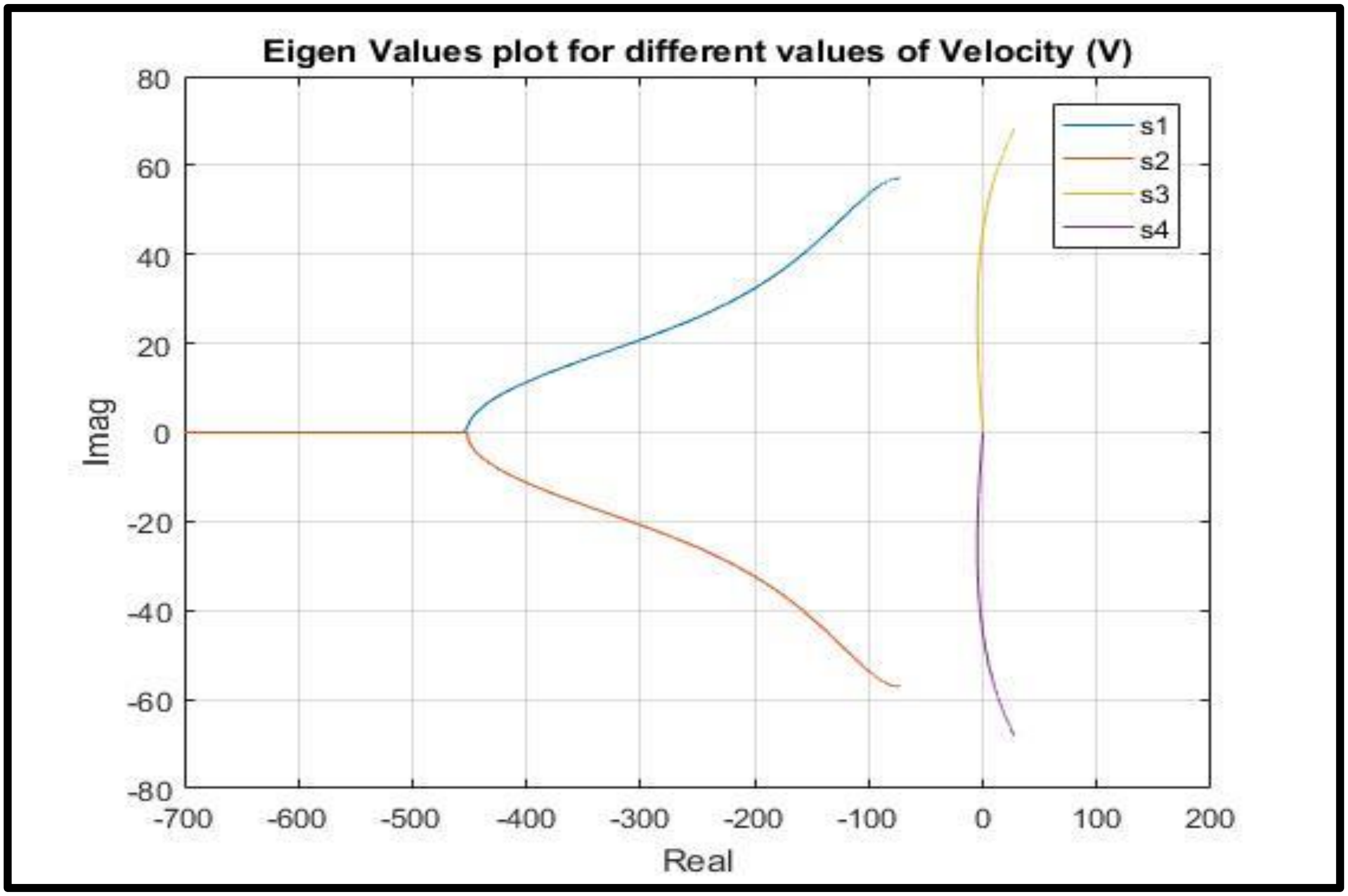
$$eliminate(\{exp1, exp2\}, Y)$$

$$\left[\left\{Y = -\frac{\psi\left(b2\, s^2 + ab1\, s + b1\, s + ab0\right)}{b0}\right\}, \left\{-\psi\left(a2\, b2\, s^4 + a1\, b2\, s^3 + a2\, ab1\, s^3 + a2\, b1\, s^3 + a0\, b2\, s^2 + a1\, ab1\, s^2 + a1\, b1\, s^2 + a2\, ab0\, s^2 + a0\, ab1\, s + a0\, b1\, s + a1\, ab0\, s - ba1\, s\, b0 + a0\, ab0 - ba0\, b0\right)\right\}\right] \tag{3}$$

How to run the code:

1. Run "**Ques_2_1.m**"
2. Click on "**run button**", wait for the code run time (approx. 5 seconds)
3. Open the figures to see the **Eigen Value plot for different values of Velocity(V)**

Plot:


Eigen Values plot for different values of Velocity (V)
s1
s2
s3
s4
Imag
Real
80
60
40
20
0
-20
-40
-60
-80
-700
-600
-500
-400
-300
-200
-100
0
100
200

Rail Road Vehicle Dynamics

Assignment 3 Report

Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Structure of the Code, Question 3:

"Question_3.m": Solving the equations of motion for the two-axle rail vehicle given in Q3 to obtain the critical speeds, by solving the Eigenvalue problem

1. **Defining Constants:**

```
m = 1250;
I_z = 700;
I_y = 250;
W = 78.48*(10^3);
k_y = 0.23*(10^6);
k_psi = 2.5*(10^6);
c_y = 0;
c_psi =0;
r_o = 0.45;
l = 0.7452;
lambda_o = 0.1174;
epsilon_o_star = 6.423;
delta_o = 0.0493;
sigma = 0.0508;
f11 = 7.44*(10^6);
f22 = 6.79*(10^6);
f23 = 13.7*(10^3);
k_phi = 1*(10^6);
h = 3.7;
d = 0.2;
I = 700;
m_b = 13500;
I_xb = 161000;
I_zb = 170000;
I_yb = 250;

N_o = 39340; %normal force
kappa = delta_o*(1 - f23/(N_o*r_o));
V = 0.1:0.1:100; %In meter/sec
```

2. **Solving Equations of Motion:**

```
for i=1:length(V)


%X1 = [y_1,psi_1,y_b,phi_b,psi_b,y_2,psi_2];
M1 = diag([m,I_z,m_b,I_xb,I_zb,m,I_z],0);
C1 = [2*f22/V(i),    (2*f23/V(i)-I_y*kappa*V(i)/(r_o*l)),    0,    0,    0,    0,         0;
      -(2*f23/V(i)-I_y*delta_o*V(i)/(r_o*l)), 2*f11*l*l/V(i),  0,    0,    0,    0,      0;
      0,             0,            0,   0,    0,    0,                                 0;
      0,             0,            0,   0,    0,    0,                                 0;
      0,             0,            0,   0,    0,    0,                                 0;
      0,             0,     0,    0,    0,    2*f22/V(i),        (2*f23/V(i)-I_y*kappa*V(i)/(r_o*l)) ;
      0,             0,     0,    0,    0,   -(2*f23/V(i)-I_y*delta_o*V(i)/(r_o*l)), 2*f11*l*l/V(i)];



 K1 = [2*N_o*epsilon_o_star/l*(1-f23/(N_o*r_o))+k_y, -2*f22, -k_y , k_y*d , -k_y*h , 0, 0 ;
      2*f11*lambda_o*l/r_o , 2*N_o*l*(-delta_o + f23/(N_o*l))+k_psi, 0  , 0  , -k_psi , 0 , 0;
      -k_y  , 0  , 2*k_y  , -2*k_y*d  , 0 , -k_y      , 0  ;
      k_y*d  , -k_psi  , 2*k_y*d, 2*k_y*d*d+2*k_phi, 0  , k_y*d, 0 ;
      -k_y*h  , -k_psi  , 0      , 0  , 2*k_y*h*h+2*k_psi, k_y*h , k_psi ;
      0  ,  0   , -k_y   , k_y*d , -k_y*h , 2*N_o*epsilon_o_star/l*(1-f23/(N_o*r_o))+k_y , -2*f22;
      0   , 0 , 0 , 0 , -k_psi , 2*f11*lambda_o*l/r_o , 2*N_o*1*(-delta_o + f23/(N_o*1))+k_psi];
```

```
A= [zeros(7, 7), eye(7); -inv(M1)*K1, -inv(M1)*C1];
[eigvec,eigval]=eig(A);
s_Real(:,i) = real(diag(eigval));
s_Imag(:,i) = imag(diag(eigval));
end

for j=1:14
  H(j) = max(find(s_Real(j,:)<0));
end

Stable_Velocity = V(min(H(1:10)))*18/5;% Answer(in Kmph)    %ignoring s11 to
s14 poles because they are oscillating and not dominant
```

3. **Plot:** Eigen Value plot for different values of Velocity(V)

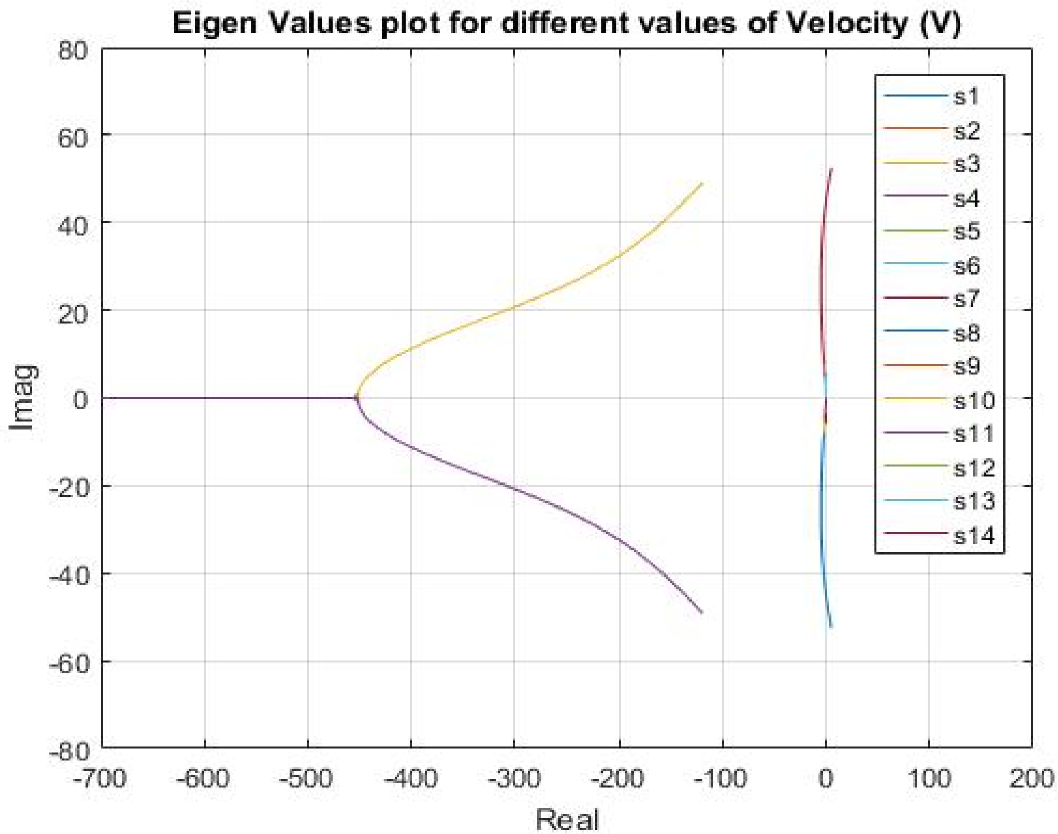
```
%% Plots
plot(s_Real(1,:),s_Imag(1,:));
hold on
plot(s_Real(2,:),s_Imag(2,:));
plot(s_Real(3,:),s_Imag(3,:));
plot(s_Real(4,:),s_Imag(4,:));
plot(s_Real(5,:),s_Imag(5,:));
plot(s_Real(6,:),s_Imag(6,:));
plot(s_Real(7,:),s_Imag(7,:));
plot(s_Real(8,:),s_Imag(8,:));
plot(s_Real(9,:),s_Imag(9,:));
plot(s_Real(10,:),s_Imag(10,:));
plot(s_Real(11,:),s_Imag(11,:));
plot(s_Real(12,:),s_Imag(12,:));
plot(s_Real(13,:),s_Imag(13,:));
plot(s_Real(14,:),s_Imag(14,:));

grid on
axis([-700 200 -80 80])
title('Eigen Values plot for different values of Velocity (V)')
ylabel('Imag') % x-axis label
xlabel('Real') % y-axis label
legend('s1','s2','s3','s4','s5','s6','s7','s8','s9','s10','s11','s12','s13','s1
4');
```

How to run the code:

1. Run "**Question_3.m**"
2. Click on "**run button**", wait for the code run time (approx. 5 seconds)
3. Open the figures to see the **Eigen Value plot for different values of Velocity(V)**

Results: Stable_Velocity = 282.6000

Plot:


Eigen Values plot for different values of Velocity (V)
80
60
40
20
0
-20
-40
-60
-80
-700
-600
-500
-400
-300
-200
-100
0
100
200
Real
Imag
s1
s2
s3
s4
s5
s6
s7
s8
s9
s10
s11
s12
s13
s14

Rail Road Vehicle Dynamics

Assignment 4 Report

Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Introduction

Linear set of equations:-

$$m\ddot{y} + \frac{2f_{22}}{V}\dot{y} + \Big(\frac{2f23}{V} - \frac{I_y\kappa V}{r_o l}\Big)\dot{\psi} - 2f_{22}\psi + K_y y = Q_y \tag{1}$$

$$I_z\ddot{\psi} + \frac{2f_{11}l^2}{V}\dot{\psi} - \Big(\frac{2f23}{V} - \frac{I_y\delta_o V}{r_o l}\Big)\dot{y} + \frac{2f_{11}\lambda_o l}{r_o}y + K_\psi\psi = Q_\psi \tag{2}$$

where,

$$K_y = k_y + \frac{2N_o\epsilon_o^*}{l}\Big(1 - \frac{f_{23}}{N_o r_o}\Big), \quad K_\psi = k_\psi + 2N_o l\Big(-\delta_o + \frac{f_{23}}{N_o l}\Big)$$

We are trying to validate our answers for:-

- Routh Hurwitz criterion
- Root Locus plot and
- Nyquist Criterion

Given Parameters

Parameter	Value	Unit
m	1250	kg
I_z	700	kgm2
I_y	250	kgm2
W	78.48	kN
k_y	0.23	MN/m
k_ψ	2.5	MNm/rad
c_y	0	-
c_ψ	0	-
r_o	0.45	m
l	0.7452	m
λ_o	0.1174	-
ϵ_o	6.423	-
δ_o	0.0493	-
σ	0.0508	-
f_{11}	7.44	MN
f_{22}	6.79	MN
f_{23}	13.7	kN

Routh Hurwitz Criterion

Solving equations (1) and (2) to satisfy Routh Hurwitz criterion as follows:-

$$ms^2Y + \frac{2f_{22}}{V}sY + \left(\frac{2f23}{V} - \frac{I_y\kappa V}{r_o l}\right)s\Psi - 2f_{22}\Psi + K_yY = 0 \tag{3}$$

$$I_z s^2\Psi + \frac{2f_{11}l^2}{V}s\Psi - \left(\frac{2f23}{V} - \frac{I_y\delta_o V}{r_o l}\right)sY + \frac{2f_{11}\lambda_o l}{r_o}Y + K_\psi\Psi = 0 \tag{4}$$

Substituting Y in Ψ leads to an equation of degree 4 in s, which is as follows:-

$$p_4s^4 + p_3s^3 + p_2s^2 + p_1s + p_0 = 0 \tag{5}$$

Where,

$$p_4 = mI_z$$
$$p_3 = 2(mf_{11}l^2 + lf_{22})/V$$
$$p_2 = mK_\psi + I_zK_y + 4f_{11}f_{22}l^2/V^2 + (2f_{23}/V - I_y\sigma V/r_ol)(2f_{23}/V - I_y\kappa V/r_ol)$$
$$p_1 = 2f_{22}K_\psi/V + 2f_{11}l^2K_y/V - 4f_{23}(f_{22} + f_{11}\lambda l/r_o)/V + 2I_yV(\sigma f_{22} + \kappa f_{11}\lambda l/r_o)/r_ol$$
$$p_0 = K_yK_\psi + 4f_{22}f_{11}\lambda l/r_o$$

Now, in Routh matrices, R_2 and R_3 should be zero for critical velocity which provides us with our desired velocity.

Code

1. Q4_routh.m

```
V = 10:0.1:250; %In meter/sec
m=1250;
Iz= 700;
Iy = 250;
W = 78.48*1000;
ky = 0.23*10^6;
kshi = 2.5*10^6;
cy = 0;
cshi = 0;
ro = 0.45;
l = 0.7452;
lambdao = 0.1174;
epshilao = 6.423;
deltao = 0.0493;
sigma = 0.0508;
f11 = 7.44*10^6;
f22 = 6.79*10^6;
f23 = 13.7*10^3;
% Taken from slide (not given in the ques.)
No = 39340; %normal force
k = deltao*(1 - f23/(No*ro));

Ky = ky + (2*No*epshilao/l)*(1 - f23/(No*ro));
Kshi = kshi + (2*No*l)*(-deltao +f23/(No*l));

for i=1:length(V)
a2(i) = m;
a1(i) = 2*f22/V(i);
a0(i) = Ky;
ba1(i) = (2*f23/V(i)) - (Iy*k*V(i))/(ro*l);
ba0(i) = -2*f22;
```

```
b2(i) = Iz;
b1(i) = 2*f11*l*l/V(i);
b0(i) = Kshi;
ab1(i) = -( 2*f23/V(i) -  Iy*deltao*V(i)/(ro*l) );
ab0(i) = 2*f11*lambdao*l/ro;

%polynomial Constants
P4(i) = -a2(i)*b2(i);
P3(i) = -a1(i)*b2(i) -a2(i)*b1(i);
P2(i) = -a0(i)*b2(i) -a1(i)*b1(i) -a2(i)*b0(i) + ab1(i)*ba1(i);
P1(i) = -a0(i)*b1(i) -a1(i)*b0(i) +ab1(i)*ba0(i) +ab0(i)*ba1(i);
P0(i) = -a0(i)*b0(i) + ab0(i)*ba0(i);

%roots
p = [P4(i) P3(i) P2(i) P1(i) P0(i)];

%finding no of poles on right hand side at different velocity
poles(i)=rhstability(p);
end

H=find(poles>0);
Stable_Velocity = V(H(1)-1)*18/5
```

2. Rhstability.m

```
%% Routh-Hurwitz stability criterion
function z= rhstability(r)

coeffVector = r;
ceoffLength = length(coeffVector);
rhTableColumn = round(ceoffLength/2);

rhTable = zeros(ceoffLength,rhTableColumn);
rhTable(1,:) = coeffVector(1,1:2:ceoffLength);

% Check if length of coefficients vector is even or odd
if (rem(ceoffLength,2) ~= 0)
    % if odd, second row of table will be
    rhTable(2,1:rhTableColumn - 1) = coeffVector(1,2:2:ceoffLength);
else
    % if even, second row of table will be
    rhTable(2,:) = coeffVector(1,2:2:ceoffLength);
end

%% Calculate Routh-Hurwitz table's rows

epss = 0.01;
for i = 3:ceoffLength

    % special case: row of all zeros
    if rhTable(i-1,:) == 0
        order = (ceoffLength - i);
        cnt1 = 0;
        cnt2 = 1;
        for j = 1:rhTableColumn - 1
            rhTable(i-1,j) = (order - cnt1) * rhTable(i-2,cnt2);
            cnt2 = cnt2 + 1;
            cnt1 = cnt1 + 2;
        end
    end

    for j = 1:rhTableColumn - 1

        firstElemUpperRow = rhTable(i-1,1);
```

```
        rhTable(i,j) = ((rhTable(i-1,1) * rhTable(i-2,j+1)) - ....
        (rhTable(i-2,1) * rhTable(i-1,j+1))) / firstElemUpperRow;
    end

    if rhTable(i,1) == 0
        rhTable(i,1) = epss;
    end
end

% Compute number of right hand side poles(unstable poles)
unstablePoles = 0;
%   Check change in signs
for i = 1:ceoffLength - 1
    if sign(rhTable(i,1)) * sign(rhTable(i+1,1)) == -1
        unstablePoles = unstablePoles + 1;
    end
end
z=unstablePoles;
end
```

3. Q4_root_nyquist.m

```
%% rootlocus and nyquist plot for stable velocity
V=78.5;
m = 1250;
I_z = 700;
I_y = 500;
W = 78480;
k_y = 0.23*10^6;
k_psi = 2.5*10^6;
c_y = 0;
c_psi = 0;
r_o = 0.45;
l = 0.7542;
lambda_o = 0.1174;
epsilon_o = 6.423;
delta_o = 0.0493;
sigma = 0.0508;
f11 = 7.44*10^6;
f22 = 6.79*10^6;
f23 = 13.7*10^3;
% Data values
N_o = 39240;
K_y = k_y;
K_psi = k_psi;
kappa = delta_o*(1-f23/(N_o*r_o));

%getting the transfer function
a1 = m;
a2 = 2*f22/V;
a3 = -(2*f23/V) + (I_y*kappa*V/(r_o*l));
a4 = 2*f22;
a5 = K_y;
b1 = I_z;
b2 = 2*f11*l^2/V;
b3 = -(2*f23/V) + (I_y*delta_o*V/(r_o*l));
b4 = 2*f11*lambda_o*l/r_o;
b5 = K_psi;

A1 = I_z;
A2 = 2*f11*(l^2)/V;
A3 = K_psi;
B1 = a1*b1;
B2 = a1*b2 + a2*b1;
B3 = a1*b5 + b1*a5 + a2*b2 + a3*b3;
B4 = a2*b5 + a5*b2 + a3*b4 + b3*a4;
B5 = a5*b5 + a4*b4;
```

```
%
G1 = tf([A1 A2 A3],[B1 B2 B3 B4 B5]);

figure(1)
rlocus(G1);
figure(2)
nyquist(G1);
```

Root locus method

Closed loop transfer function:-

$$\frac{Y}{F_o} = \frac{[G(i\omega)]}{1+[G(i\omega)][H(i\omega)]} \tag{6}$$

Where, open transfer function is:-

$$[G(i\omega)][H(i\omega)] = \frac{(2f_{22}\psi - (2f_{23}V - I_y\kappa V/r_o l))(i\omega)(2f_{11}\lambda_o I/r_o - (2f_{23}/V - I_y\delta_o V/r_o l)(i\omega))}{(-m\omega^2 + 2(i\omega)f_{22}/V + K_y)(-Iz\omega^2 + 2f_{11}l^2(i\omega)/V + k_\psi)} \tag{7}$$

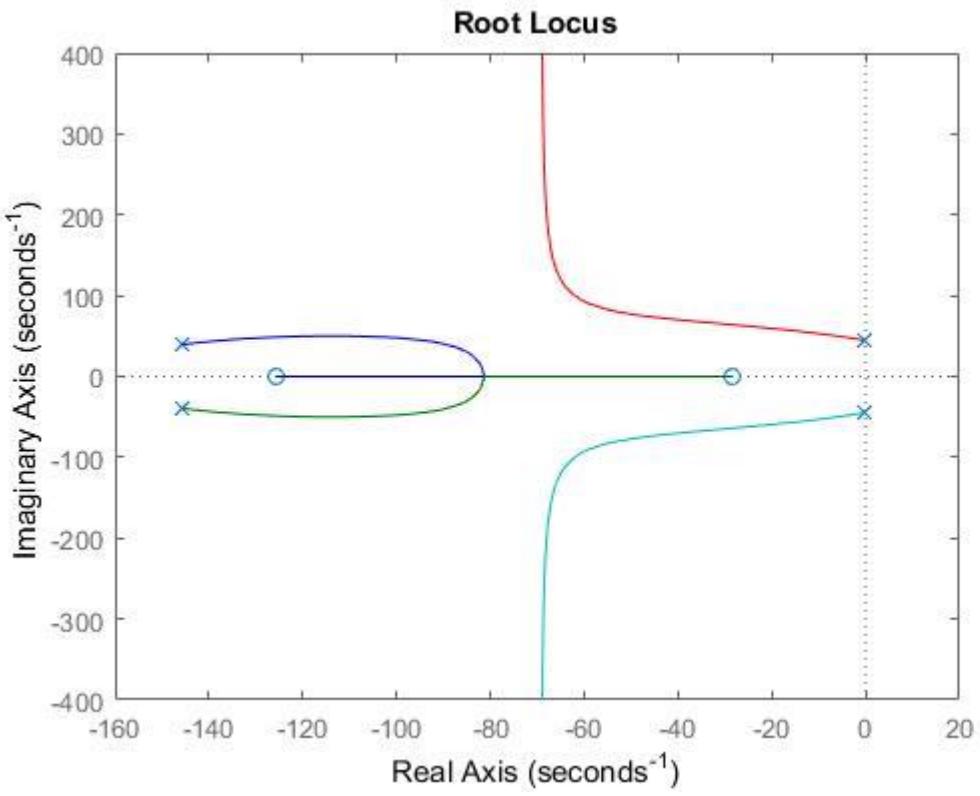


Figure 1: Root Locus plot for given parameters at critical velocity

Nyquist criterion

For nyquist criterion the transfer function calculated is same as shown above, in the case of root locus for the matlab function.

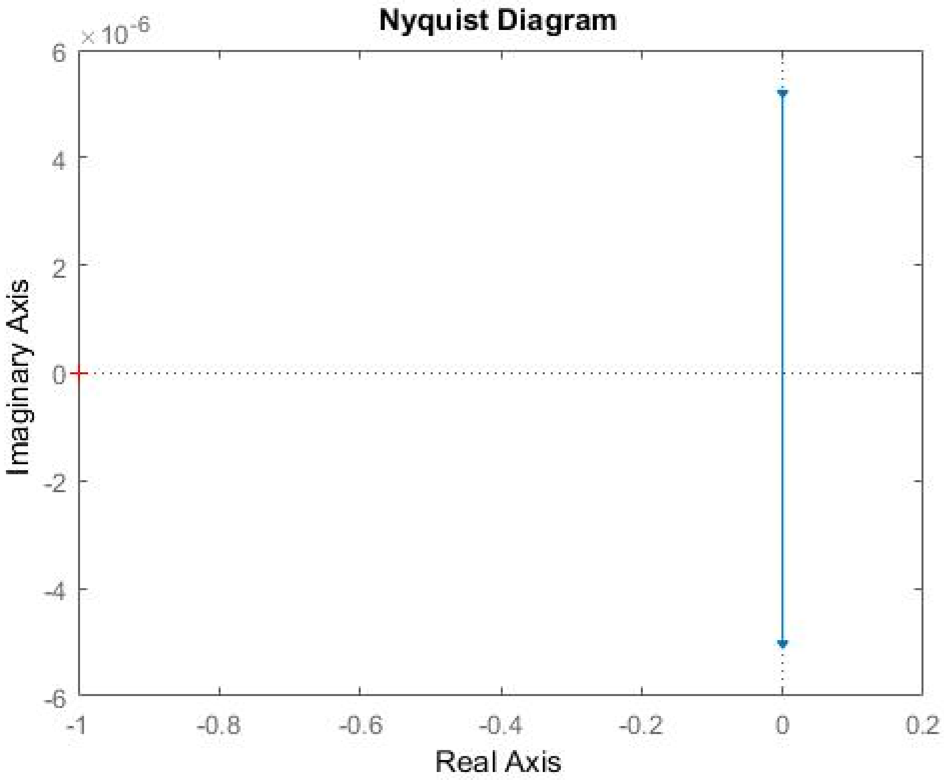


Figure 2: Nyquist plot for given parameters at critical velocity

Conclusion

In all 3 stability criterion, the critical velocity found was same and is equal to 282.60 km/hr.

Rail Road Vehicle Dynamics

Assignment 5

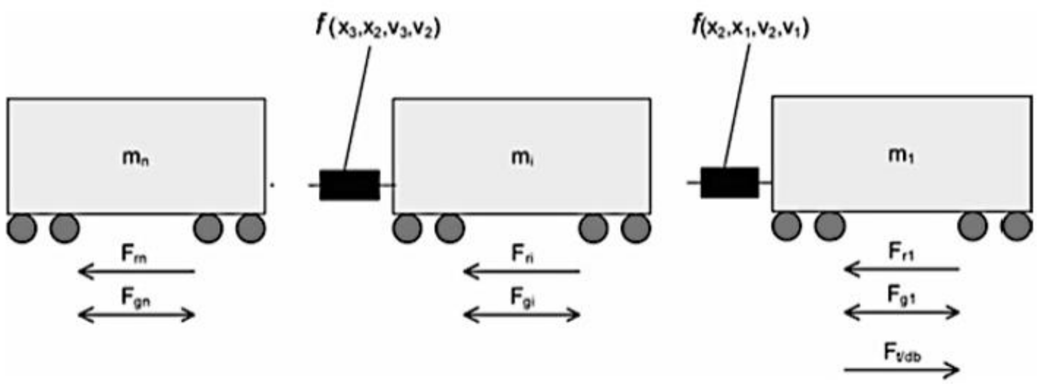
Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Problem Statement:

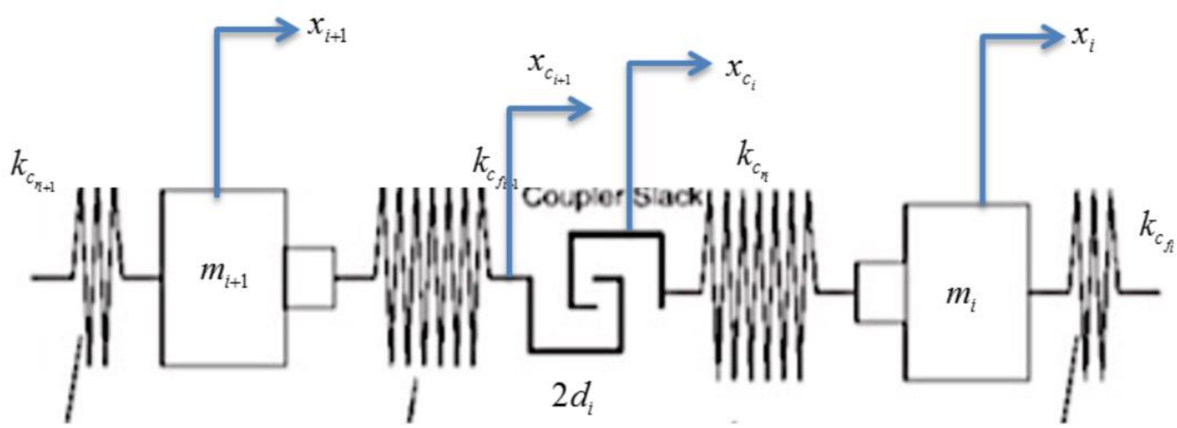
Obtain the Longitudinal natural frequencies and mode shapes for a vehicle-coupler model shown below.

Given Parameters:

Wheelset Paramters (A.H. Wickens Table 2.3)		
Parameter	Value	Unit
$m_1 = m_2 = m_3$	60	tonne
$k_{1r} = k_{2f} = k_{2r} = k_{3f}$	10	MN/m
d	0.05	m



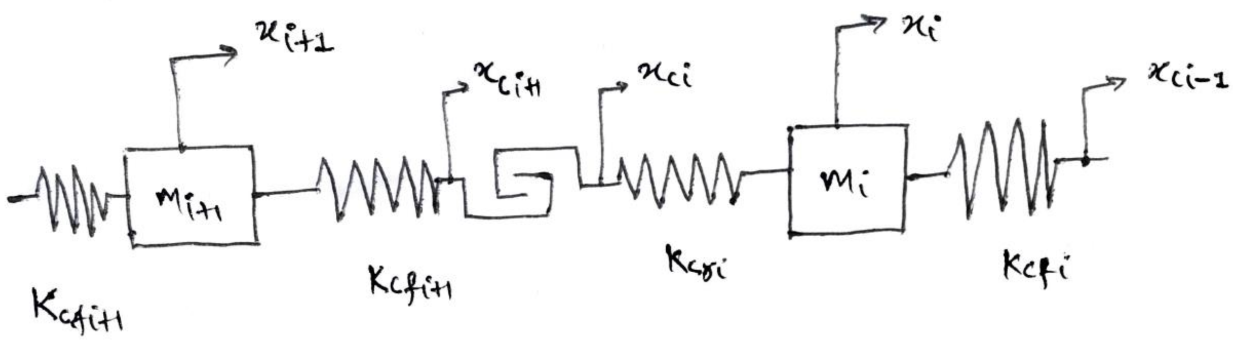
Train of bogies

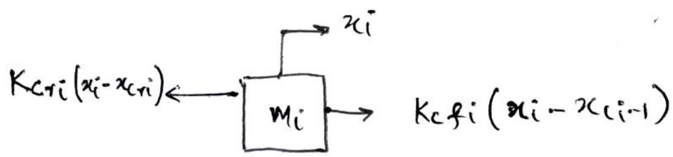


Idealized coupler and vehicle model

Equation of motions:

For general case:



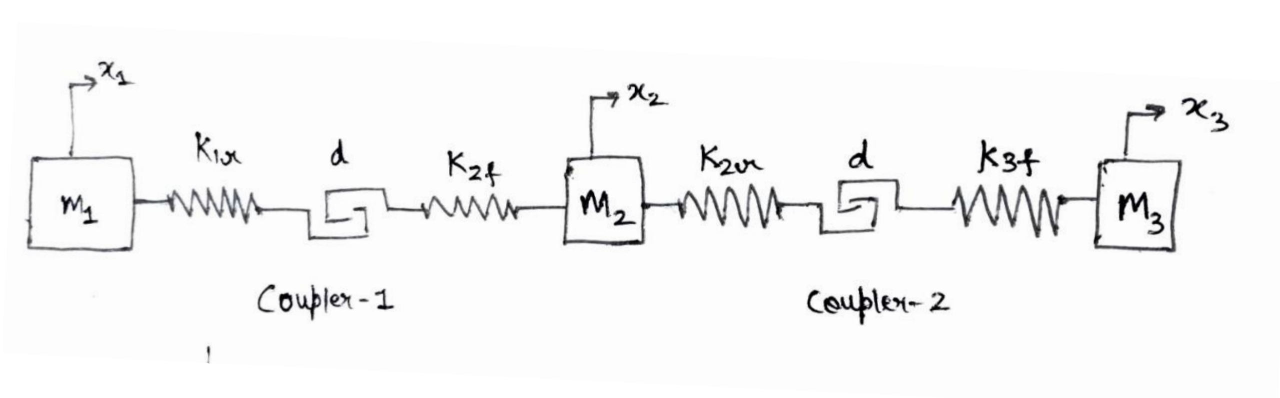


$$m_i\ddot{x}_i + K_{cri}(x_i - x_{cri}) - K_{cfi}(x_i - x_{ci-1}) = 0$$

$$\boxed{m_i\ddot{x}_i + (-K_{cfi} + K_{cri})x_i + K_{cfi}x_{ci-1} - K_{cri}x_{ci} = 0}$$

& $|xc_i - xc_{i-1}| <= 2d_i$

For this problem, there are 3 masses so we will have three different cases:



- m1=m2=m3=m & $k_{1r}=k_{2f}=k_{2r}=k_{3f}=k$

Case1. Both coupler are disengaged

Equation of motion:

$$m\ddot{x}_1 = 0 \qquad (1)$$
$$m\ddot{x}_2 = 0 \qquad (2)$$
$$m\ddot{x}_3 = 0 \qquad (3)$$

Natural frequency and mode shapes:

S. No	Natural Frequency (rad/sec)	Mode Shape
1.	Ω1= 0	{1 0 0}
2.	Ω2= 0	{0 1 0}
3.	Ω3= 0	{0 0 1}

Case2. One coupler is engaged and other is disengaged

Equation of motion:

$$m\ddot{x}_1 - \frac{k}{2}(x_2 - x_1 - d) = 0 \qquad (1)$$
$$m\ddot{x}_2 + \frac{k}{2}(x_2 - x_1 - d) = 0 \qquad (2)$$
$$m\ddot{x}_3 = 0 \qquad (3)$$

Natural frequency and mode shapes:

S. No	Natural Frequency (rad/sec)	Mode Shape
1.	Ω1= 0	{0 0 1}
2.	Ω2= 0	{1 1 0}
3.	Ω3= 12.909	{1 -1 0}

Case3. Both coupler are engaged

Equation of motion:

$$m\ddot{x}_1 - \frac{k}{2}(x_2 - x_1 - d) = 0 \qquad (1)$$
$$m\ddot{x}_2 + \frac{k}{2}(x_2 - x_1 - d) + \frac{k}{2}(x_2 - x_3 - d) = 0 \qquad (2)$$
$$m\ddot{x}_3 - \frac{k}{2}(x_2 - x_3 - d) = 0 \qquad (3)$$

Natural frequency and mode shapes:

S. No	Natural Frequency (rad/sec)	Mode Shape
1.	Ω1= 0	{1 1 1}
2.	Ω2= 9.1287	{1 0 -1}
3.	Ω3= 15.8114	{-0.5 1 -0.5}

Matlab script to solve eigenvalue problem: `[v,d]=eig(k,m);`

Rail Road Vehicle Dynamics

Assignment 6 Report

Question : Obtain the coupled vertical and pitch motions natural frequencies and mode shapes for a Two Axle Rail Vehicle model.

Group Members:
Rajesh Mishra (150557)
Tarun Sharma (150764)
Vaibhav Raj Singh (150788)
Pushpendra Singh (14807510)

Algorithm:

- Formulated the mass([M]) and stiffness([K]) matrix from the 6 equations of motion given to us in the assignment.
- Obtained the matrix [D] = $[M]^{-1}[K]$.
- Calculated the eigenvalues(Natural Frequencies) and eigenvectors(Mode Shapes) of [D].

Code for the Assignment:

File Name : Assignment_6.m

```
clear all;
close all;

%Given Paramters
m_w = 1200;
I_w = 1500;
k_cf = 35*10^3;
k_cr = 45*10^3;
m_uf=  160;
m_ur = 160;
k_tf =  375*10^3;
k_tr = 375*10^3;
l_f = 1.8;
l_r = 0.9;
%Assumed Parameters
m_s = 100;
k_s = 5*10^3;

M = diag([m_uf, m_ur, m_w, I_w, m_s]);
%x = diag([x_uf, x_ur, x_w, theta_w, x_s]);
K = [k_cf + k_tf,       0,                  -k_cf,                  k_cf*l_f,                       0;
    0,                  k_cr+k_tr,          -k_cr,                  -k_cr*l_r,                      0;
    -k_cf,              -k_cr,               (k_cf + k_cr +k_s),     (k_cr*l_r - k_cf*l_f),          -k_s;
    k_cf*l_f,           -k_cr*l_f,          (k_cr*l_r - k_cf*l_f),  (k_cf*l_f*l_f + k_cr*l_r*l_r),  0;
    0,                  0,                  -k_s,                   0,                              k_s];

[eigvec,eigval] = eig(inv(M)*K);
Natural_Frequencies = diag(eigval)
Mode_Shape = eigvec
```

How to run the code:

1. Run "**Assignment_6.m**"
2. Click on "**run button**", wait for the code run time (less than 1 second)
3. Open the variables "Natural_Frequencies" and "Mode_Shape" in MATLAB Workspace to get the desired results.

Results:

```
Natural_Frequencies =

   1.0e+03 *

    2.5717
    2.6345
    0.0981
    0.0390
    0.0648


Mode_Shape =

    0.9997   -0.0821    0.1590   -0.0068    0.0084
    0.0159    0.9963   -0.0249   -0.0306   -0.0511
   -0.0120   -0.0135    0.4478   -0.2141   -0.2781
    0.0167   -0.0225   -0.7465   -0.0752   -0.2078
    0.0002    0.0003   -0.4650   -0.9734    0.9364
```

Winter‘18 Report
Rail Road Vehicle Dynamics

Under the supervision of Prof. N.S. Vyas

Tarun Sharma
Rajesh Mishra
Pulkit Jain
Aditya Pratap Singh Rajawat

Contents

About SIMPACK

Simpack is a general purpose Multibody Simulation (MBS) software used for the dynamic analysis of any mechanical or mechatronic system. It enables engineers to generate and solve virtual 3D models in order to predict and visualize motion, coupling forces and stresses.

Simpack is used primarily within the automotive, engine, HiL/SiL, power transmission, railway, and wind energy industrial sectors, but can be applied to any branch of mechanical engineering.

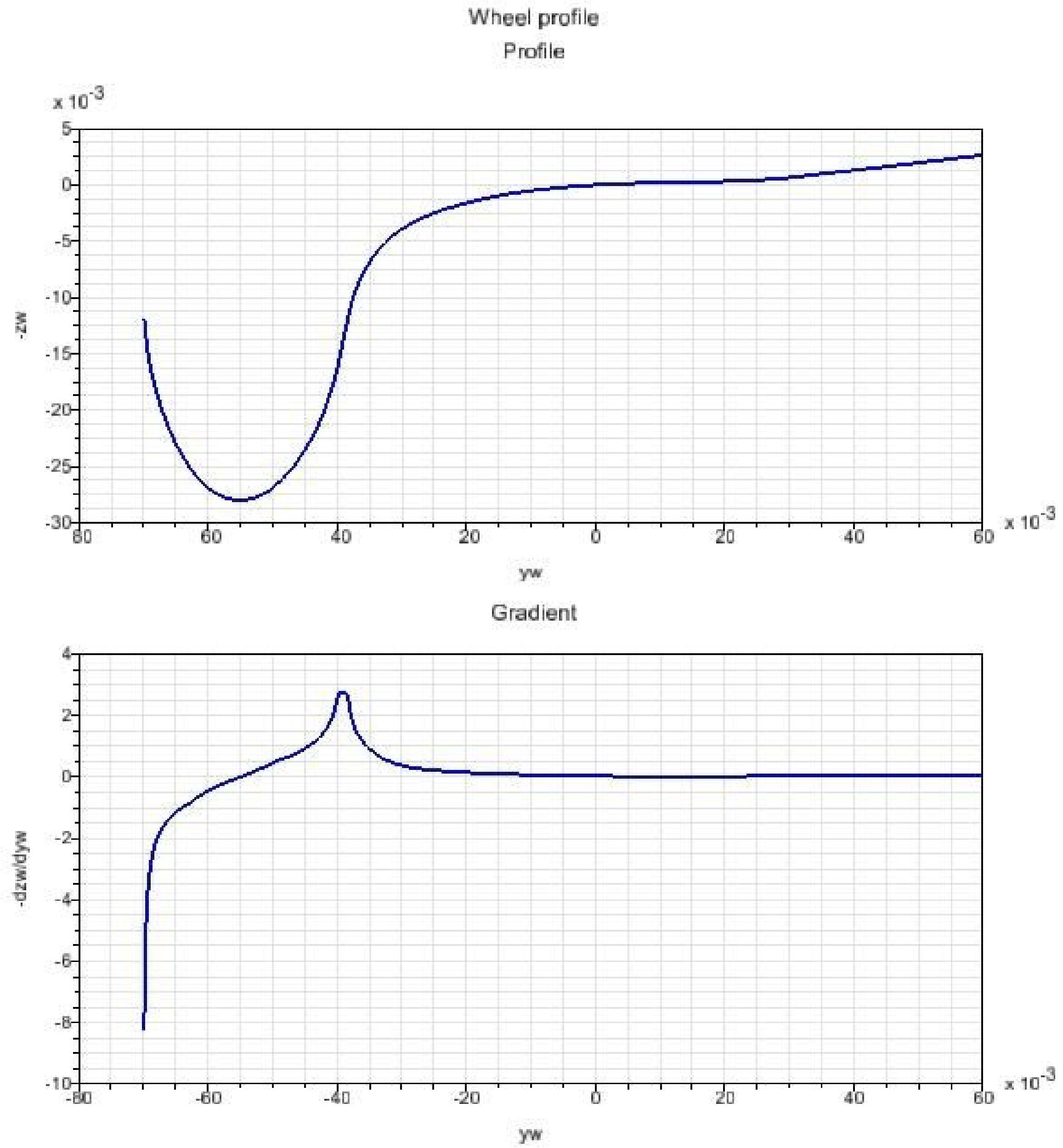
SIMPACK is used for the analysis and design of any type of rail-based vehicle or mechanism–from roller coasters, material handling systems or tramcars to complete articulated high-speed trains. Used worldwide by manufacturers and operators, SIMPACK is the leading MBS software for railway system dynamics.

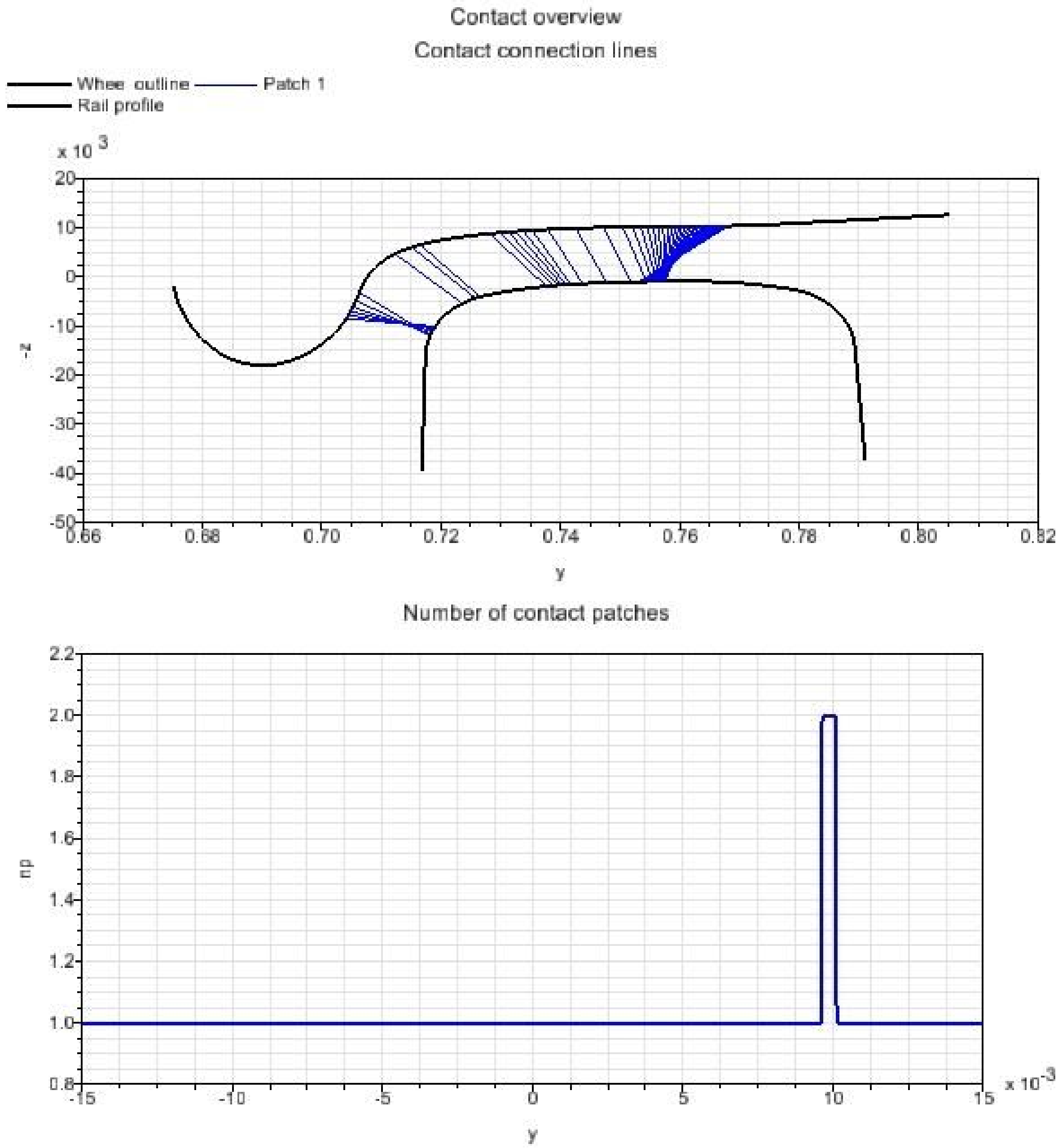
In Railway model, a wheelset could be directly imported with rail and wheel profiles and contact models are also available for the same.

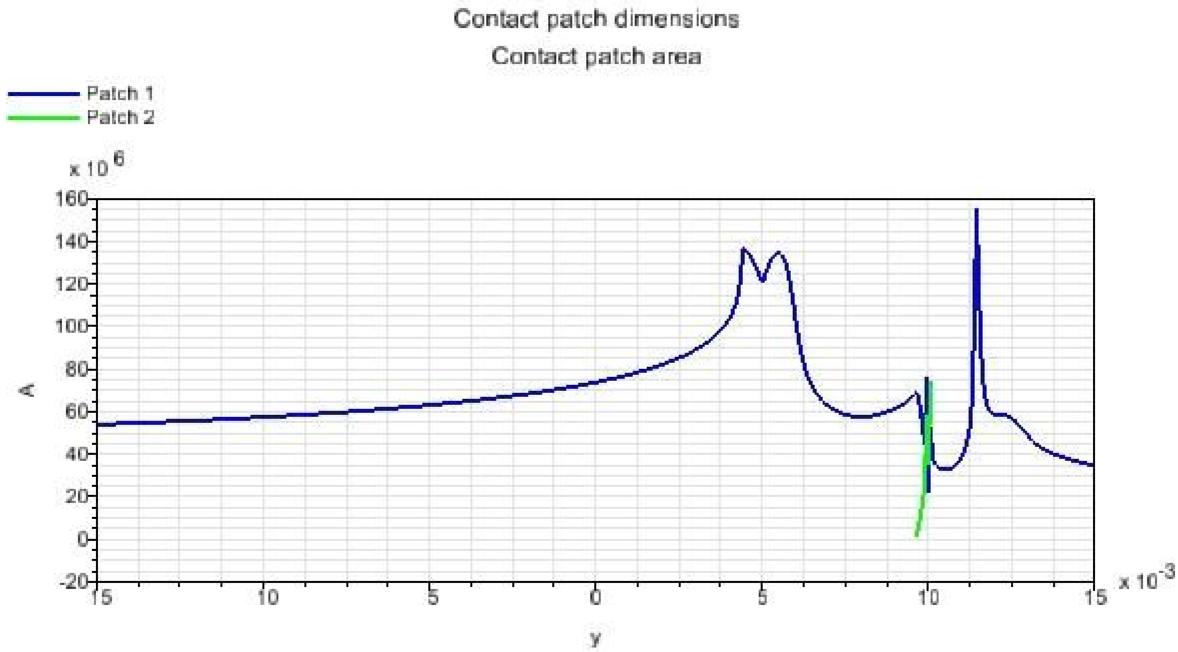
Plots for the wheelset

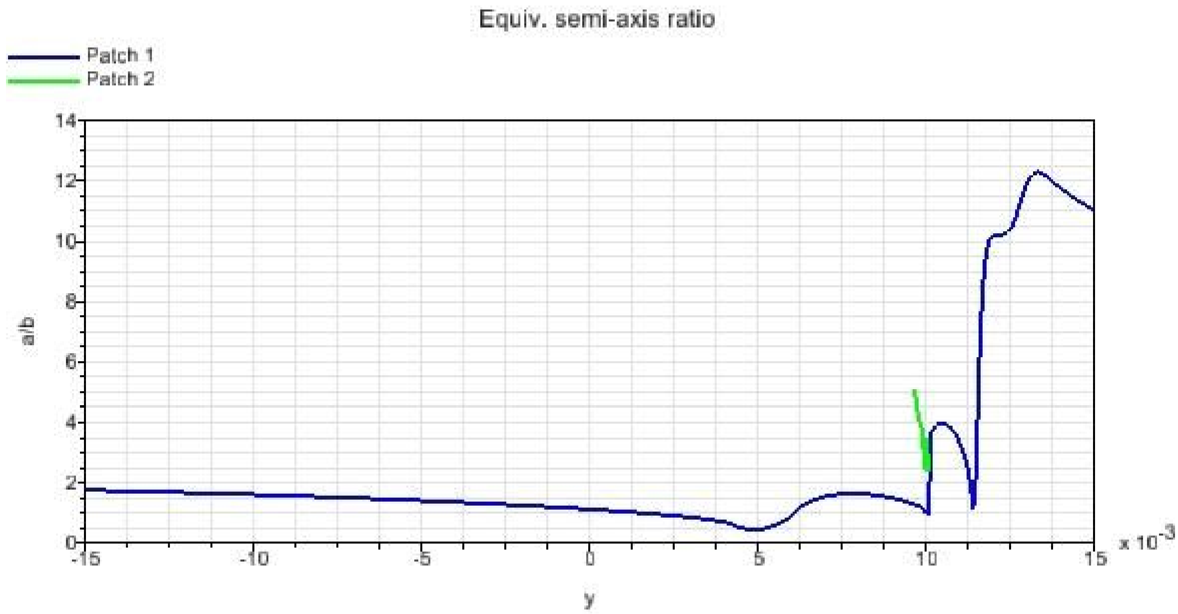
(Folder: RRVD_Report\Winter Project\RAJESH\Wheelset\Plots)

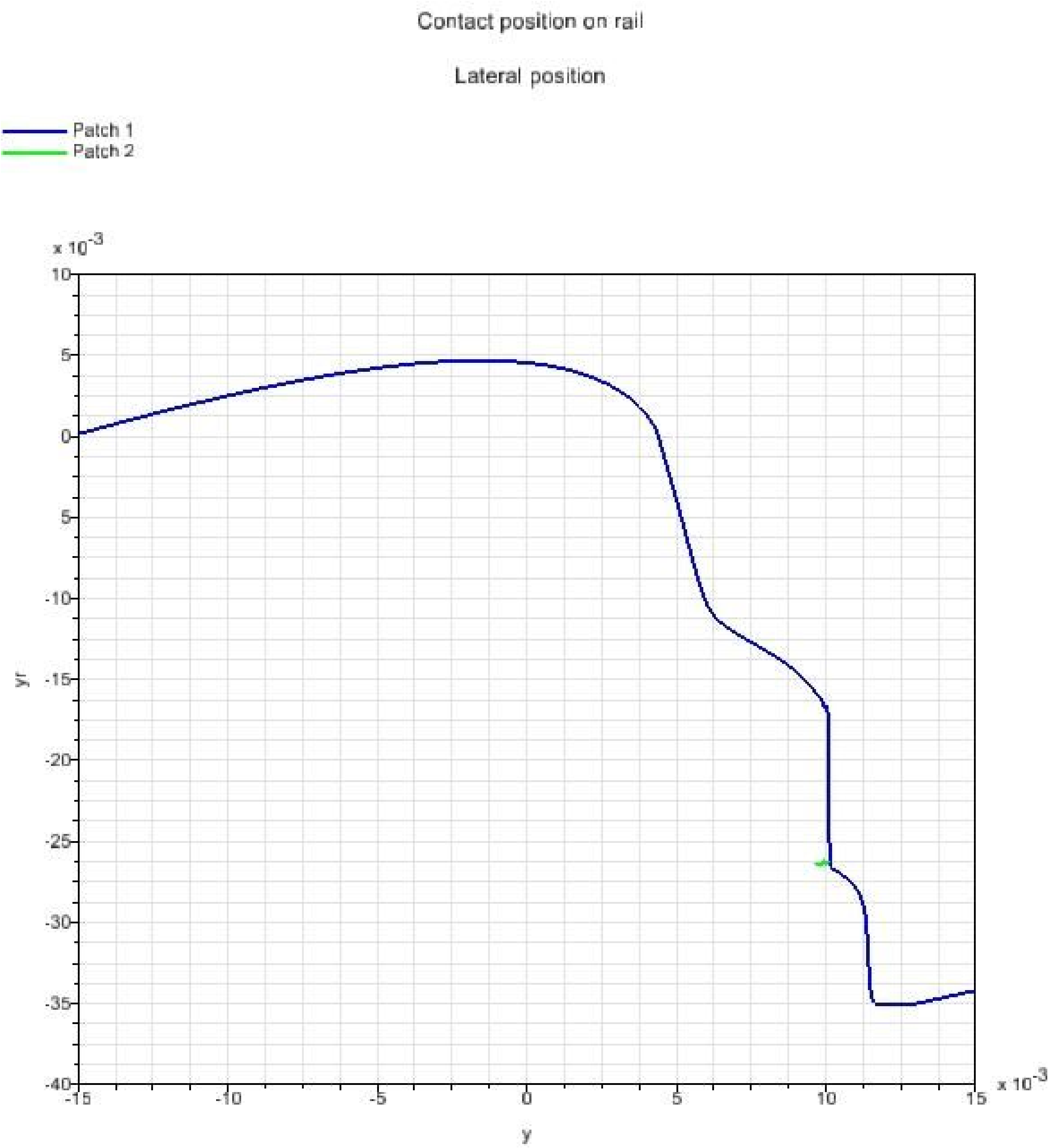
A general wheelset model was available on SIMPACK and the plots which are shown in the next few pages were available with the model.

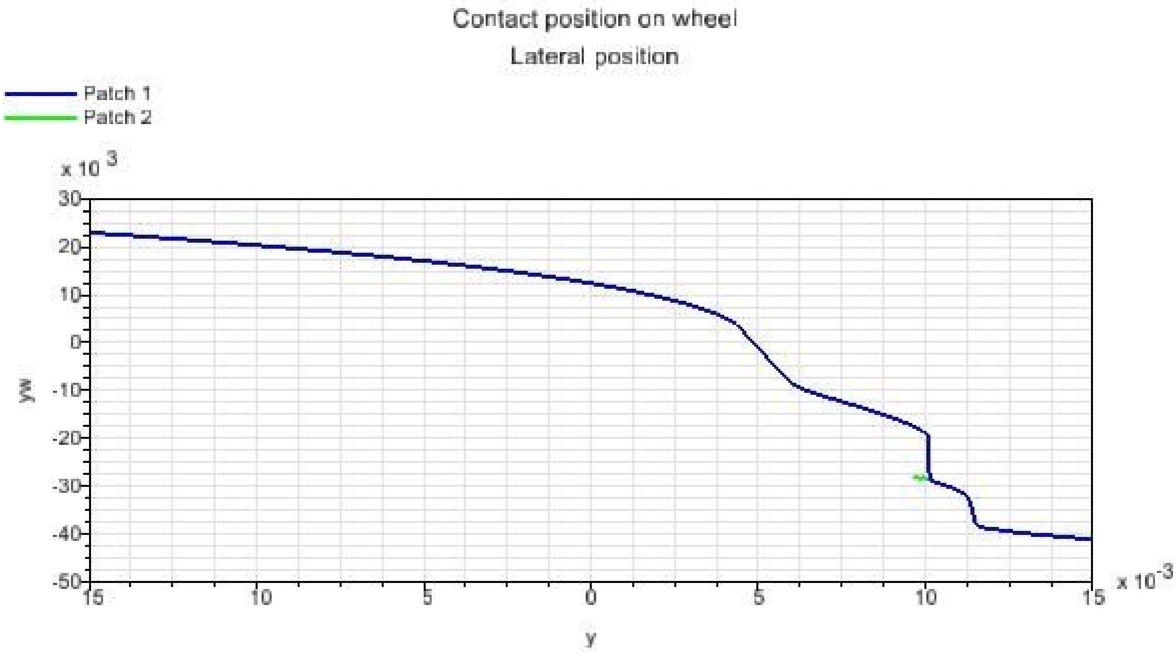


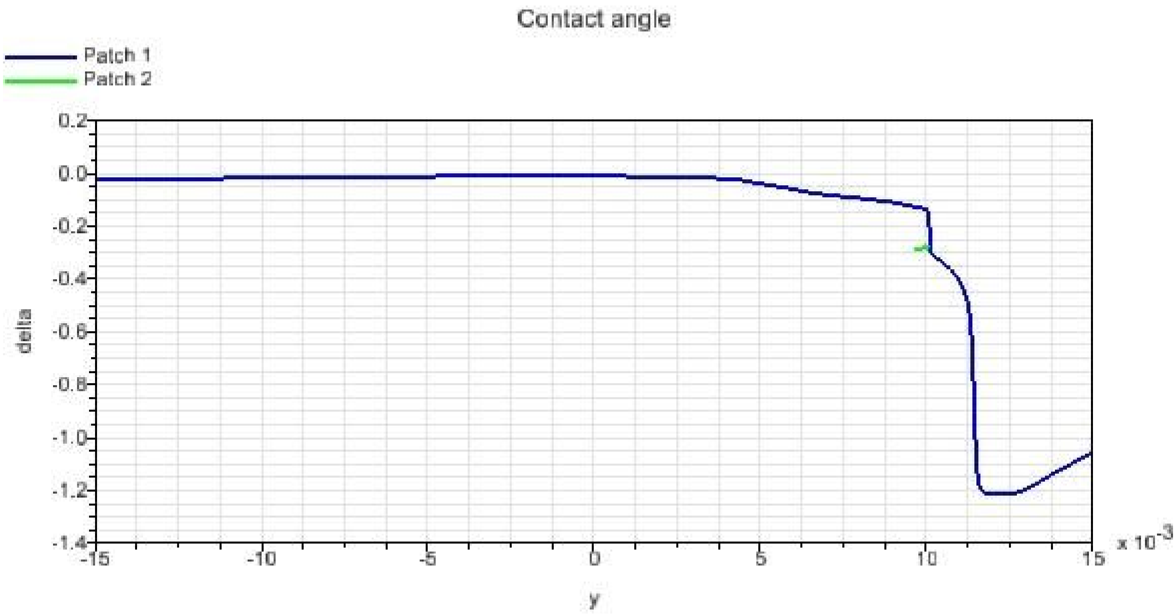

Contact overview
Contact connection lines
Whee outline
Patch 1
Rail profile
x 10 3
-z
y
Number of contact patches
np
y
x 10-3


Contact patch dimensions
Contact patch area
Patch 1
Patch 2
x 10^-6
A
y
x 10^-3


Equiv. semi-axis ratio
Patch 1
Patch 2
a/b
y
x 10^-3

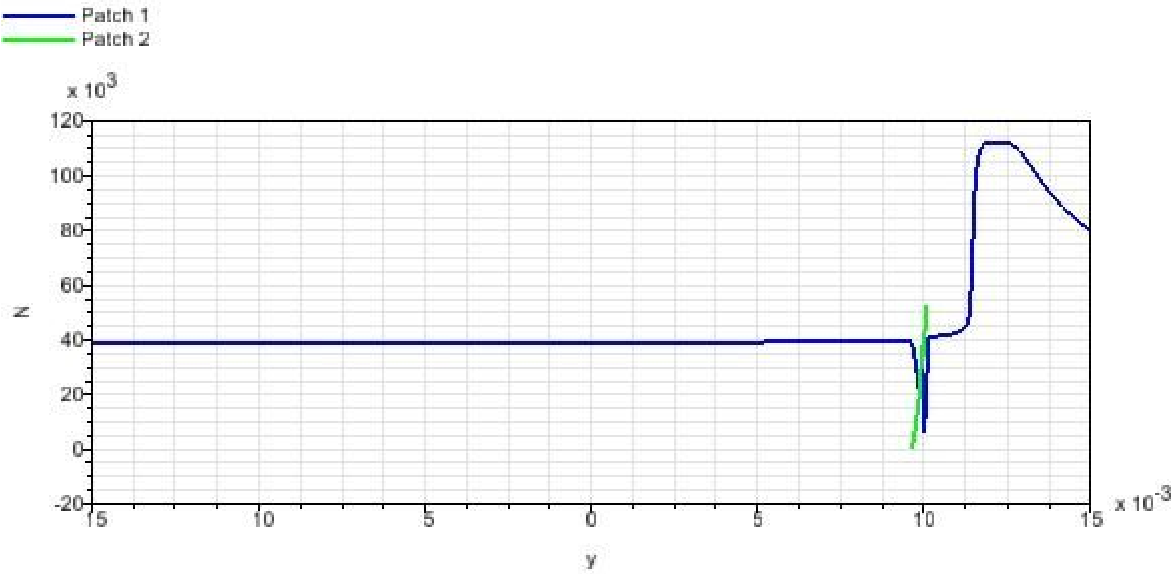

Contact position on rail
Lateral position
Patch 1
Patch 2
x 10^-3
10
5
0
5
-10
-15
-20
-25
-30
-35
-40
yr
-15
-10
-5
0
5
10
15
x 10^-3
y


Contact position on wheel
Lateral position
Patch 1
Patch 2
x 10 3
yw
y
x 10-3

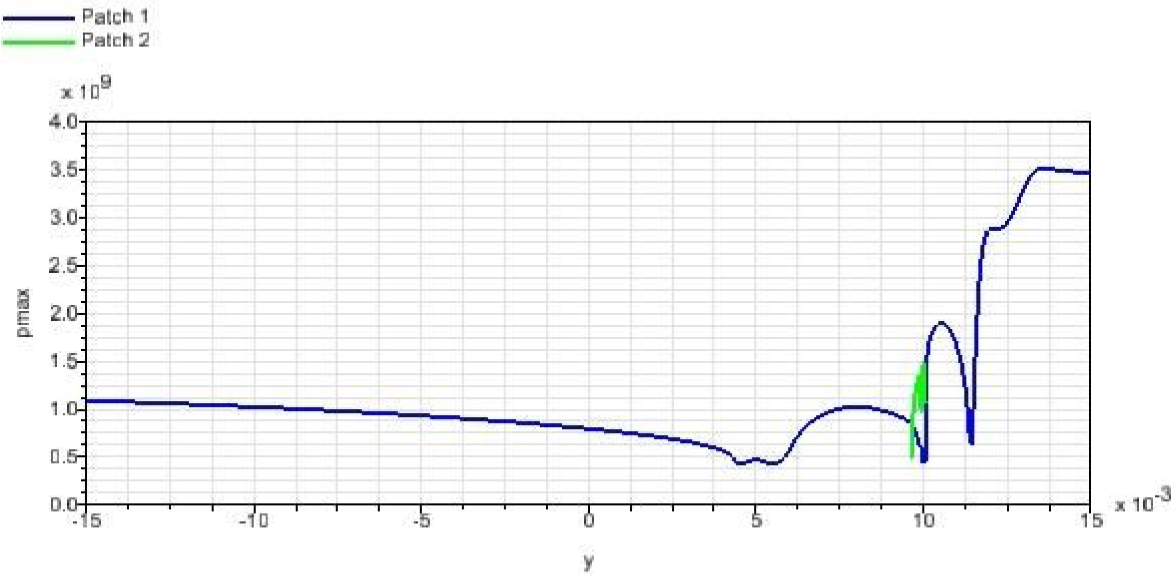

Contact angle
Patch 1
Patch 2
delta
y
x 10-3

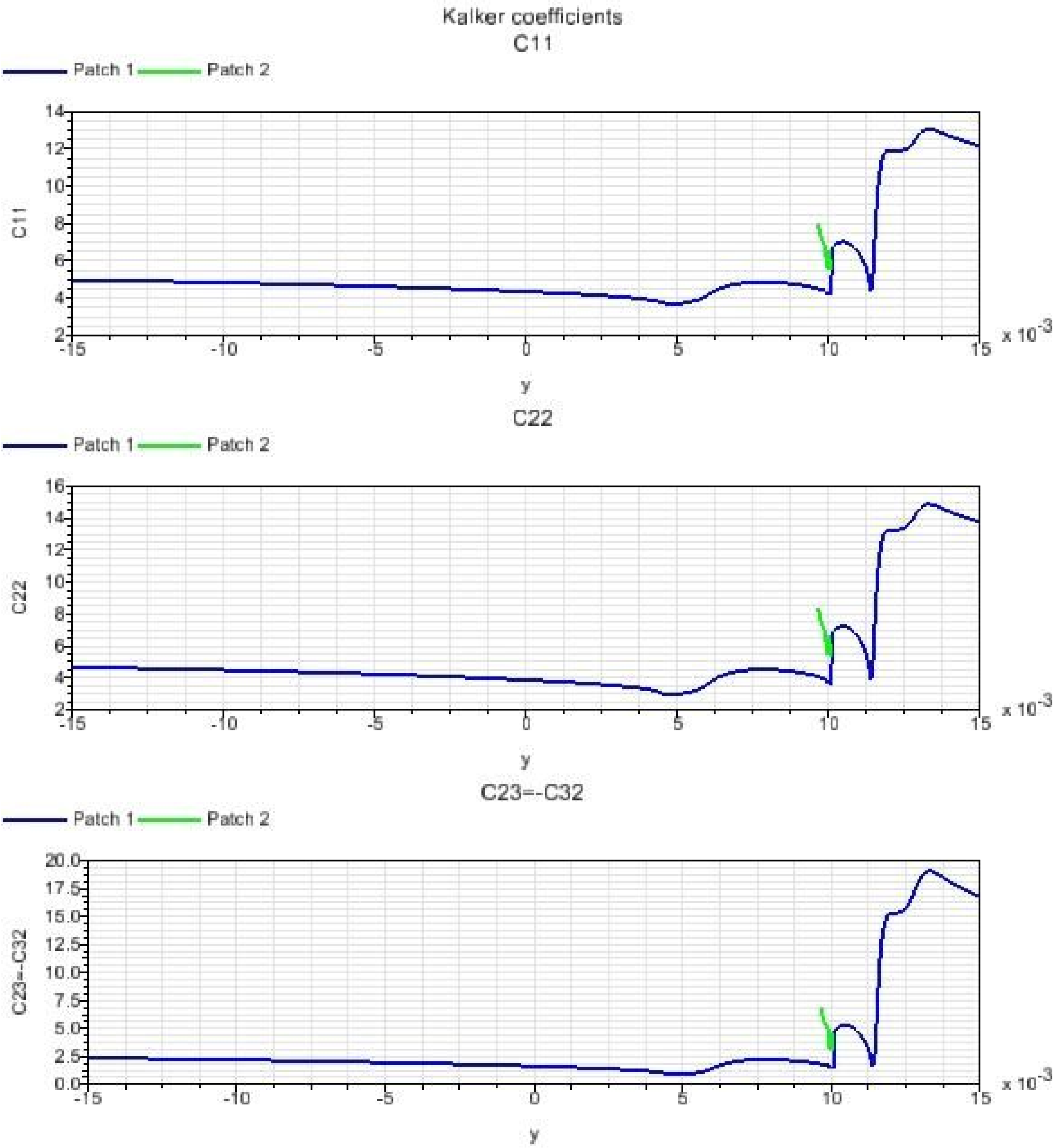
Normal contact

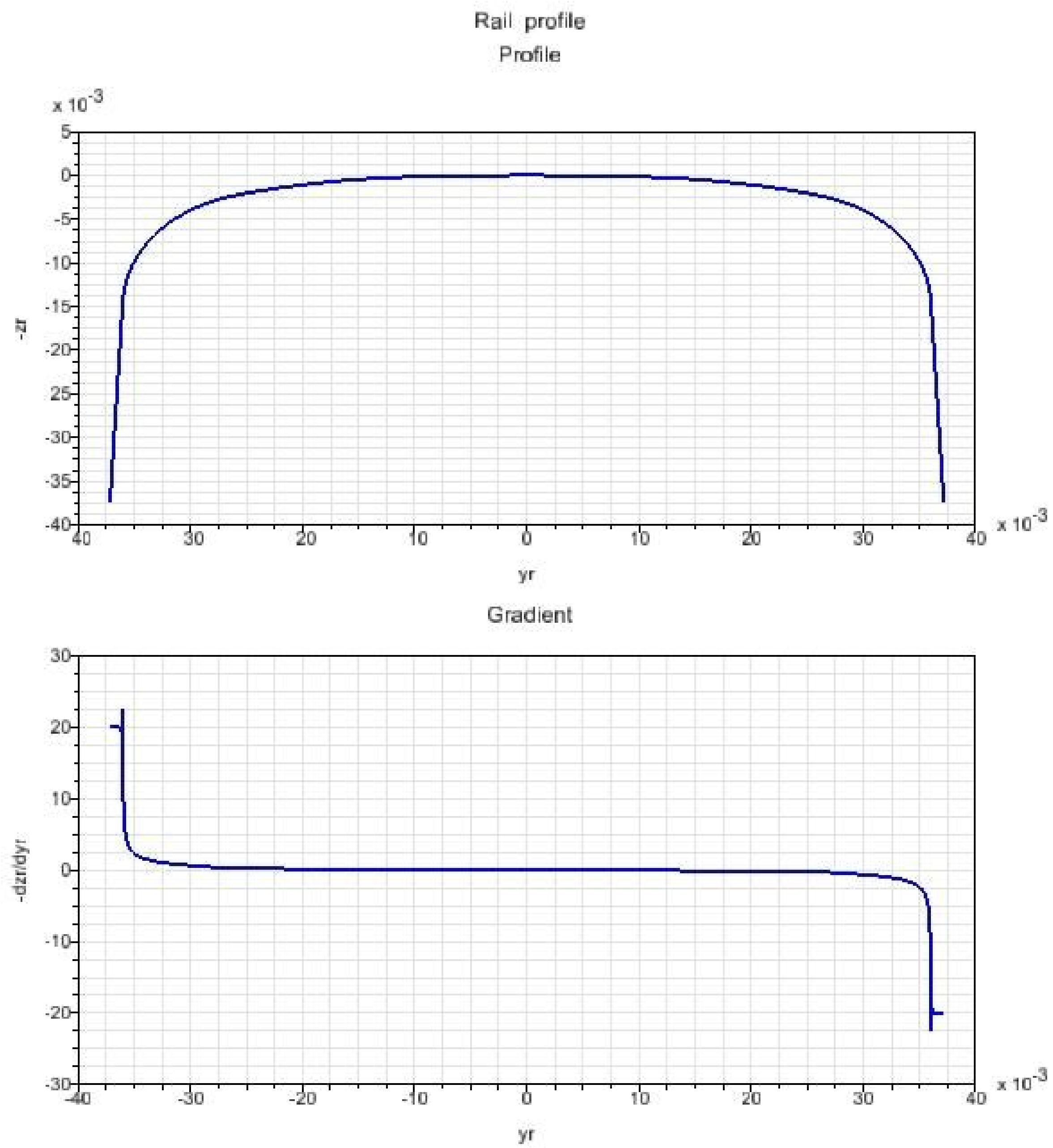
Normal force



Max. Hertzian pressure




Kalker coefficients
C11
Patch 1
Patch 2
C11
y
x 10-3
C22
Patch 1
Patch 2
C22
y
x 10-3
C23=-C32
Patch 1
Patch 2
C23=-C32
y
x 10-3

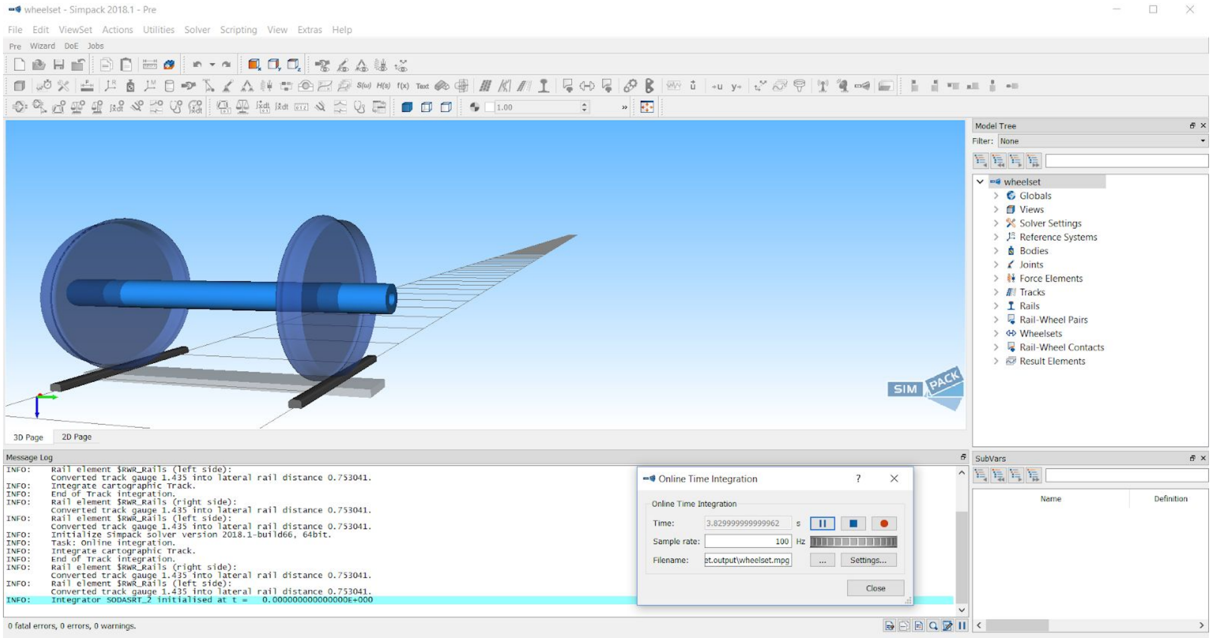


Rail profile
Profile
x 10^-3
-zr
yr
x 10^-3
Gradient
-dzr/dyr
yr
x 10^-3

Single Wheelset Model

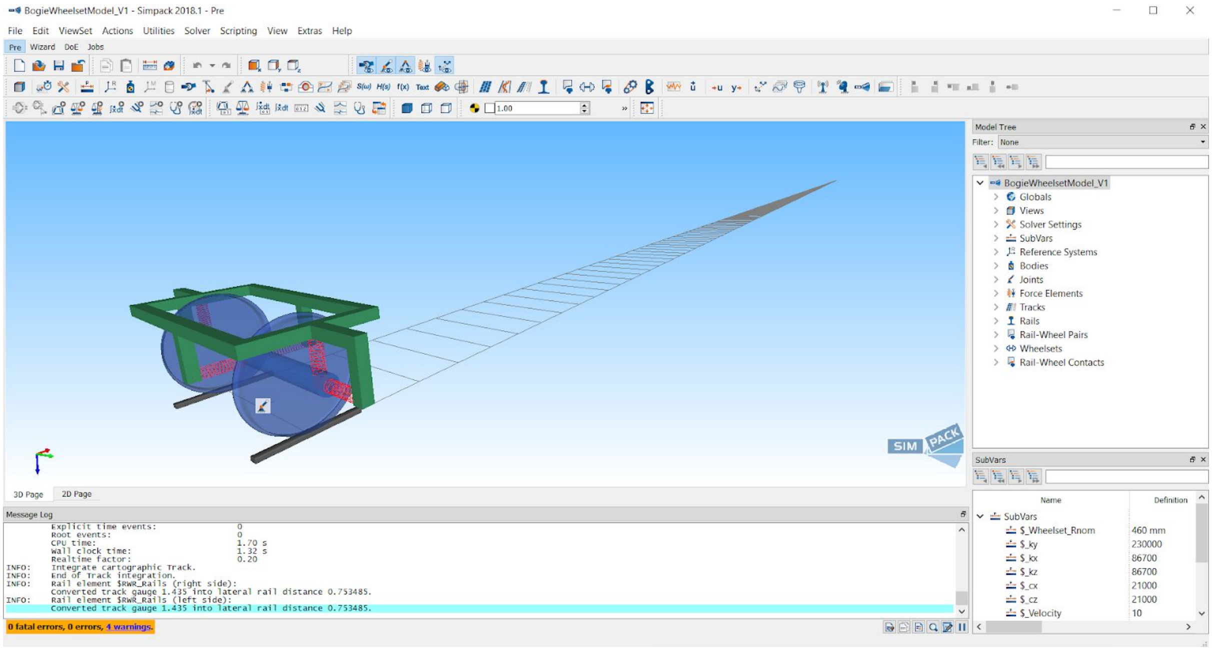
A single wheelset model with track, bolster is created in geometric modeling for basic verification of wheelset kinematics.
It is tested for a constant velocity over a straight track.

File : wheelset



A single wheelset with springs in x, y and z direction were added just to see it's motion behavior. The wheelset is not running properly in this case because of unstable eigenvalues.

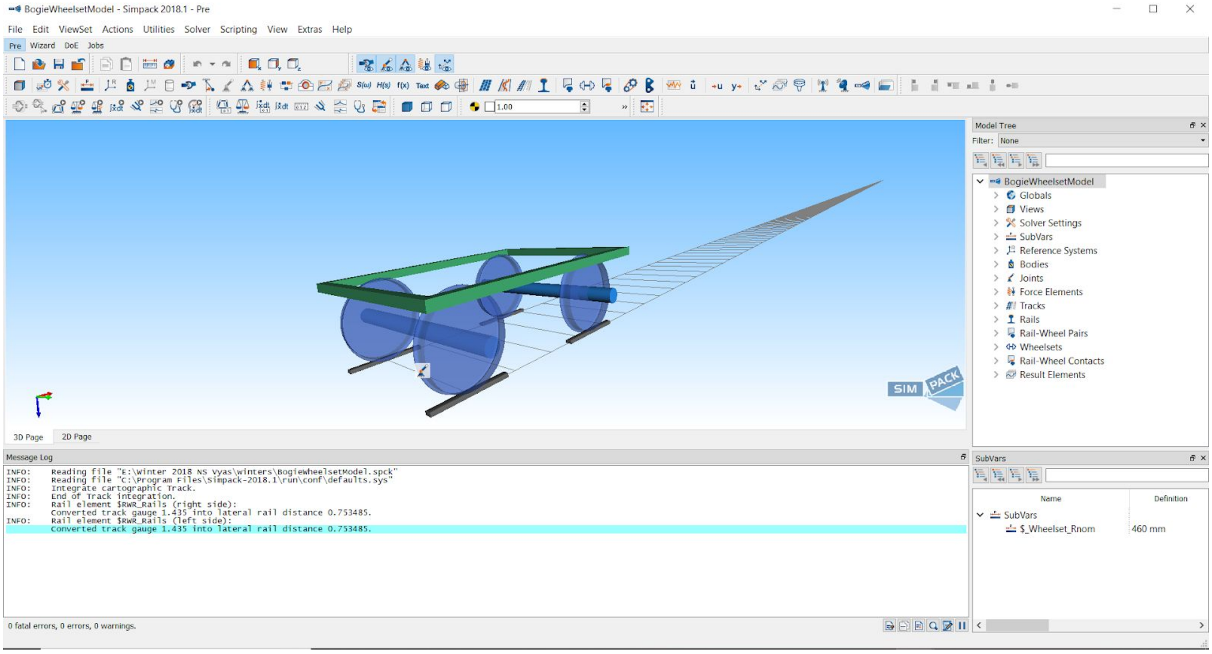
Folder: RRVD_Report\Rajesh\winters\Single Wheelset
File: BogieWheelsetModel_V1



Double Wheelset Model with Frame

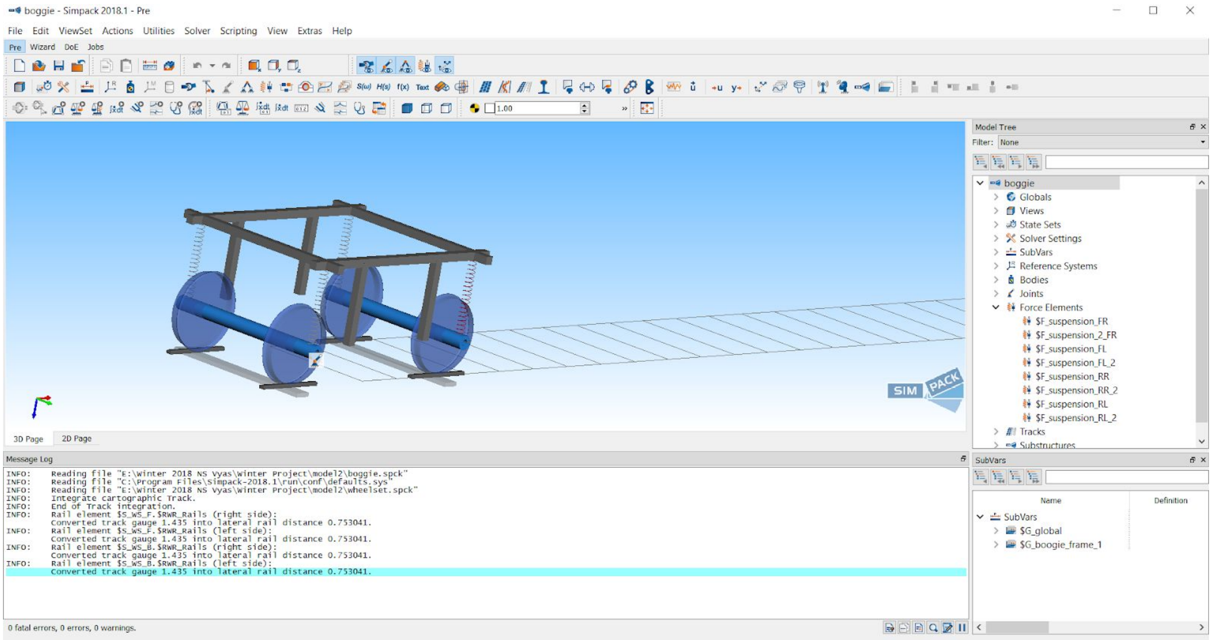
A simple bogie model is created through geometric modeling with 2 wheelsets and point to point forcing elements (springs).

Folder : winters
File : BogieWheelsetModel

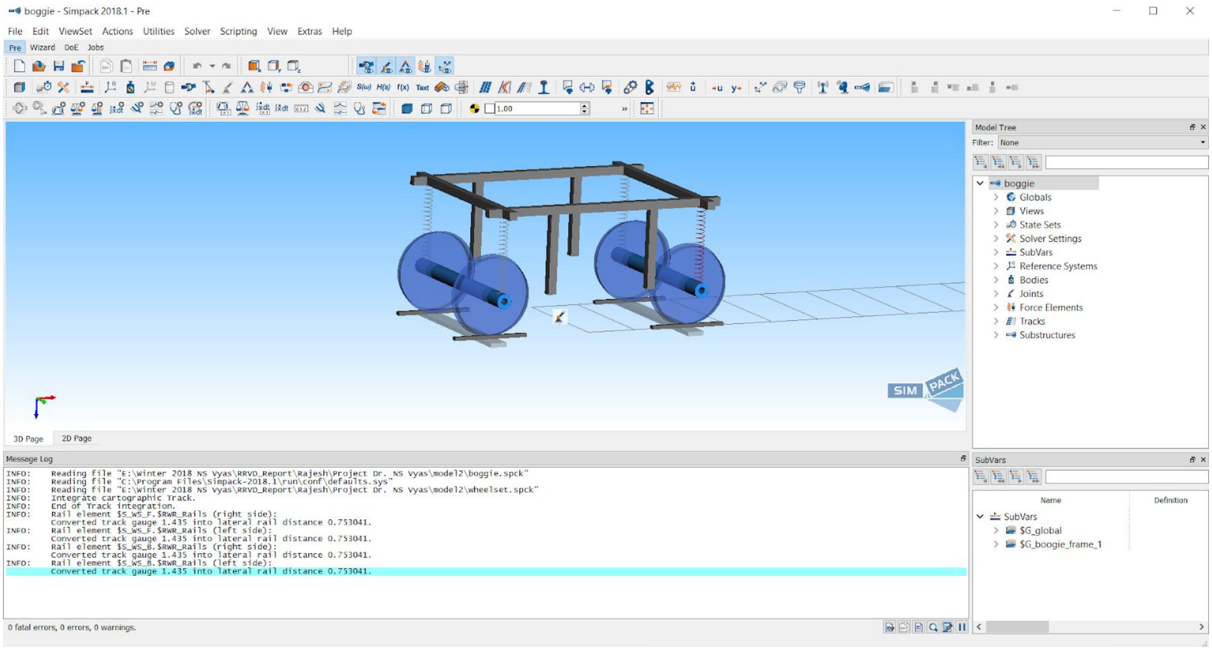


The wheelsets were connected to the frame using forcing elements in the x and z directions. After preloading the structure and making it come to its equilibrium state, eigenvalue analysis was done which showed that the structure was unstable(due to lack of forcing elements in the y direction).

Folder : Winter Project/model2
File : boggie

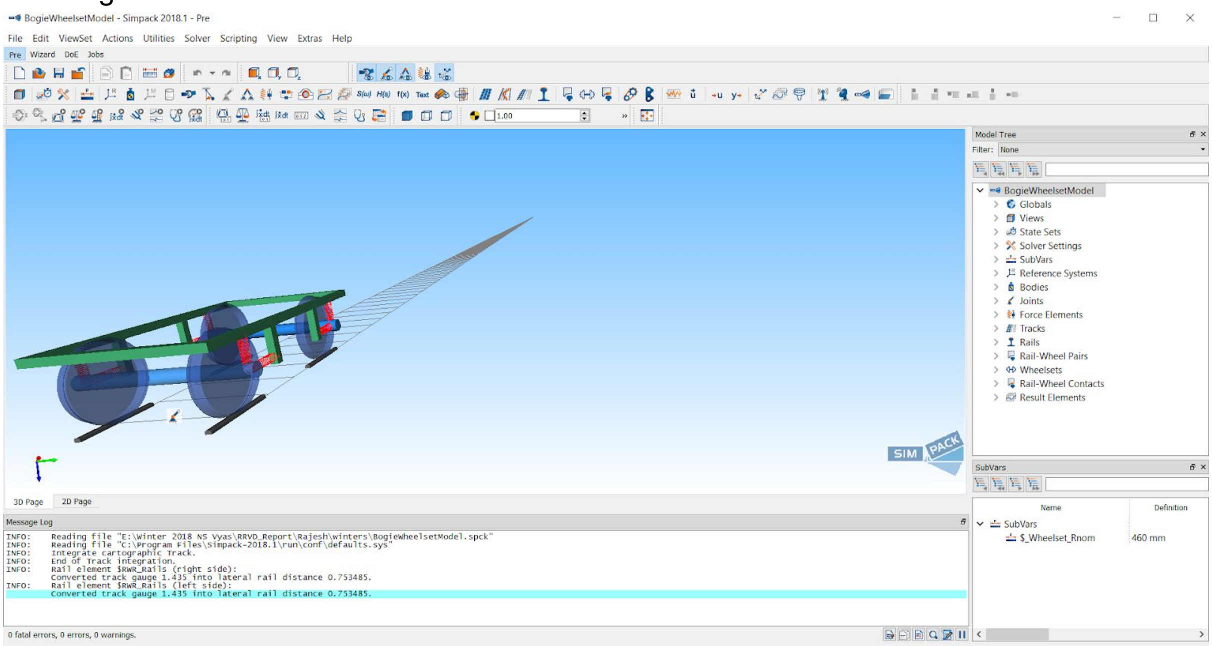


Folder : RRVD_Report\Rajesh\Project Dr. NS Vyas\model2
File: boggie



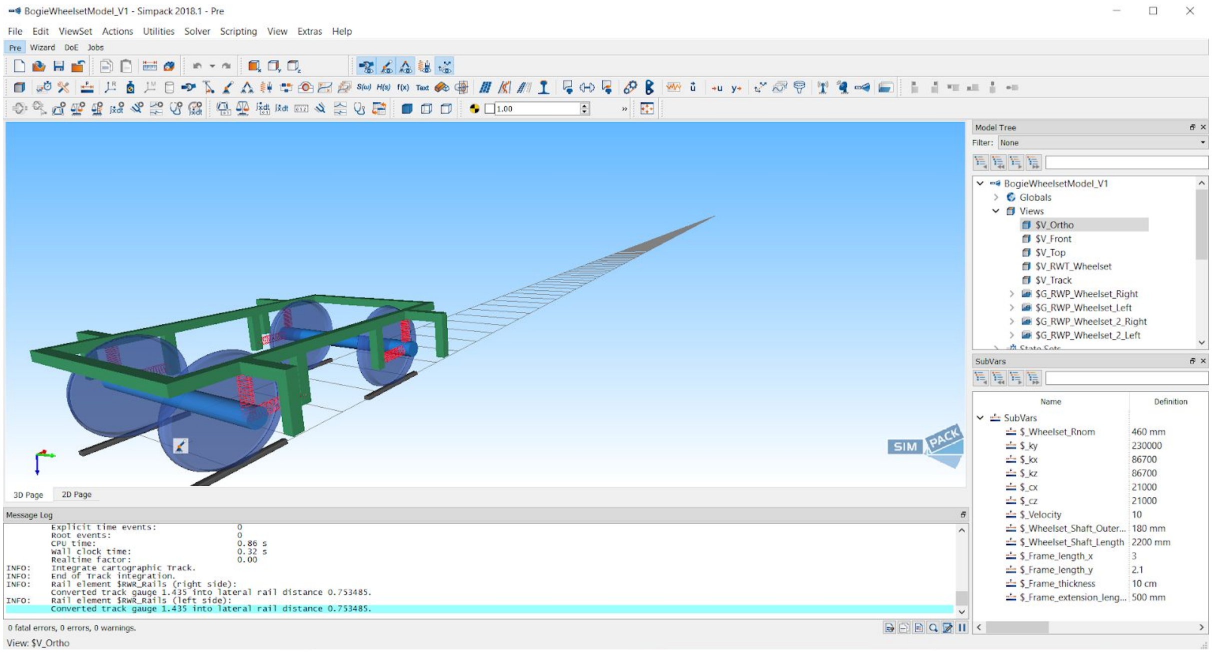
The two wheelsets were now connected to the frame using point to point forcing elements in x, y and z directions and eigenvalue analysis of the model was done to check the stability of various modes. The wheelset was now running.

Folder: RRVD_Report\Rajesh\winters
File: BogieWheelsetModel

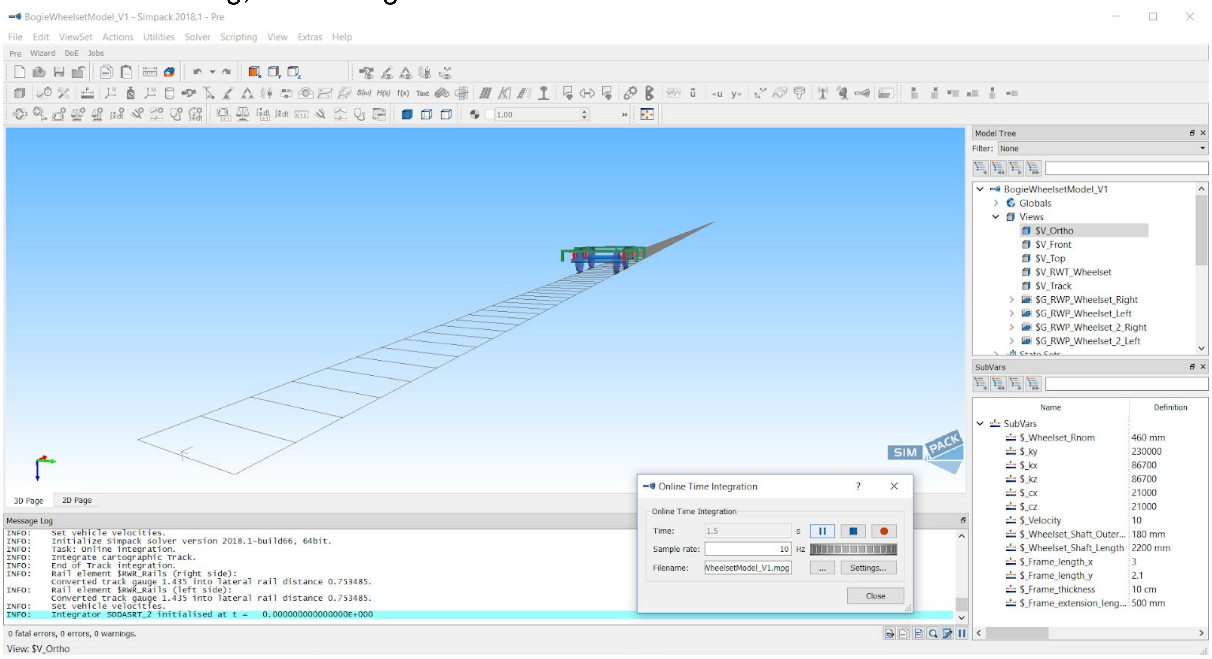


Folder: RRVD_Report\Rajesh\winters

File: BogieWheelsetModel_V1

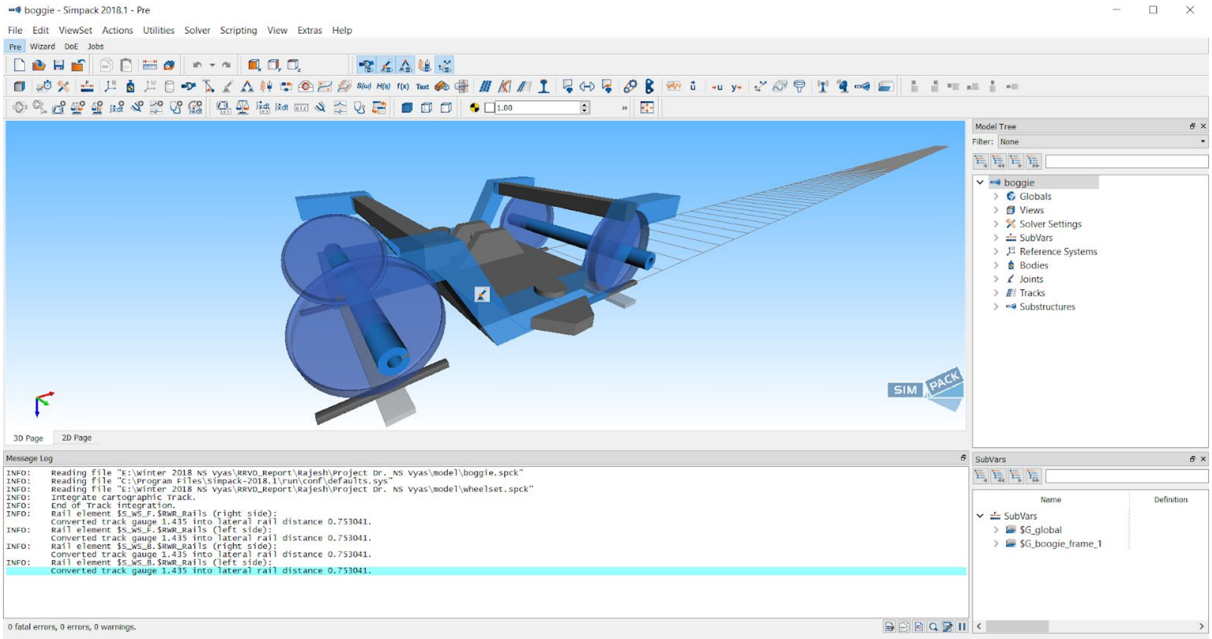


Remarks: Running, See the figure below.

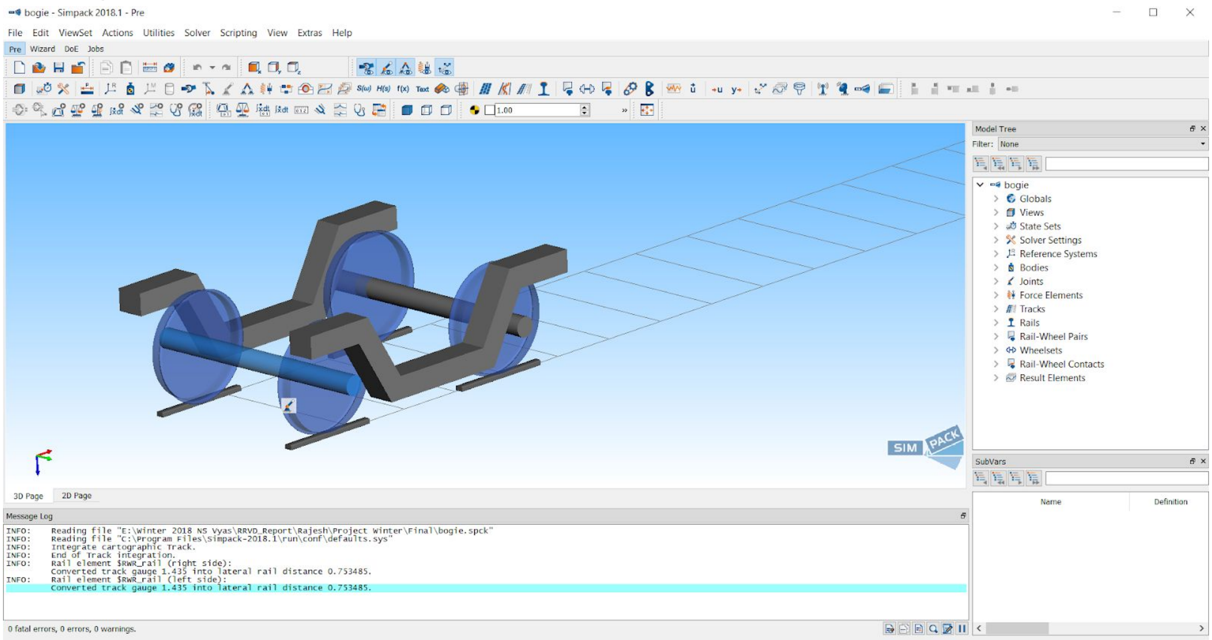


A model of the bogie was made but we did not proceed ahead with this model because of the complexity involved and were trying with the simple models of bogies.

Folder : RRVD_Report\Rajesh\Project Dr. NS Vyas\model
File: boggie
Remarks: Not running. It is Just a model made.



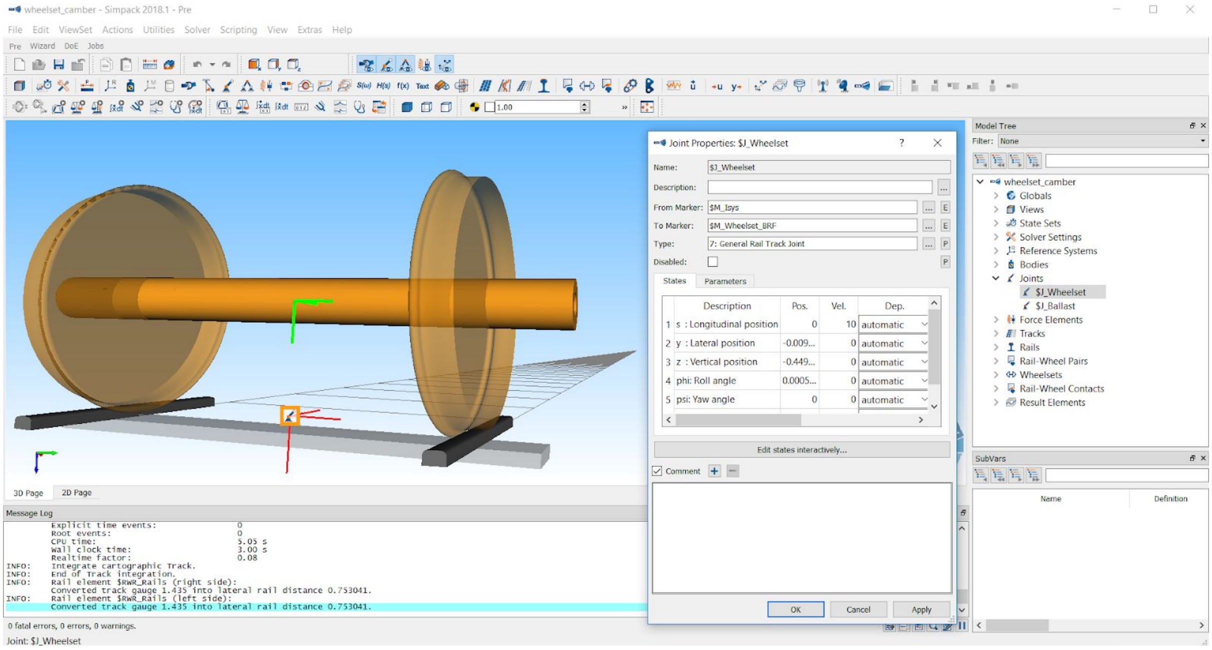
Folder : Folder: RRVD_Report\Rajesh\Project Winter\Final
File: bogie



Uncertainties imparted to the wheelset

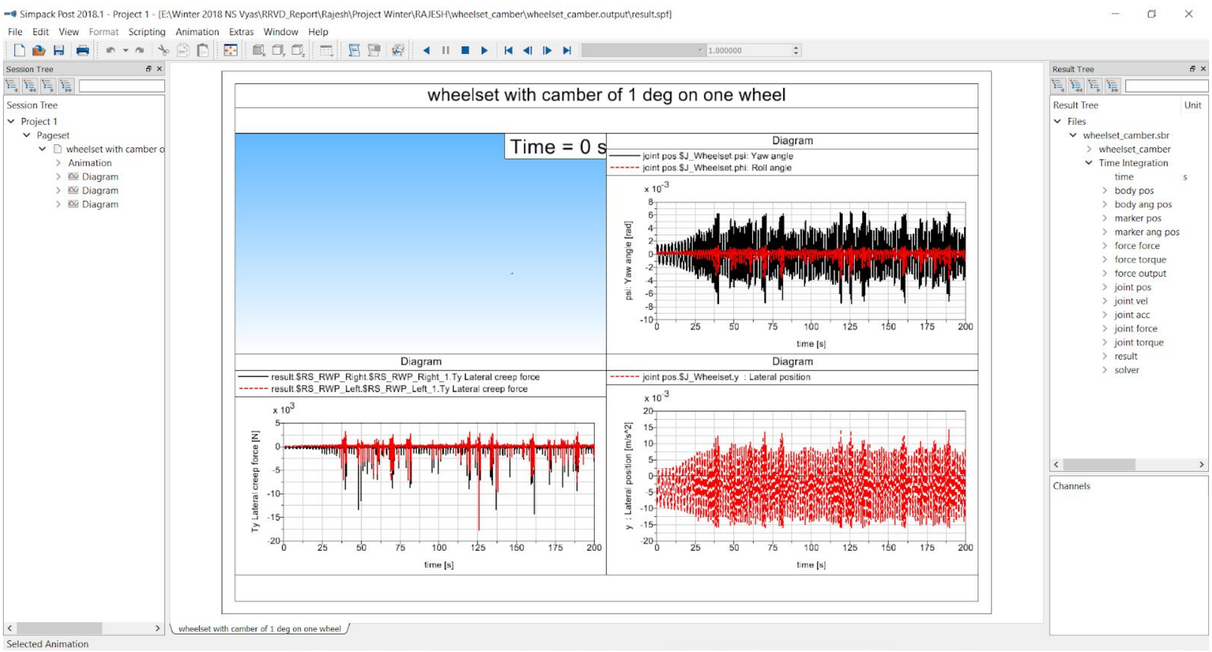
1. Camber

Folder: RRVD_Report\Rajesh\Project Winter\RAJESH\wheelset_camber
File: wheelset_camber



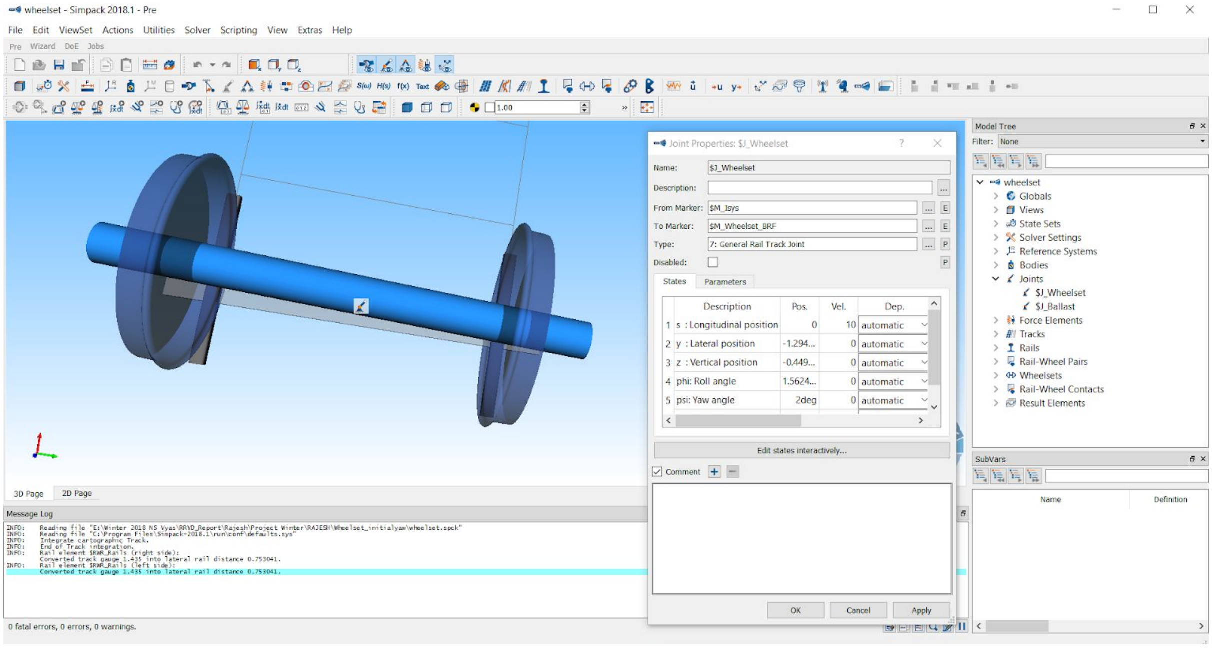
Camber of 1 degree was the imparted to the wheelset and its motion behavior was visualized using Simpack post. The next figures show the output signals as well as the animation.

Folder: RRVD_Report\Rajesh\Project Winter\RAJESH\wheelset_camber\wheelset_camber.output
File: result



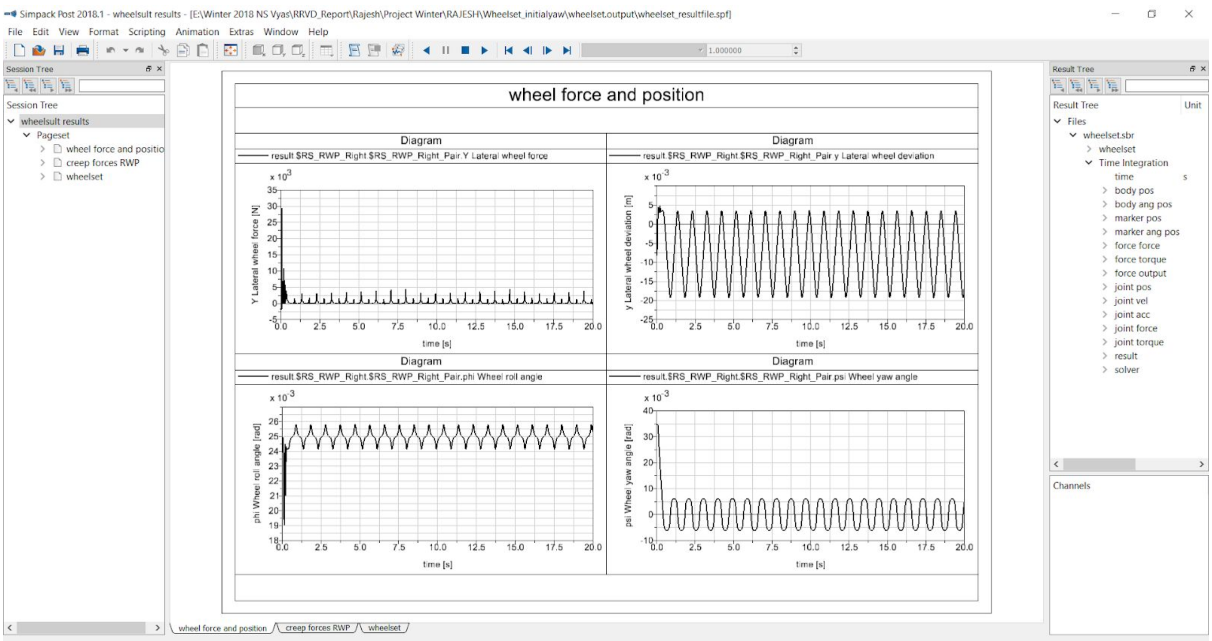
2. Yaw

Folder: RRVD_Report\Rajesh\Project Winter\RAJESH\Wheelset_initialyaw
File: wheelset



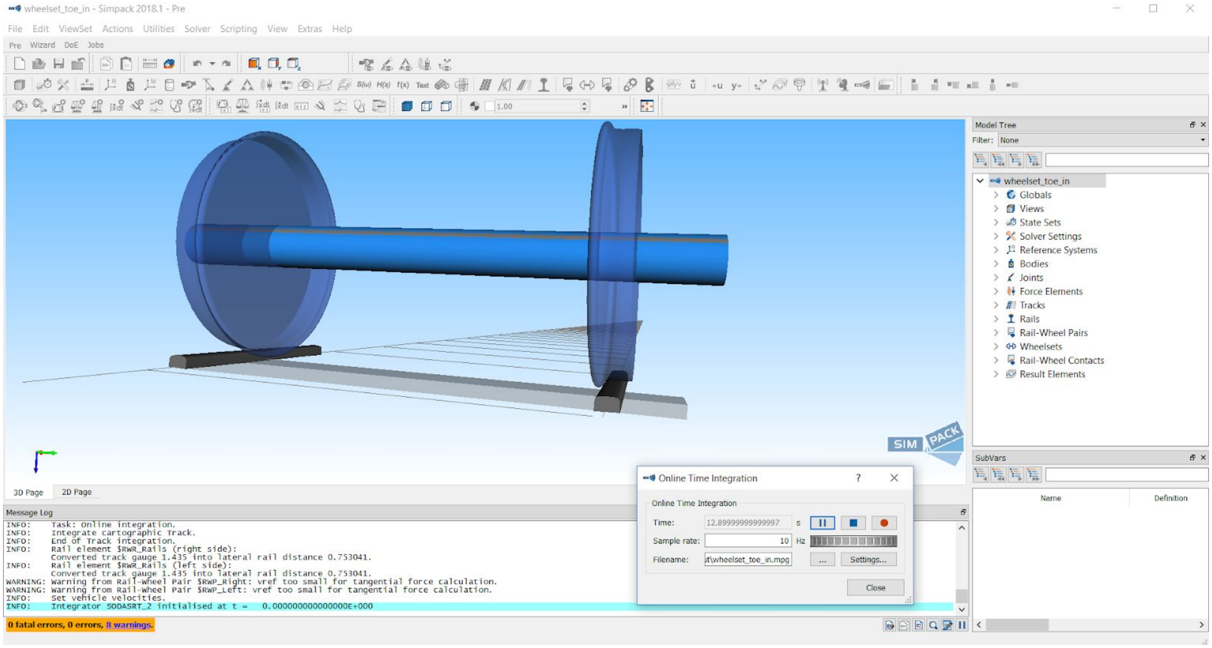
Initial yaw of 2 degrees was imparted to the wheelset and its motion behavior was visualized using Simpack post. The next figures show the output signals as well as the animation.

Folder: RRVD_Report\Rajesh\Project Winter\RAJESH\Wheelset_initialyaw\wheelset.output
File: wheelset_resultfile



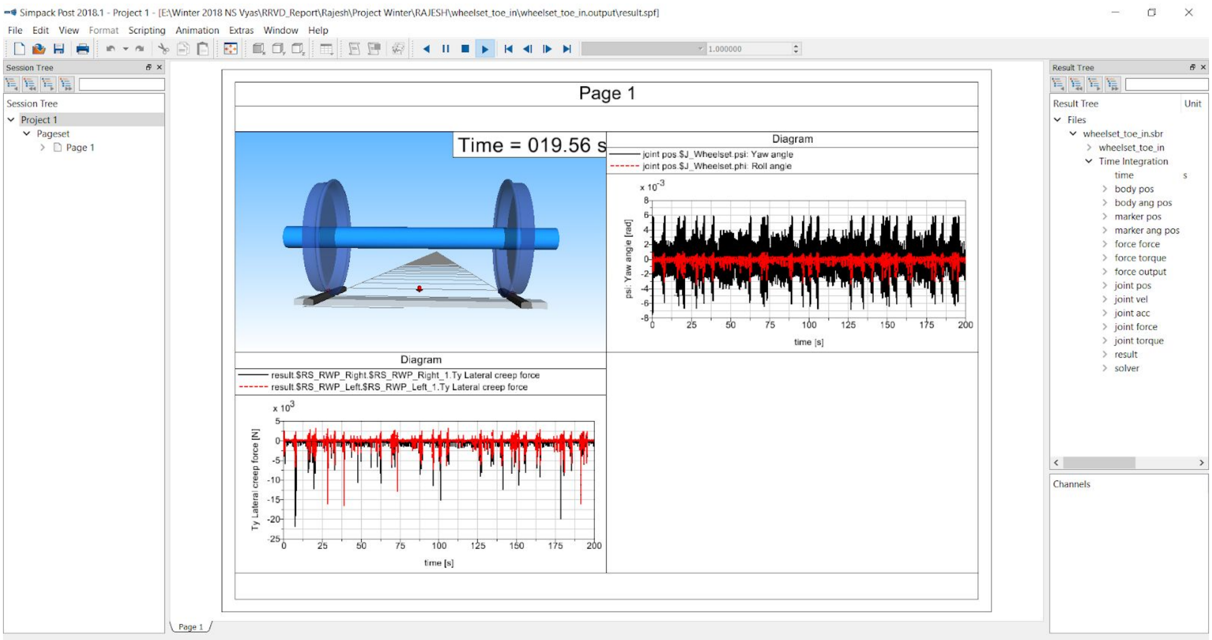
3. Toe in

Folder : RRVD_Report\Rajesh\Project Winter\RAJESH\wheelset_toe_in
File : wheelset_toe_in



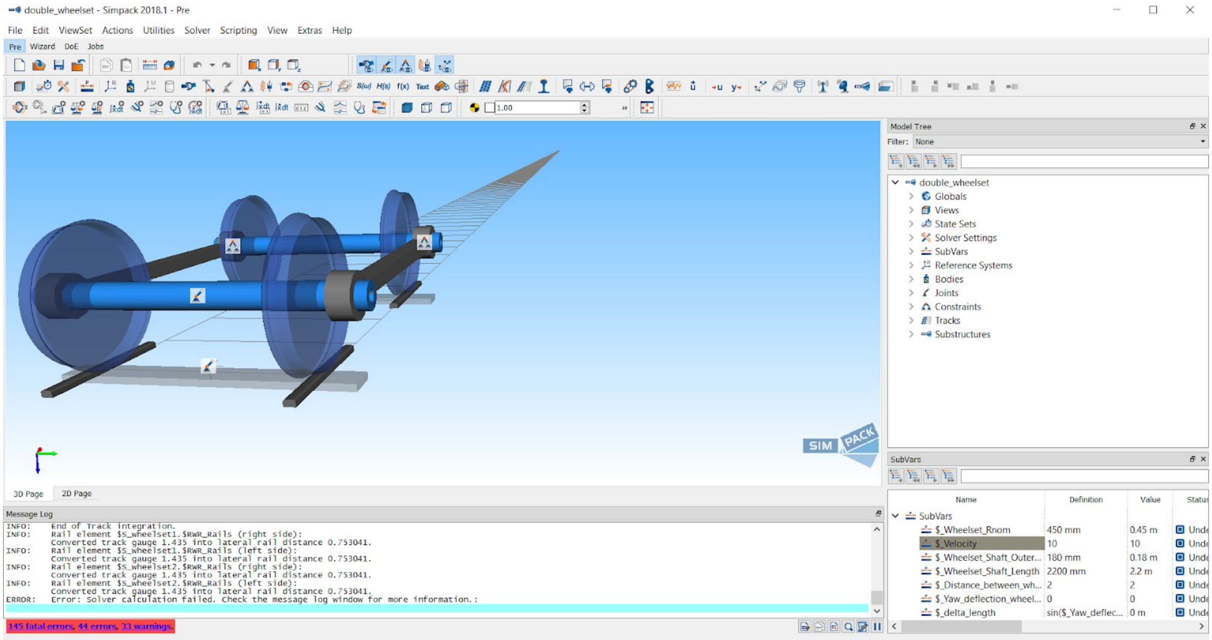
An initial toe in was given to the wheelset due to which the wheelset ran for some time but eventually derailed. The below figures from SImpack post show the motion behaviour on imparting the toe in to the wheelset.

Folder : RRVD_Report\Rajesh\Project Winter\RAJESH\wheelset_toe_in\wheelset_toe_in.output
File: result



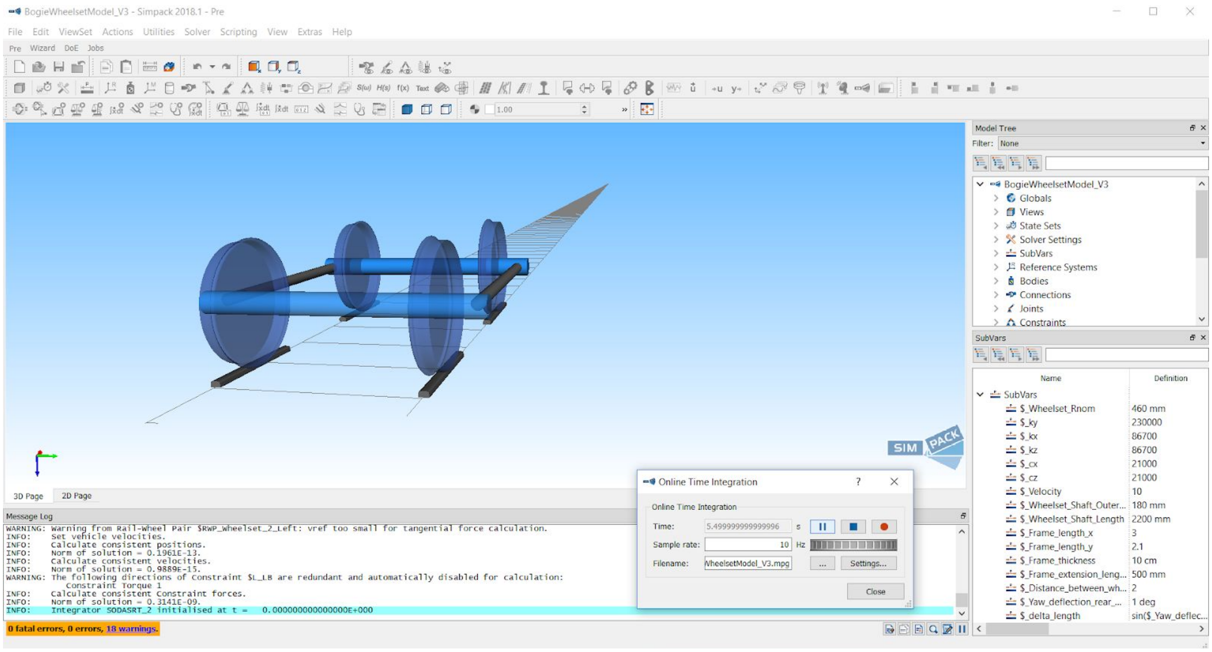
The wheelsets were connected on both sides by means of axles. The joints between the wheelset and the axle was created in such a way to mimic the bearing. In the axle the degree of freedom corresponding to the rotation of the wheelset was left free while all the other degrees of freedom were locked. But the 4 joints were over constrained (the solution is yet to be figured out) and the model did not run correctly.

Folder: RRVD_Report\Rajesh\winters\v3_Aditya
File: double_wheelset

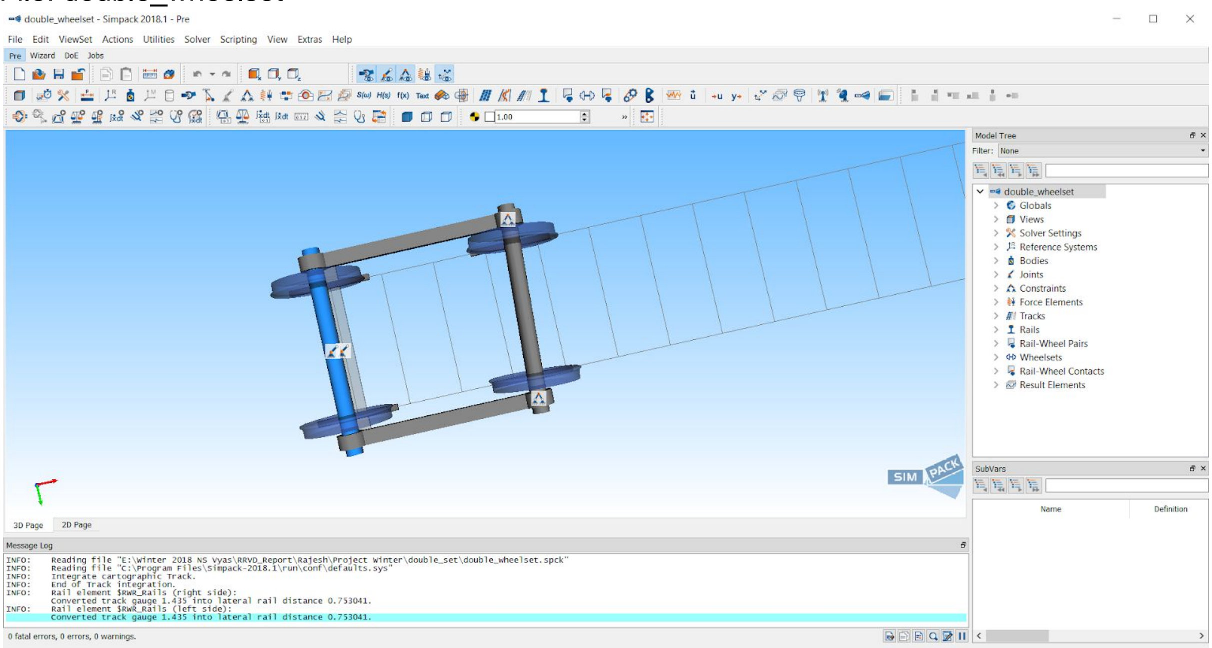


We tried to impart uncertainty in the full wheelset also by giving it a yaw of 2 degrees. The model is running but the yaw given is automatically getting adjusted to zero because of maybe joint constraints and this needs to be figured out so that the motion behavior can be analysed. We had also tried to give some uncertainties by making one axle a little smaller than the other axle so that one of the wheelsets gets a yaw and tried to study the motion behavior but were not successful.

Folder: RRVD_Report\Rajesh\winters
File: BogieWheelsetModel_V3

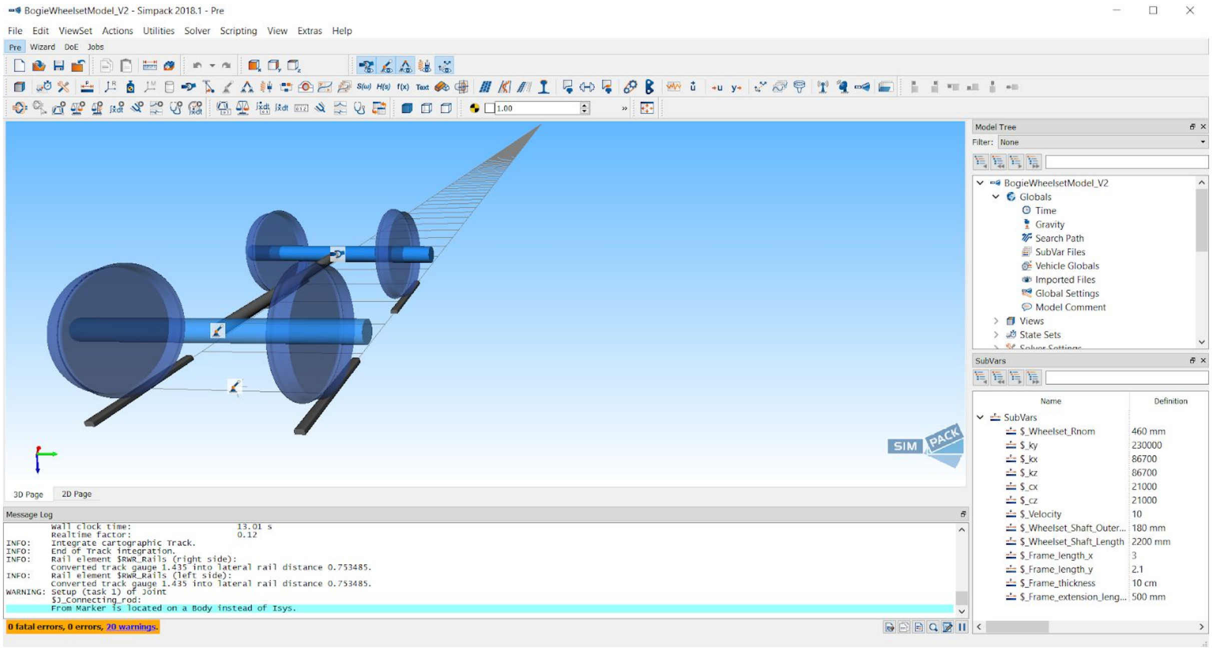


Folder: RRVD_Report\Rajesh\Project Winter\double_set
File: double_wheelset

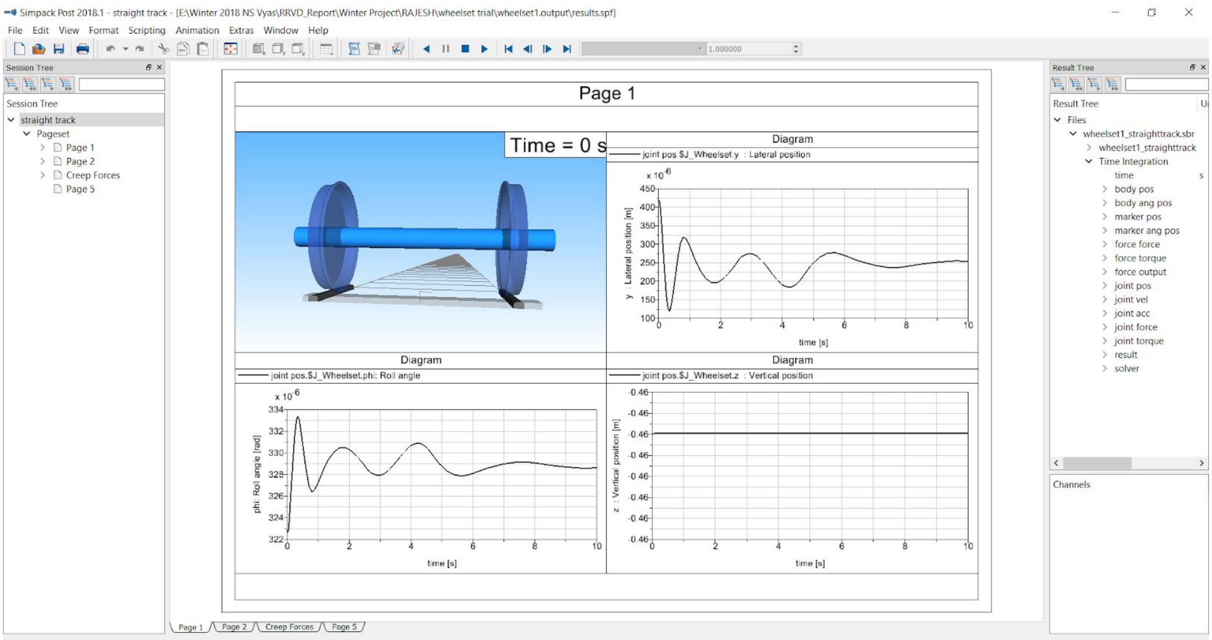


The two wheelsets were connected using a single axle and two joints on the axle were defined as a joint and a connection. These joints had only one degree of freedom free so that it allowed the wheelsets to have an angular velocity and all the degrees of freedoms were locked.

Folder: RRVD_Report\Rajesh\winters
File : BogieWheelsetModel_V2

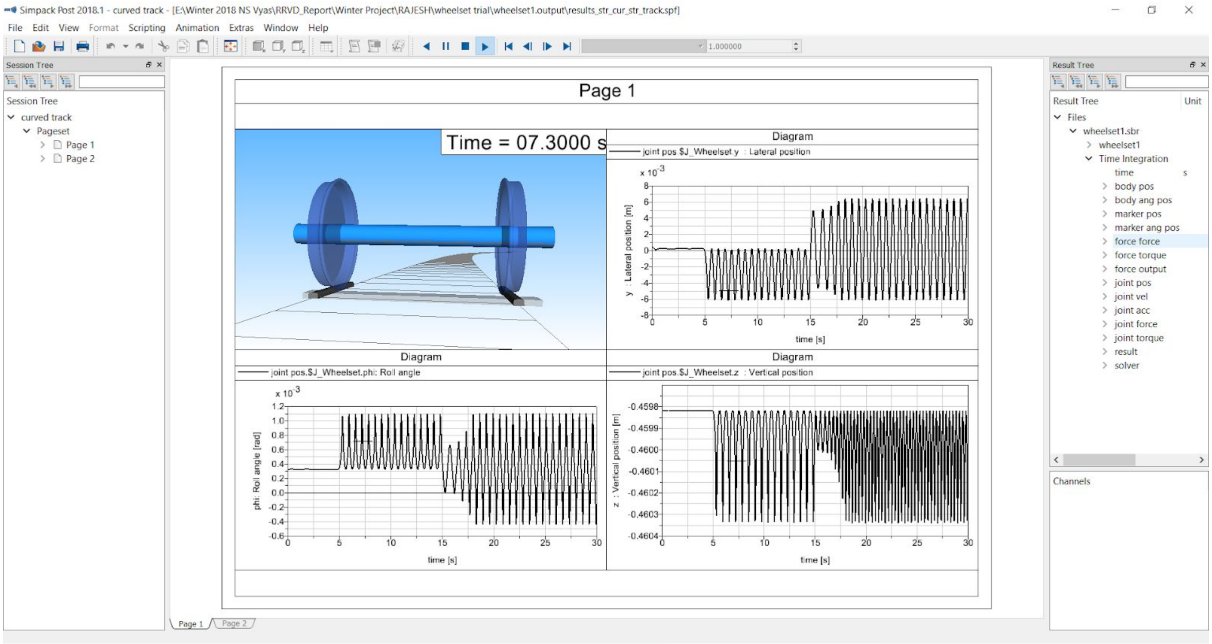


Folder: RRVD_Report\Winter Project\RAJESH\wheelset trial\wheelset1.output
File: results



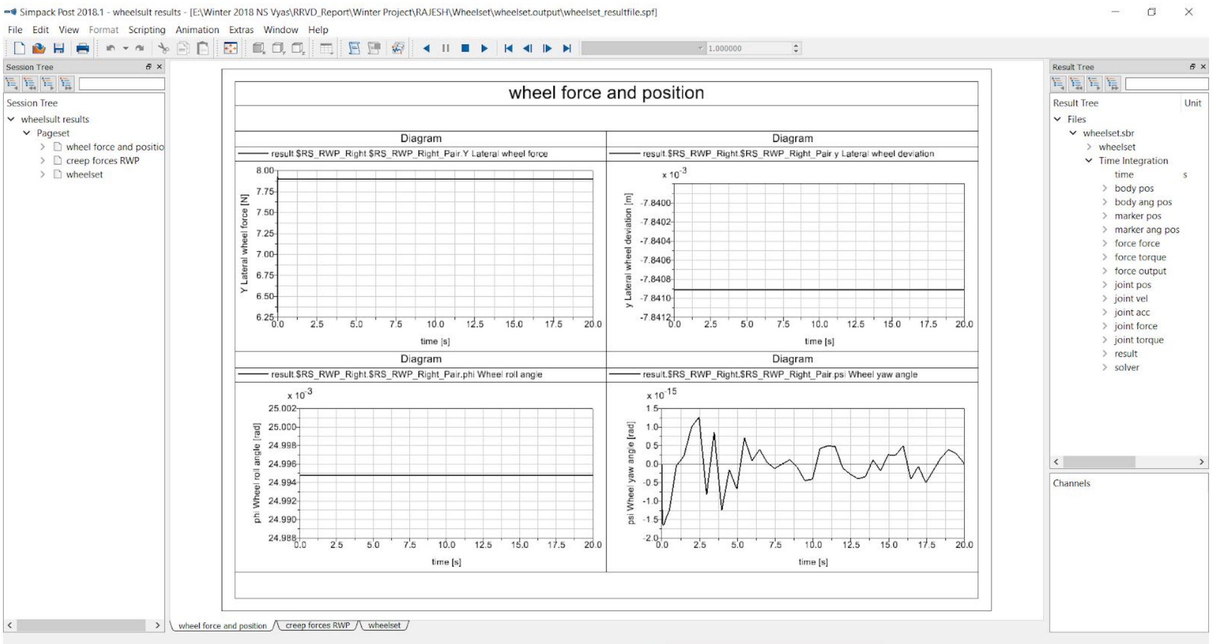
Motion of wheelset on a curved track

Folder: RRVD_Report\Winter Project\RAJESH\wheelset trial\wheelset1.output
File: results_str_cur_str_track



A curved track was made and the motion behavior of wheelset was observed which one can see in the next figure. This was done so as to see how the wheelset negotiates a radius of curvature.

Folder: RRVD_Report\Winter Project\RAJESH\Wheelset\wheelset.output
File: wheelset_resultfile



Moving load on a beam using Ansys

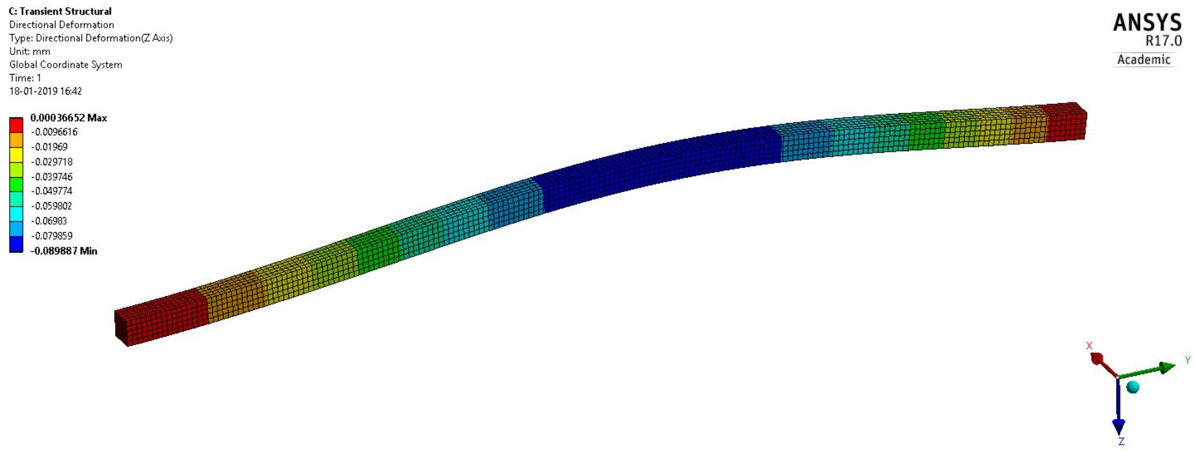
A beam with moving load was modelled to see the motion behavior of the rail track when the train is running on it.

The beam is made of structural steel of dimension 190X5X5m3. The beam is made with multiple thin features on one of the surface using ANSYS Geometry Modeller.

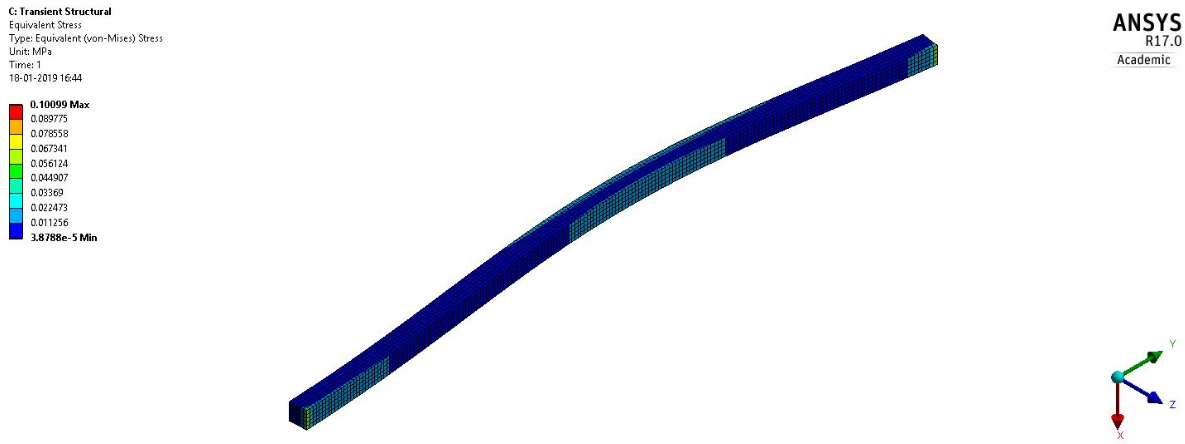
A 2D Static structural analysis is performed for a moving load on the beam. The moving load is imparted through a time-dependent forcing input which is applied on multiple sections of the face. The face of 5X190m is divided into 19 equal sections with different instants of forcing so that it approximately mimics a moving load problem.

The beam was meshed using Mesh Sizing method using cubic elements of 1m3 . Below is the image of the meshed beam model.

The figure below shows the total deformation of the beam when a force of 10kN is applied.
The beam is fixed in all degrees of freedom at the opposite edges at the two corners of the beam.



The figure below the Von Mises stress for the above given force and boundary conditions.



This problem need to be extended to a circular beam but there were problems in creating thin features on the circular inner cross-section in ANSYS. We weren't able to divide the circular cross-section into multiple sub-sections for applying moving load as done approximately in the above problem.

Beam crack modeling

ANSYS Mechanical APDL was used to model a beam with a crack.

The beam is modeled as two half sections with solid 8 node brick elements. They are meshed individually and then all the nodes are connected except at the crack. A crack of 10% is initiated in the beam at the center. Structural analysis was performed on the beam for a constant force at one of the ends with the other end fixed.

The moving load problem was not implemented on APDL beam model.