MCT412

****

Autotronics

****

Collected Models Report

Created by:

Mostafa Medhat Farook 17P8134

Table of Contents

1.0 Introduction

As we studied various systems related to Autotronic in the Couse MCT412 and studied its equations and conditions we were asked to make **three models** from what we have studied.

1. MATLAB Model for conventional car braking system and including ABS cycle algorithm and tire model.
2. MATLAB model of conventional and active suspension.
3. MATLAB model of active steering actuator is required ***(we may use physical modelling for this model only).***

2.0 Conventional Braking & ABS Algorithm

2.1 Braking Torque Section

```
%*************************************************************************%
%**************************   Braking Torque   ****************************%
%*************************************************************************%

% Pedal Force (in N). %
Max_Human_Force = 150;                % in Newton
Human_Force = 100;                    % in Newton
Pedal_ratio = 5;                      % L1 / L2
Pedal_Force = Human_Force * Pedal_ratio;

% Servo Model. %
Servo_P = 29450;                      % Servo (Booster) Pressure (in Pa)
Servo_R = 0.08;                       % Servo (Booster) radius (in m)
servo_F = Servo_P * pi * Servo_R^2; % Servo (Booster) force (in N)

% Total Force (in N). %
Servo_Ratio = Human_Force / Max_Human_Force;
Total_Force = Pedal_Force + (servo_F * Servo_Ratio);

% Master Cylinder Pressure (in Pa). %
Master_Cyl_R = 0.01;                  % in m
Master_Cyl_P = Total_Force / pi / Master_Cyl_R^2;

% Braking Torque (in N.m) (Using Fixed Disk Brake). %
Friction_Coef = 0.3;                  % Disc friction coef.
Disk_R = 0.0254;                      % in m
Disk_R_outer = 200;                   % in mm
Disk_R_inner = 0.6 * Disk_R_outer;   % in mm
numi = 2 * (Disk_R_outer^3 - Disk_R_inner^3);
domi = 3 * (Disk_R_outer^2 - Disk_R_inner^2);
Disk_R_mean = numi / domi / 1000;    % in m

Fn = Master_Cyl_P * pi * Disk_R^2;   % in N
Braking_Torque = Fn * Friction_Coef * 2 * Disk_R_mean;
```

2.2 Acceleration & Velocity & Distance

```
%*************************************************************************%
%****************************  A & V & D  ********************************%
%*************************************************************************%

% First we assume some parameters to calculate deceleration. %
Rw = 0.3;                           % Wheel radius in m
Weight = 15000;                     % in N
syms deceleration

% Deceleration. %
Braking_Force = Braking_Torque / Rw;% in N
dec = 4 * Braking_Force * 9.8 / Weight;

% Velocity (in m/s) (integration). %
Velocity_v1 = double(int(dec, deceleration, 0, dec));

% Distance (in m) (integration). %
speed = int(dec, deceleration);
Distance_v1 = double(int(speed, deceleration, 0, dec));

%********************** Kinematic Equation Method ************************%

% Velocity (in m/s) (kinematic equation). %
% To calculate velocity we assume the change in time and initial speed
delta_t = 4;                        % in seconds
init_v = 28;                        % in m/s (100.8 km/hr)
Velocity_v2 = init_v-(dec*delta_t); % in m/s

% Distance (in m) (kinematic equation). %
Distance_v2 = 0.5 * delta_t * (init_v - Velocity_v2);
```

2.3 Tire Table (Magic Formula)

```
%**************************************************************************%
%****************************   Tire Table   ******************************%
%**************************************************************************%

syms k
Fz = Weight / 4;                    % Vertical load on tire in N
E = 0.97;                           % Curvature
D = 1;                              % Peak
C = 1.9;                            % Shape
B = 10;                             % Stifness
step = 5;                           % Angle Step
road_coef = 0.8;

% Magic Formula with Constant Coefficients %
% Fx = f(κ,Fz) = Fz·D·sin(C·arctan{Bκ−E[Bκ−arctan(Bκ)]}) %
Fx = Fz*D*sin(C*atan(B*k - (E*(B*k - atan(B*k)))));
ang = 0:step:100;
Fx_arr = subs(Fx, k, ang);
%Fx_arr = (Fx_arr / max(Fx_arr)) * road_coef;

t = tiledlayout(2,2);
nexttile
plot(ang, Fx_arr)
title('Tire Graph')
```

2.4 Max Driver Force

```
%**************************************************************************%
%**************************   Max Driver Force   **************************%
%**************************************************************************%

% Calculate Road Max Torque
F_limit = road_coef * Fz;        % Fz = W/4 (one tire)
T_max = F_limit * Rw;            % Max available torque

% Max Main Cylinder Force
F_MC_max = ((T_max*(2*Master_Cyl_R)^2) /...
            (2*Friction_Coef*Disk_R_mean*(2*Disk_R)^2));

% Max Driver Braking Force (theoritically) (in N)
% After that force the tire will begin to slip
F_Driver_Max = floor((F_MC_max - servo_F) / Pedal_ratio);
```

2.5 Borch ABS Algorithm

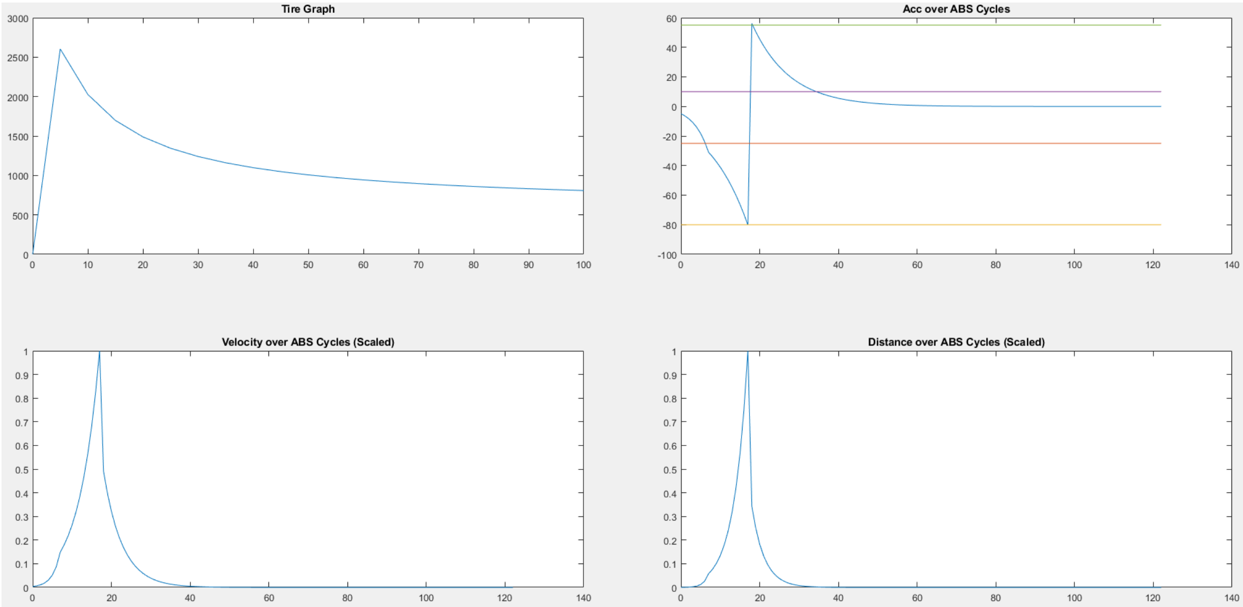
```
%**************************************************************************%
%************************  Borch ABS Algorithm  ***************************%
%**************************************************************************%

a1 = -25;
a2 = -80;
a3 = 10;
a4 = 80;
a = -dec;
slope = -1;

% Velocity to stop from is the calculated above (in line 51)
Stop_from_vel = Velocity_v1;        % Velocity to stop from in km/s
F_Driver_Input = F_Driver_Max + 5;  % in N
p = Master_Cyl_P;
sign = -1;
counter = 0;

acc_arr = -dec;
vel_arr = Velocity_v1;
dis_arr = Distance_v1;

% Apply the algorithm as long as car does not stop, or the driver stops
% applying this strong force on braking pedal.
if (F_Driver_Input > F_Driver_Max)
    while (~(Stop_from_vel < 1e-6) && (F_Driver_Input > 20))
        if (a < a2)
            p = p * 0.7;    % Decrease pressure
            slope = 1;
            sign = -1;
        elseif ( (a1 >= a) && (a > a2) )
            if (slope < 0)
                p = p*1.1;  % Hold pressure (slight increase)
            else
                p = p*0.9;  % Hold pressure (slight decrease)
            end
            sign = -1;
        elseif ( (a3 >= a) && (a > a1) )
            if (slope < 0)  % Slope -ve
                p = p*1.3;  % Increase pressure
            else            % Slope +ve
                p = p*0.9;  % Hold pressure (slight decrease)
            end
            sign = -1;
```

```
        elseif ( (a4 >= a) && (a > a3) )
            if (slope < 0)
                p = p*1.1;  % Hold pressure (slight increase)
            else
                p = p*0.9;  % Hold pressure (slight decrease)
            end
            sign = 1;
        elseif (a > a4)
            p = p * 1.3;    % Increase pressure
            slope = -1;
            sign = 1;
        end
        [a, Stop_from_vel, d] = New_Status(p, slope);
        acc_arr(end+1) = a;
        vel_arr(end+1) = Stop_from_vel;
        dis_arr(end+1) = d;
        counter = counter + 1;
    end
end

x_axis = 0:1:counter;
a1_arr = zeros(counter+1, 1) + a1;
a2_arr = zeros(counter+1, 1) + a2;
a3_arr = zeros(counter+1, 1) + a3;
a4_arr = zeros(counter+1, 1) + 55;
vel_arr = vel_arr / max(vel_arr);
dis_arr = dis_arr / max(dis_arr);


nexttile
plot(x_axis, acc_arr, x_axis, a1_arr, x_axis,a2_arr, x_axis, a3_arr,...
     x_axis, a4_arr)
title('Acc over ABS Cycles')

nexttile
plot(x_axis, vel_arr)
title('Velocity over ABS Cycles (Scaled)')

nexttile
plot(x_axis, dis_arr)
title('Distance over ABS Cycles (Scaled)')
```

2.6 Local Function used in Algorithm

```
%**************************************************************************%
%******************** Function to calculate (A & V & D)*********************%
%**************************************************************************%

function [new_a_out, New_v, New_d] = New_Status(pressure, sign, Disk_R,...
                                Friction_Coef, Disk_R_mean, Rw, Weight)

if (nargin < 3)
    Disk_R = 0.0254;
    Friction_Coef = 0.3;
    Disk_R_mean = 0.16334;
    Rw = 0.3;
    Weight = 15000;
end

Fn_new = pressure * pi * Disk_R^2;  % in N
Braking_Torque_new = Fn_new * Friction_Coef * 2 * Disk_R_mean;

syms deceleration_new

% Deceleration. %
Braking_Force_new = Braking_Torque_new / Rw;% in N
new_a = 4 * Braking_Force_new * 9.8 / Weight;
if (sign < 0)
    new_a_out = -new_a;
else
    new_a_out = new_a;
end

% Velocity (in m/s). %
New_v = double(int(new_a, deceleration_new, 0, new_a));

% Distance (in m). %
speed = int(new_a, deceleration_new);
New_d = double(int(speed, deceleration_new, 0, new_a));

end
%********************************* END **********************************%
```

2.7 Output (Graphs)



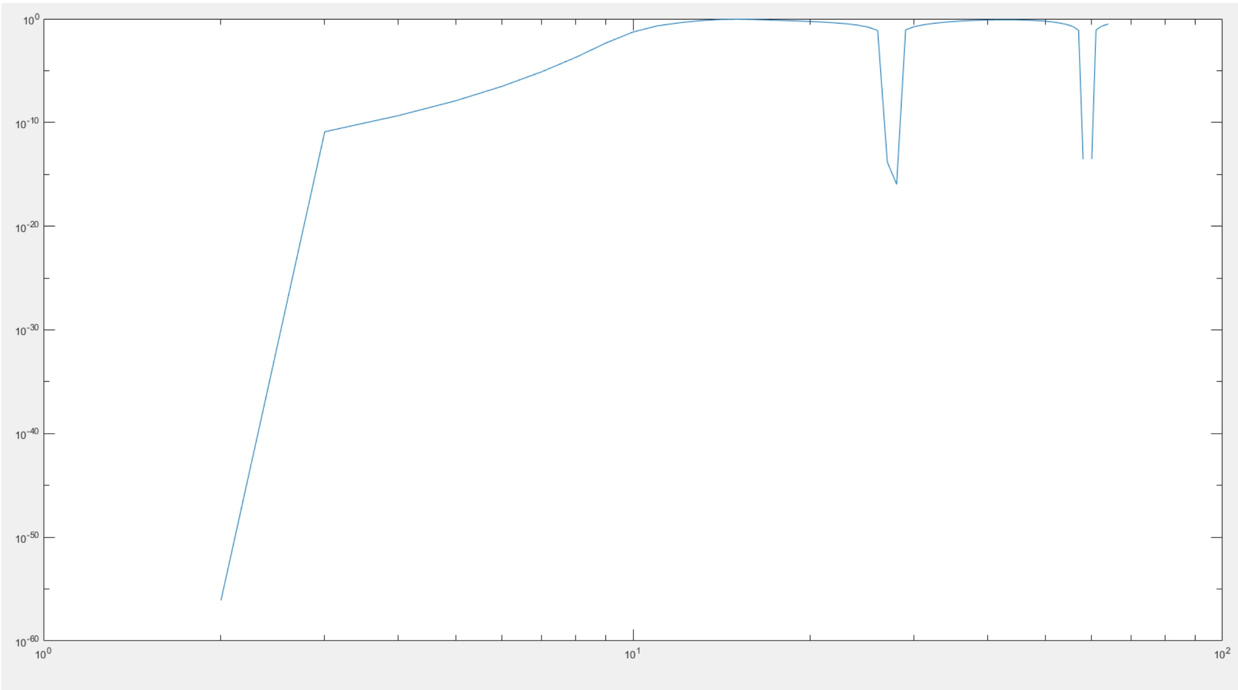
3.0 Conventional Suspension & Active Suspension

3.1 M File

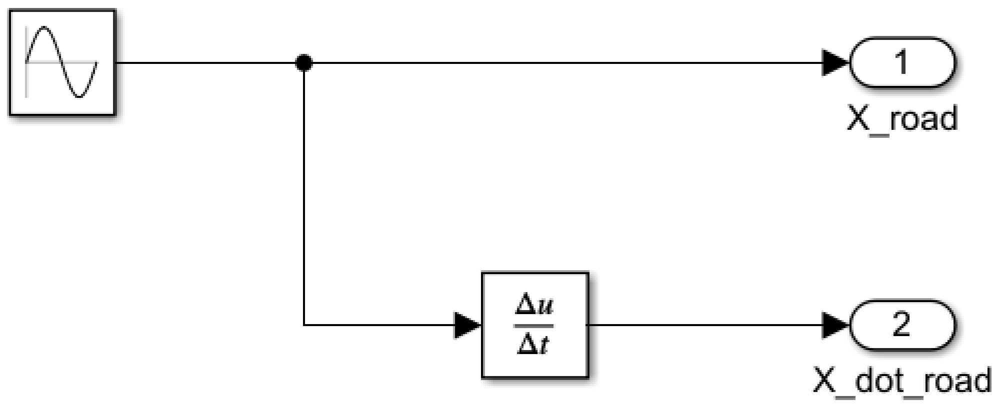
The M file will define the parameters used in **Simulink**, after that run the Simulink file and the conventional Simulink file passes the suspension travel to a variable ***(out)*** then plot the suspension travel with log scale ***(Ktr / Ks = 6.25)***, then set active suspension parameters and open both conventional and active suspension.

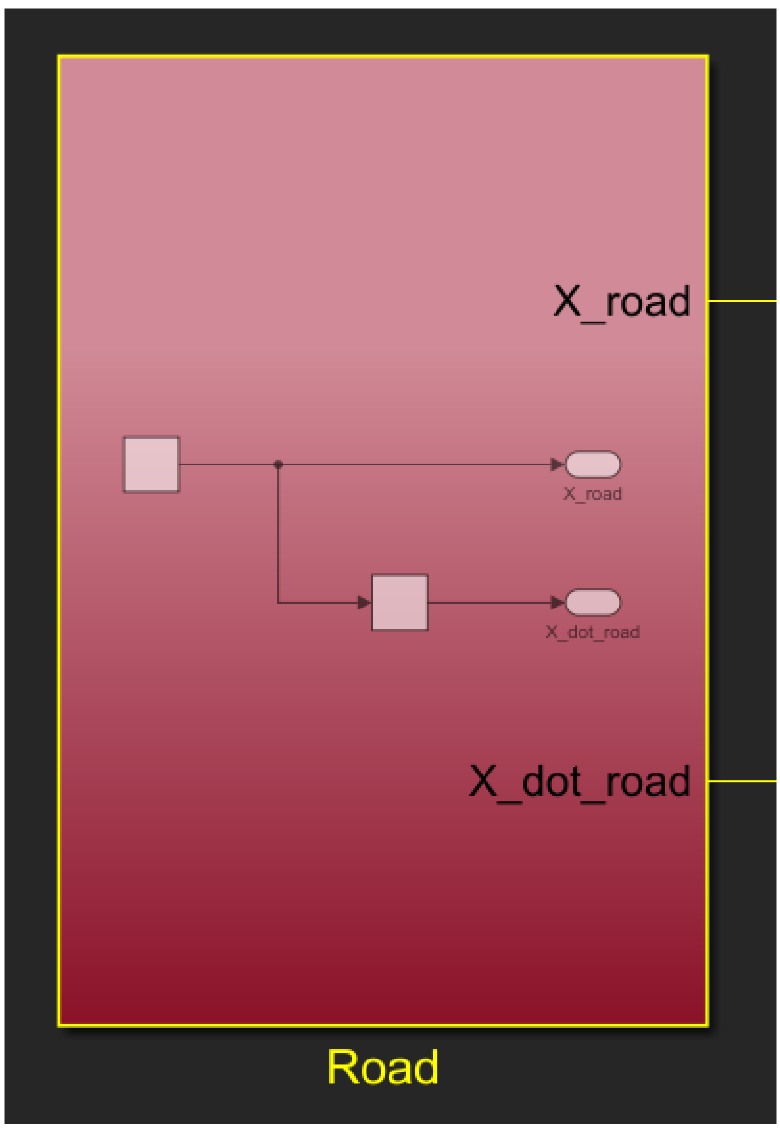
```
% Suspention Parameters
Ks = 80000; % Sus spring rate (N/m)
Cs = 350;   % Sus damping coeffient
Kt = 500000;% Tire spring rate (N/m)
Ct = 15020; % Tire damping coeffient
MR = 1;     % Motion ration (Damper)
Ms = 2500;  % Sprung mass (kg)
Mu = 320;   % Unsprung mass (kg)

% Open Conventional Sus model
open('Conv_Suspention.slx');

% Run the Simulink file
out = sim('Conv_Suspention.slx');

% Get susoention travel ratio from slx file
y = out.SUS_Travel.signals.values;
% Plot the result in log scale with (Ktr / Ks) = 6.25
loglog(y)

% Set active sus controller parameters
travel_sus_factor = 0.3;
dynamc_def_factor = 0.4;
sprung_acc_factor = 0.3;

% Open Active Sus model
open('Active_Suspention_v3.slx');
```

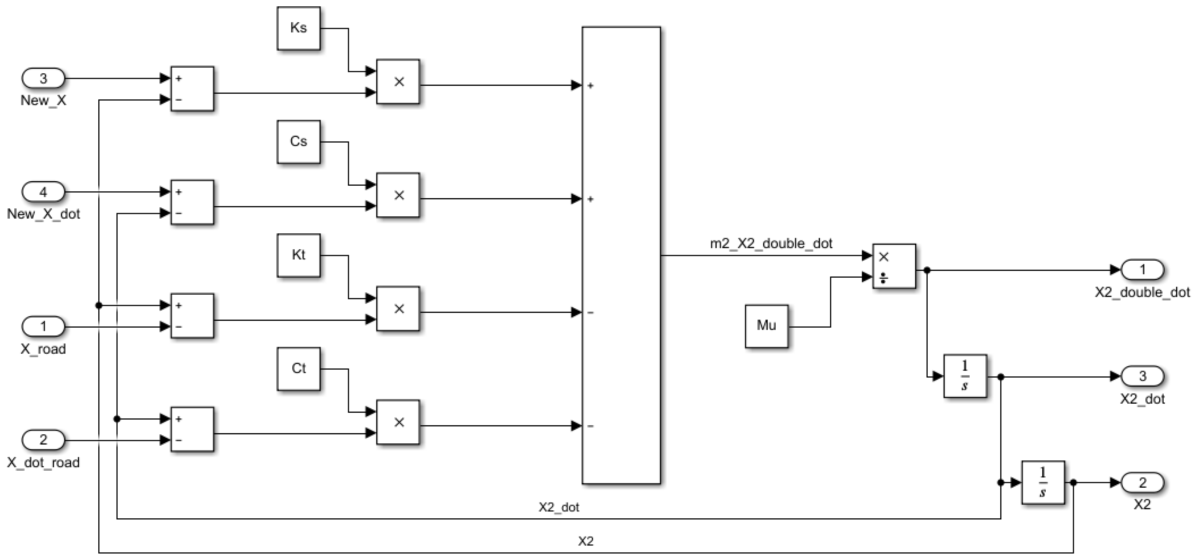
3.2 Suspension Travel Graph

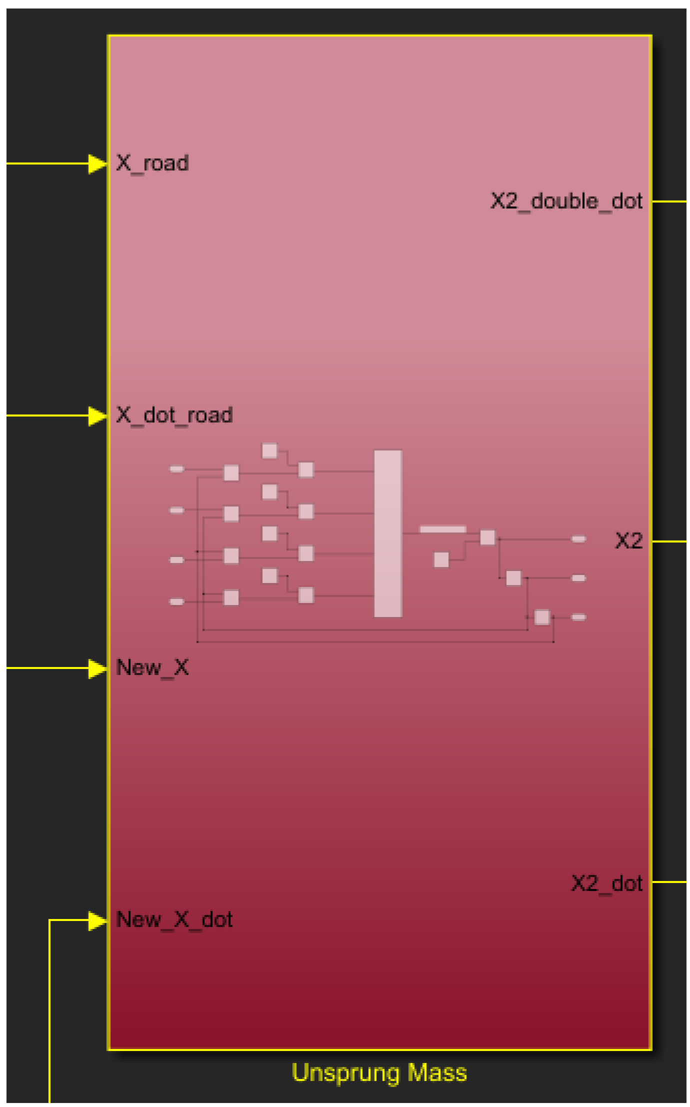


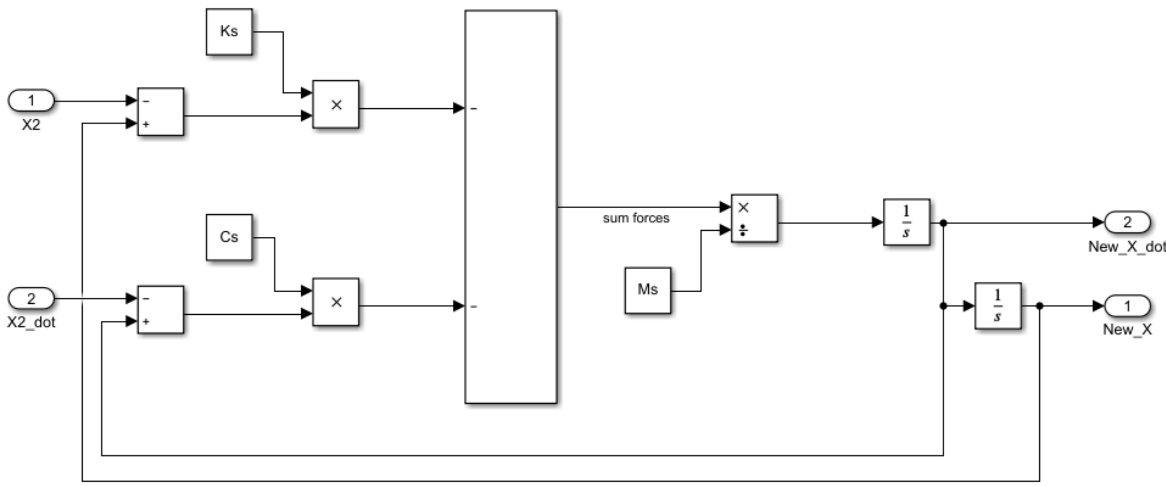
3.3 Conventional Suspension Model



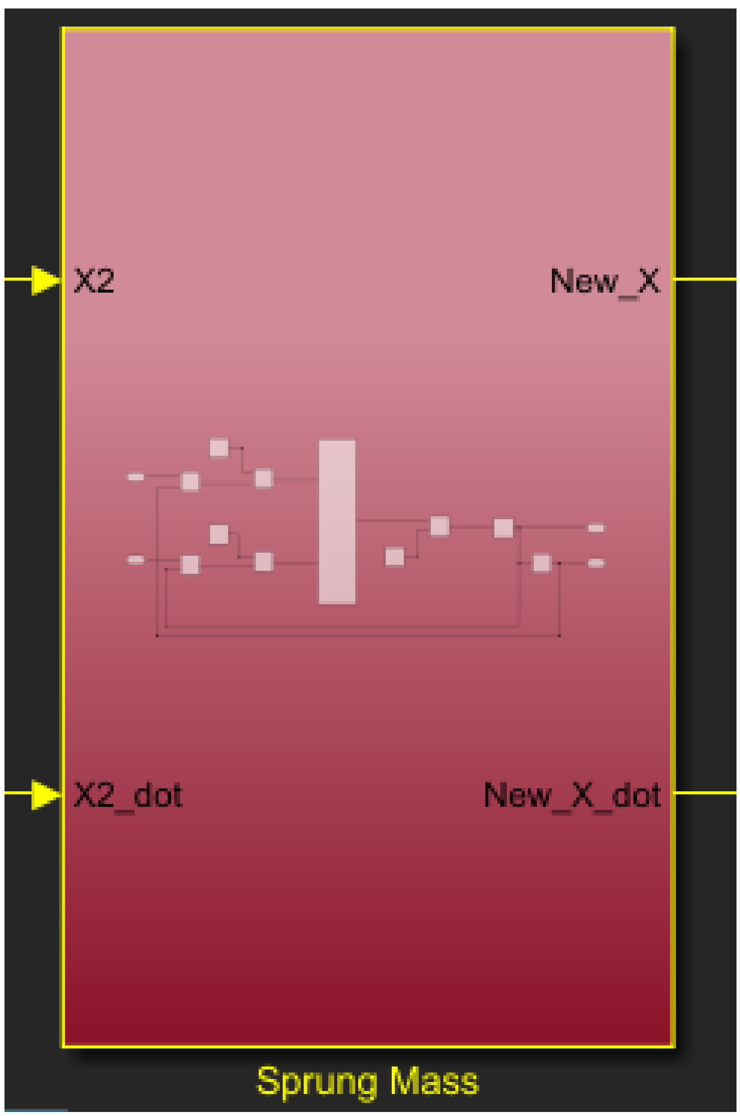



Ks
Cs
Kt
Ct
3
New_X
4
New_X_dot
1
X_road
2
X_dot_road
m2_X2_double_dot
Mu
1
X2_double_dot
3
X2_dot
2
X2
X2_dot
X2

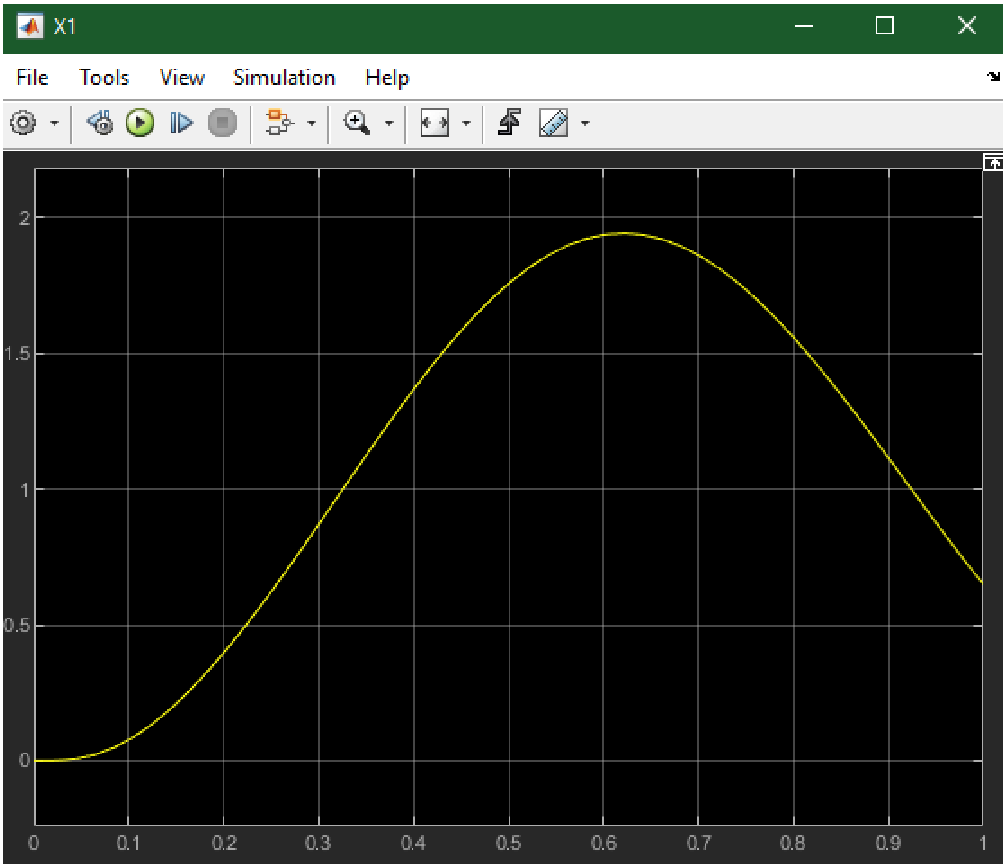

X_road
X2_double_dot
X_dot_road
X2
New_X
X2_dot
New_X_dot
Unsprung Mass

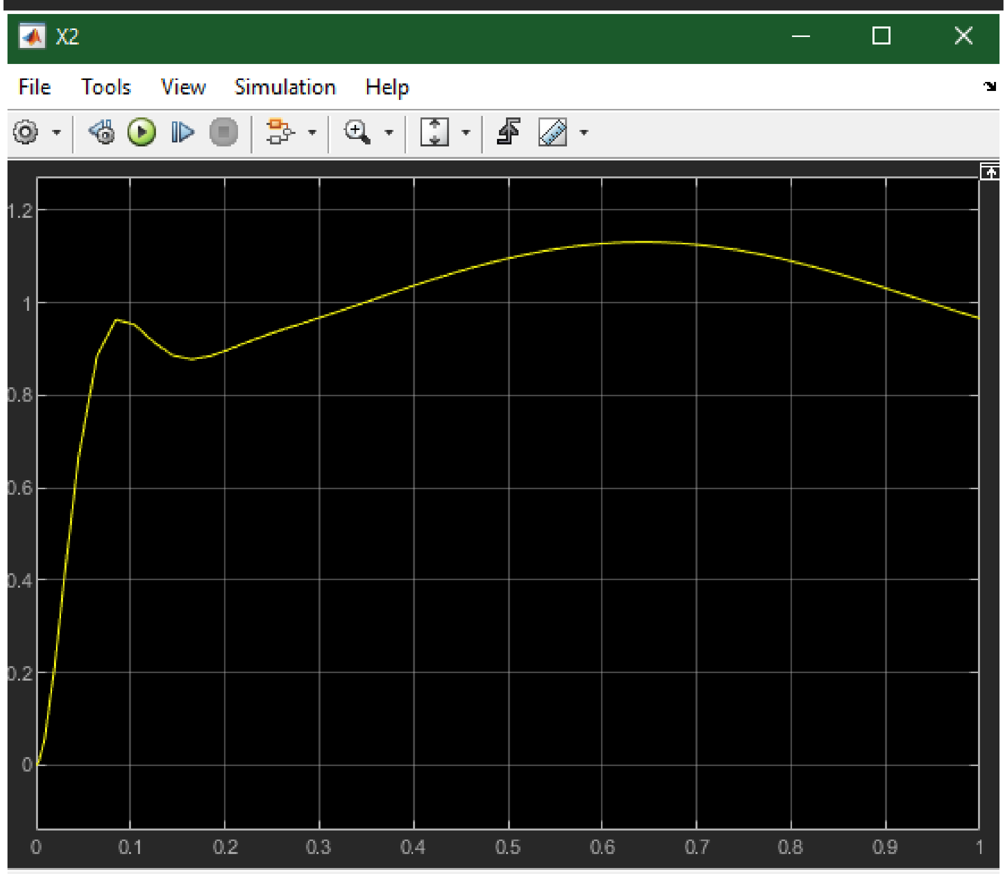


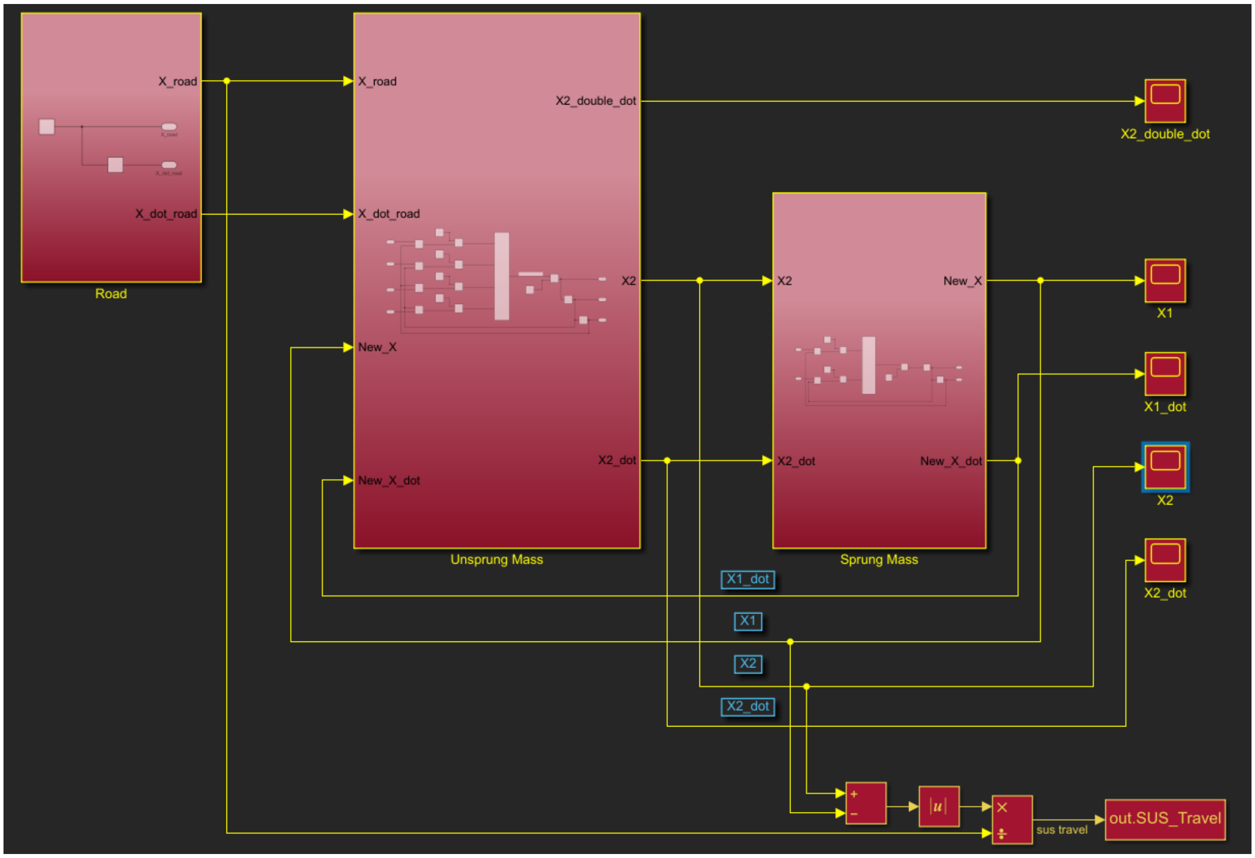
Ks
1
X2
Cs
2
X2_dot
sum forces
Ms
2
New_X_dot
1
New_X



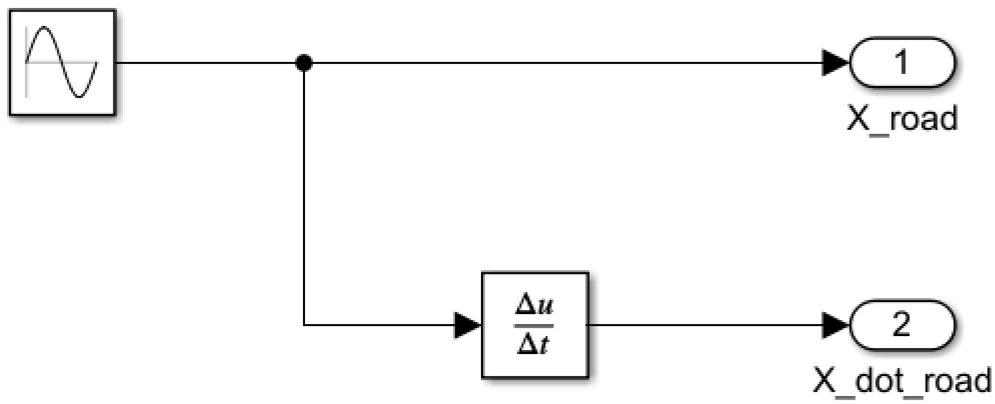
X2
New_X
X2_dot
New_X_dot
Sprung Mass

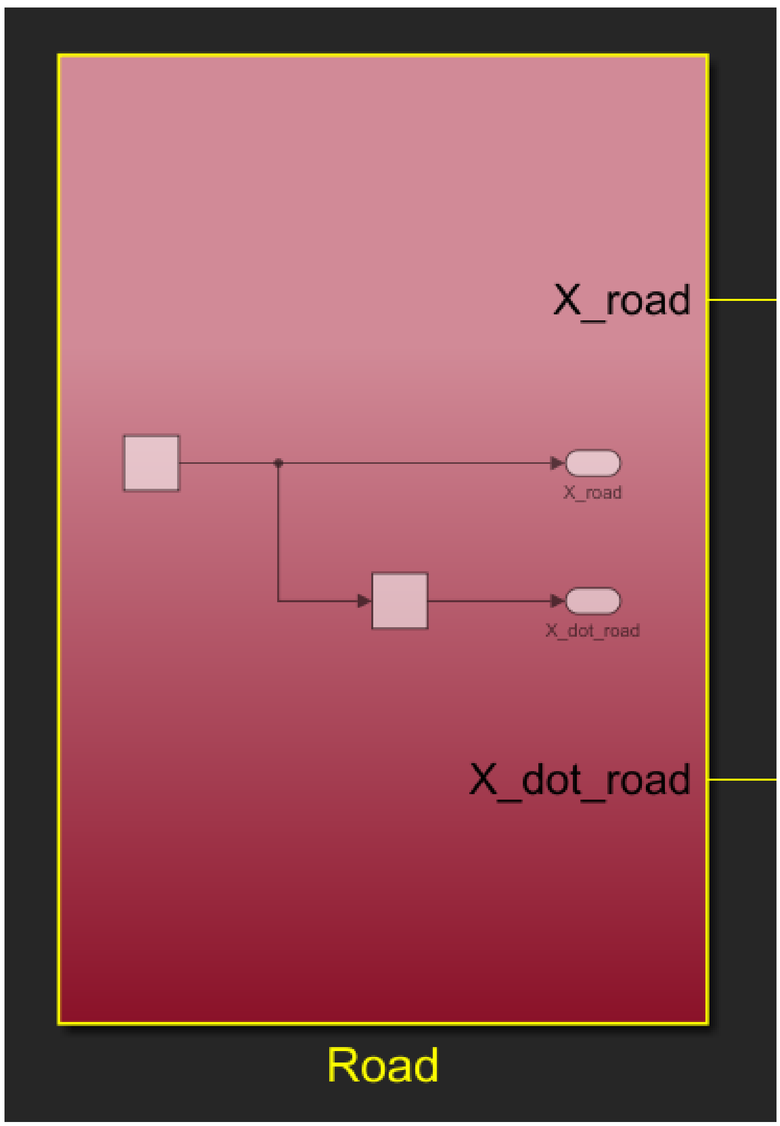

X1
File Tools View Simulation Help
2
1.5
1
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

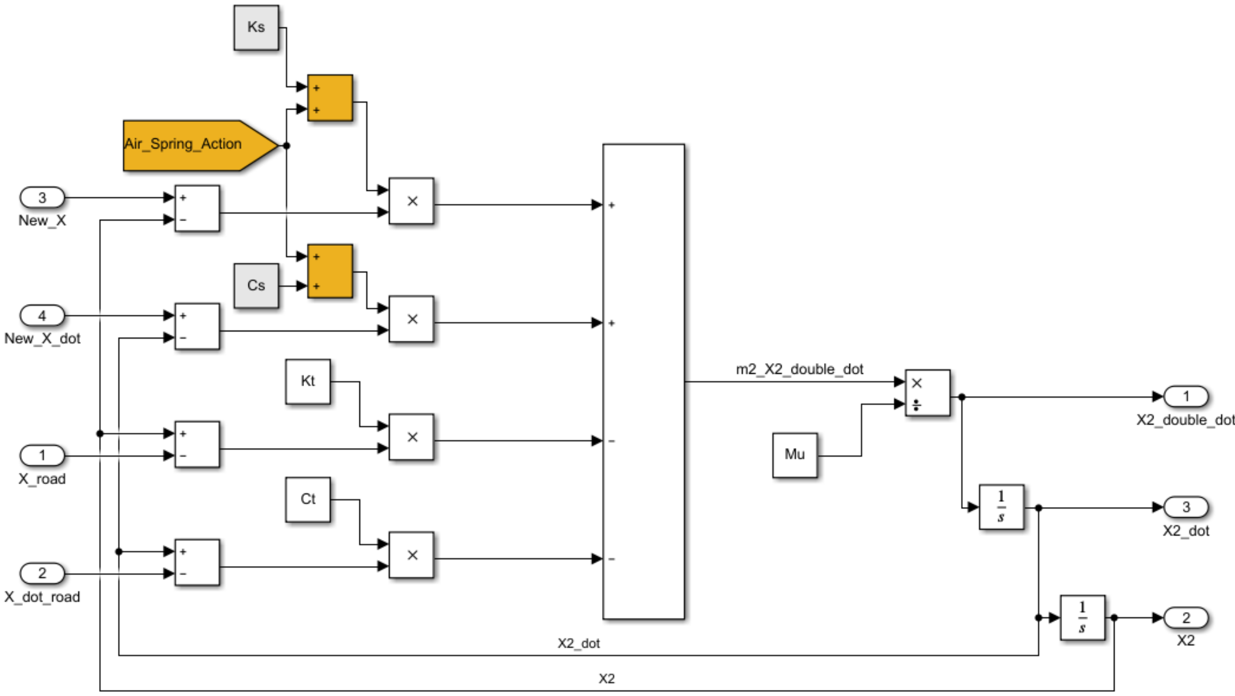

X2
File Tools View Simulation Help
1.2
1
0.8
0.6
0.4
0.2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

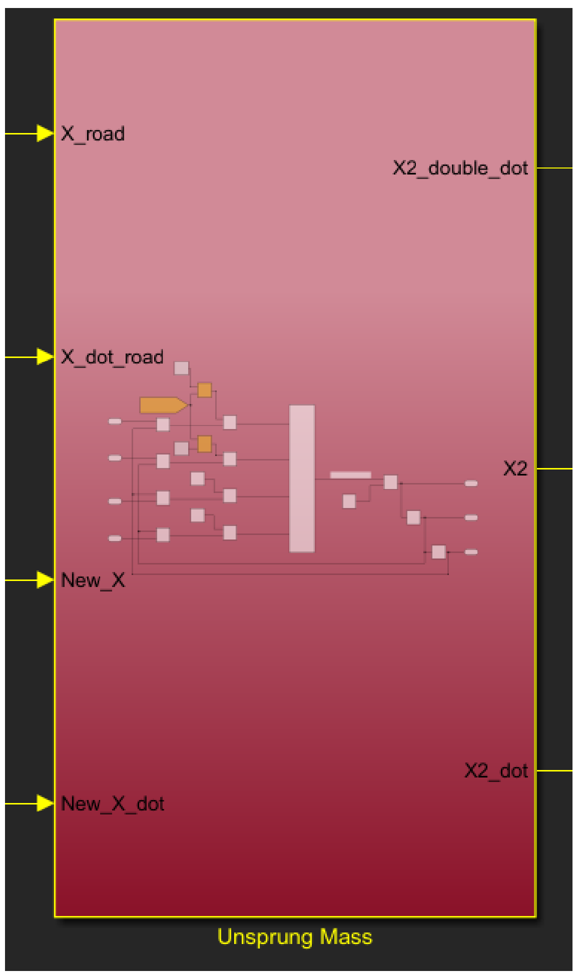

X_road
X_dot_road
Road
X_road
X_dot_road
New_X
New_X_dot
X2_double_dot
X2
X2_dot
Unsprung Mass
X2
X2_dot
New_X
New_X_dot
Sprung Mass
X2_double_dot
X1
X1_dot
X2
X2_dot
X1_dot
X1
X2
X2_dot
sus travel
out.SUS_Travel

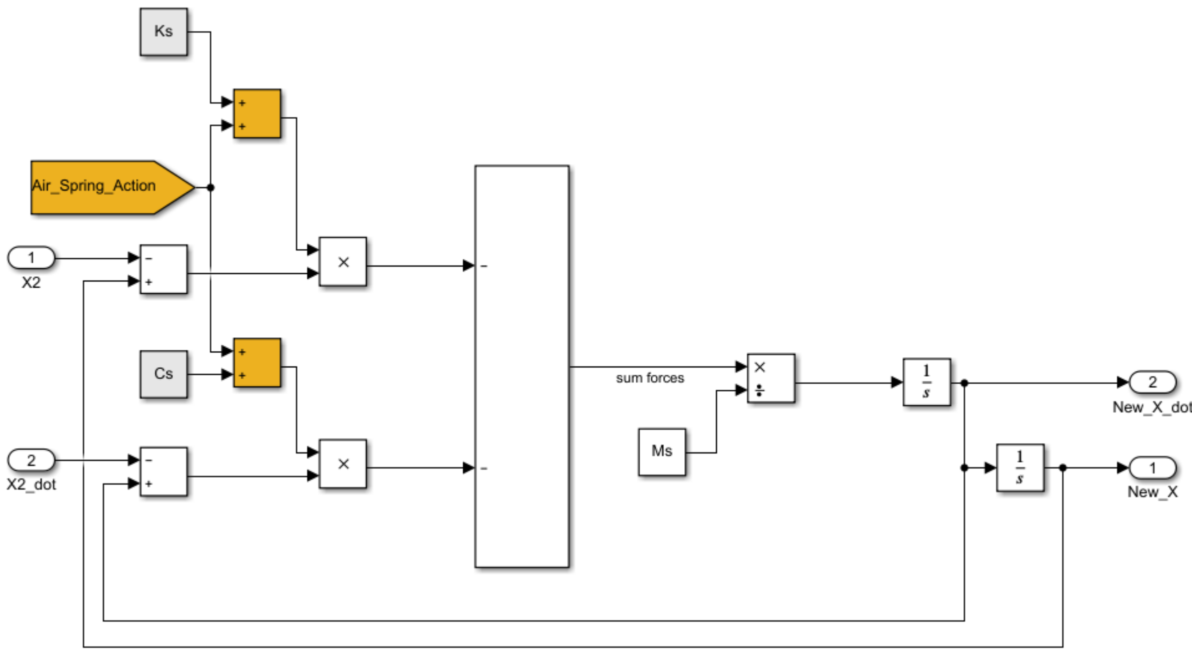
3.4 Active Suspension Model



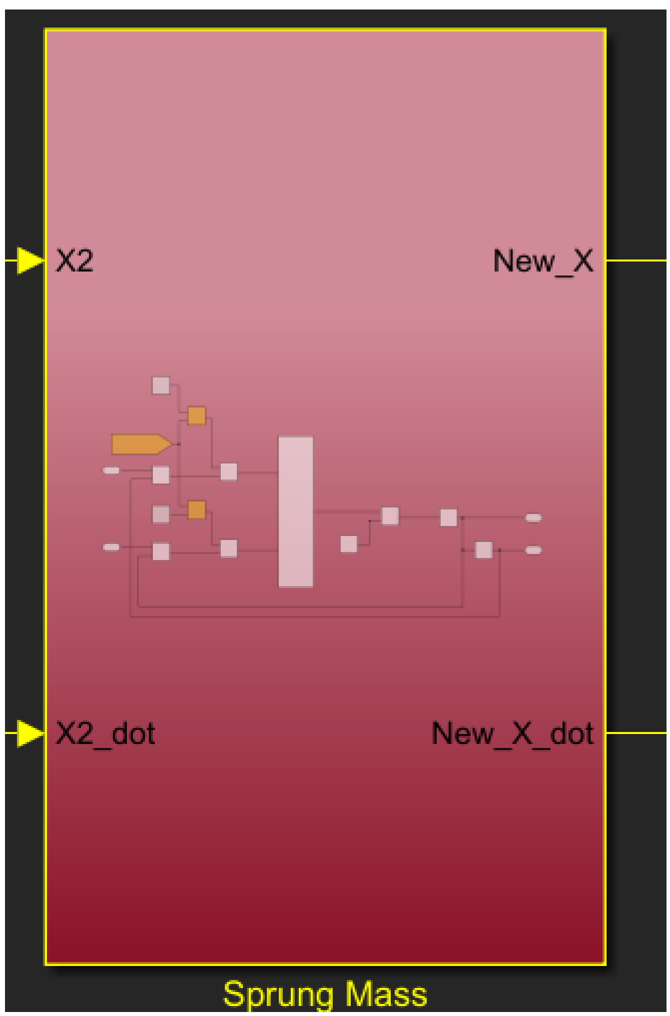



Ks
Air_Spring_Action
Cs
Kt
Ct
Mu
3
New_X
4
New_X_dot
1
X_road
2
X_dot_road
m2_X2_double_dot
1
X2_double_dot
3
X2_dot
2
X2
X2_dot
X2

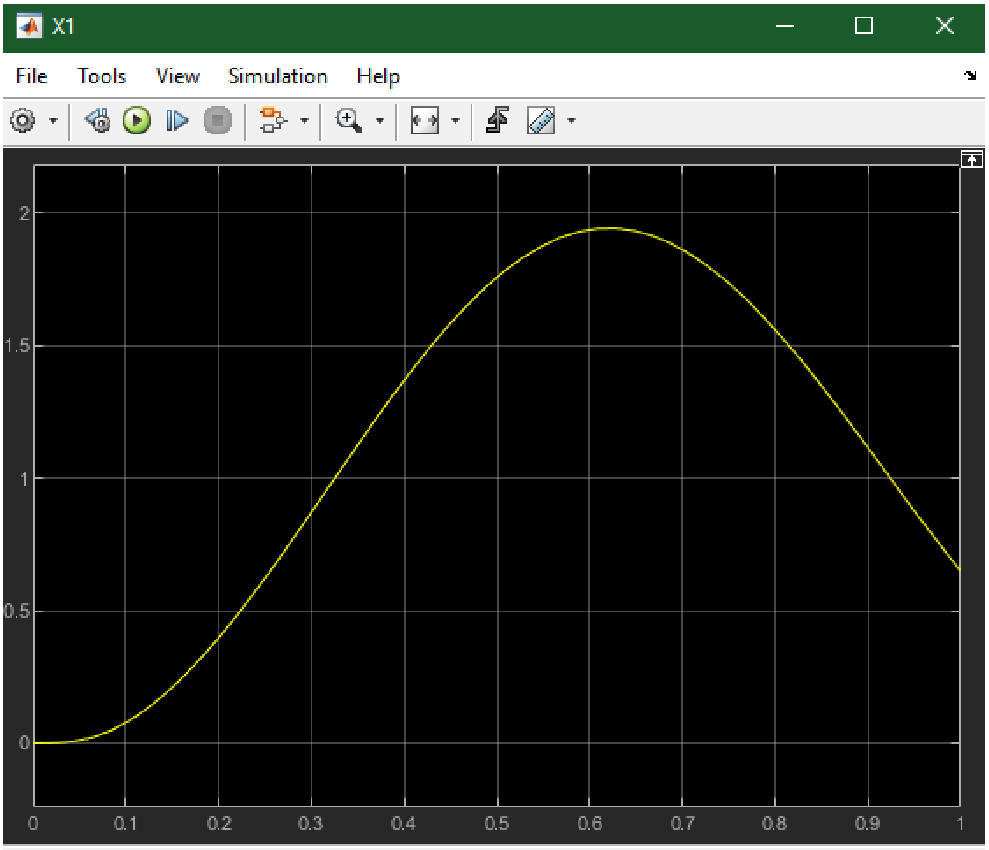

X_road
X_dot_road
New_X
New_X_dot
X2_double_dot
X2
X2_dot
Unsprung Mass



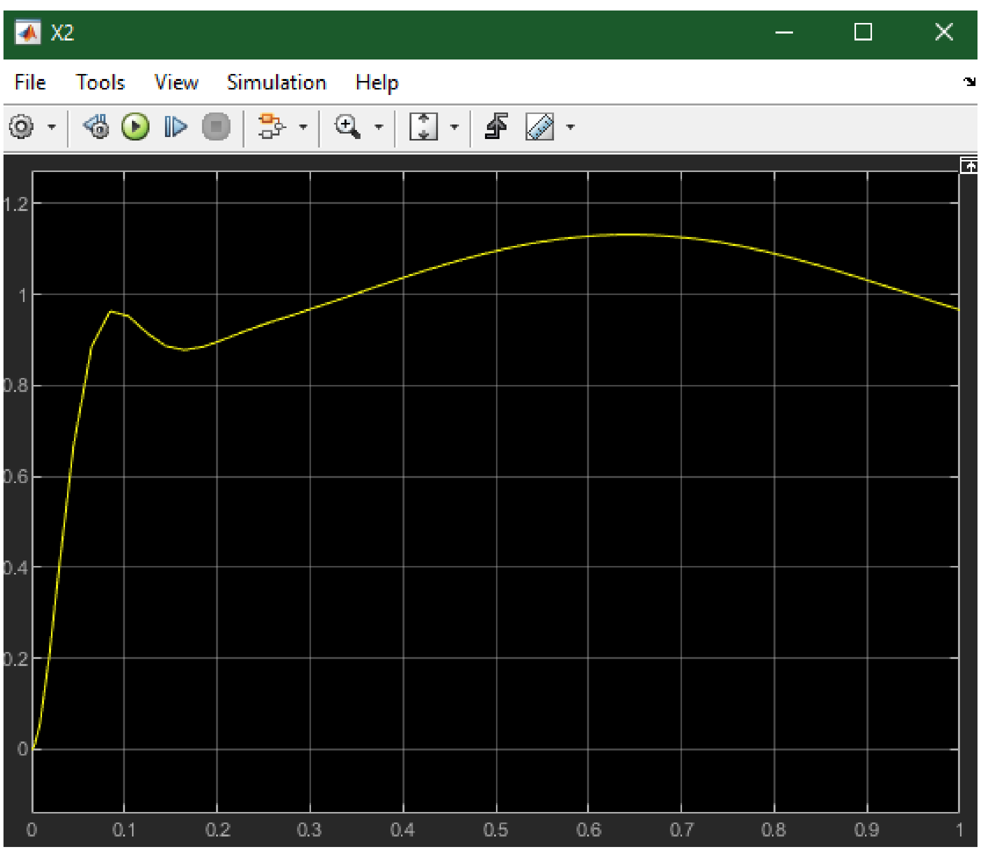
Ks
Air_Spring_Action
1
X2
Cs
2
X2_dot
sum forces
Ms
2
New_X_dot
1
New_X



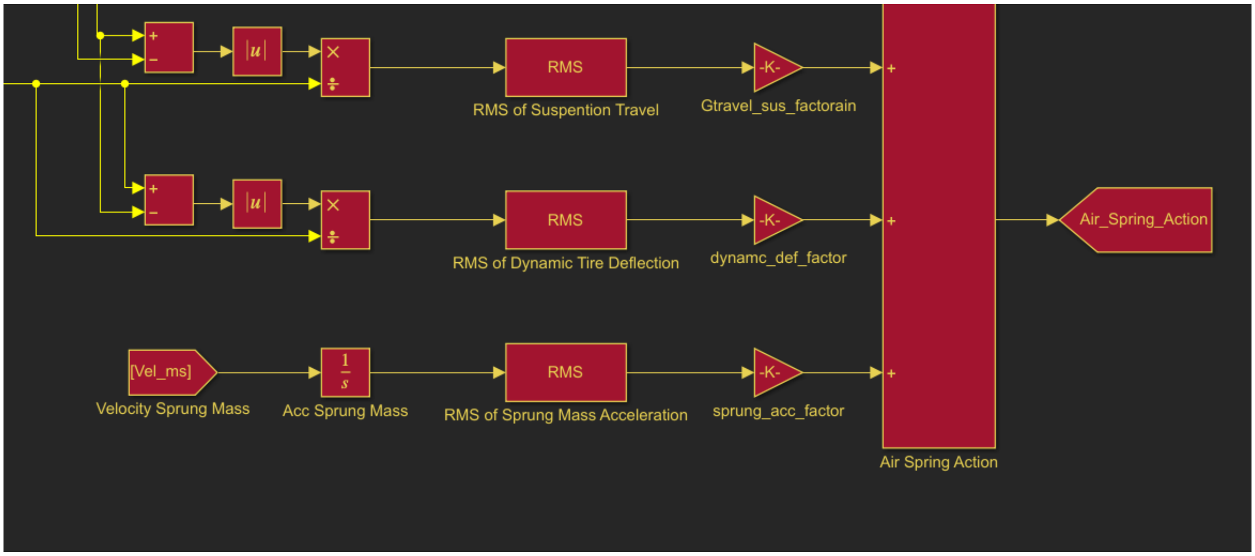
X2
New_X
X2_dot
New_X_dot
Sprung Mass

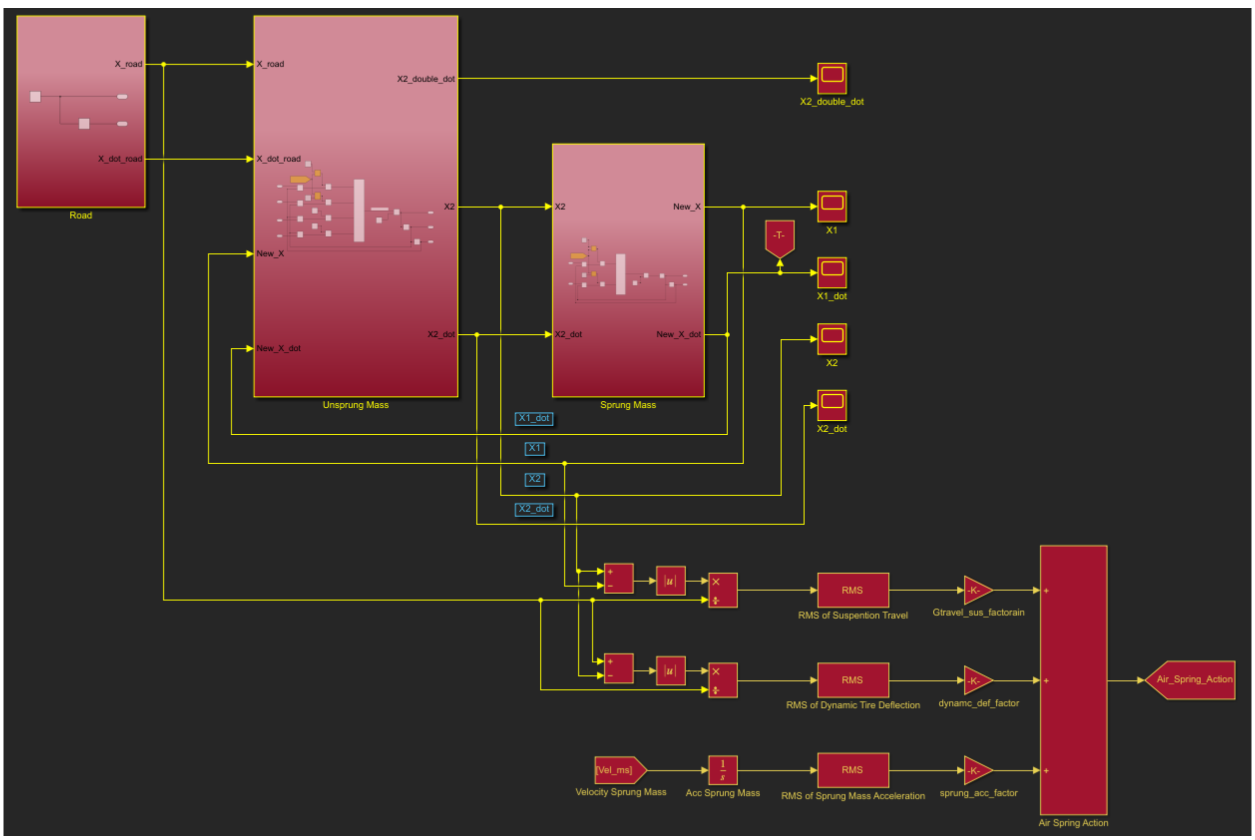


X1
File Tools View Simulation Help
2
1.5
1
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

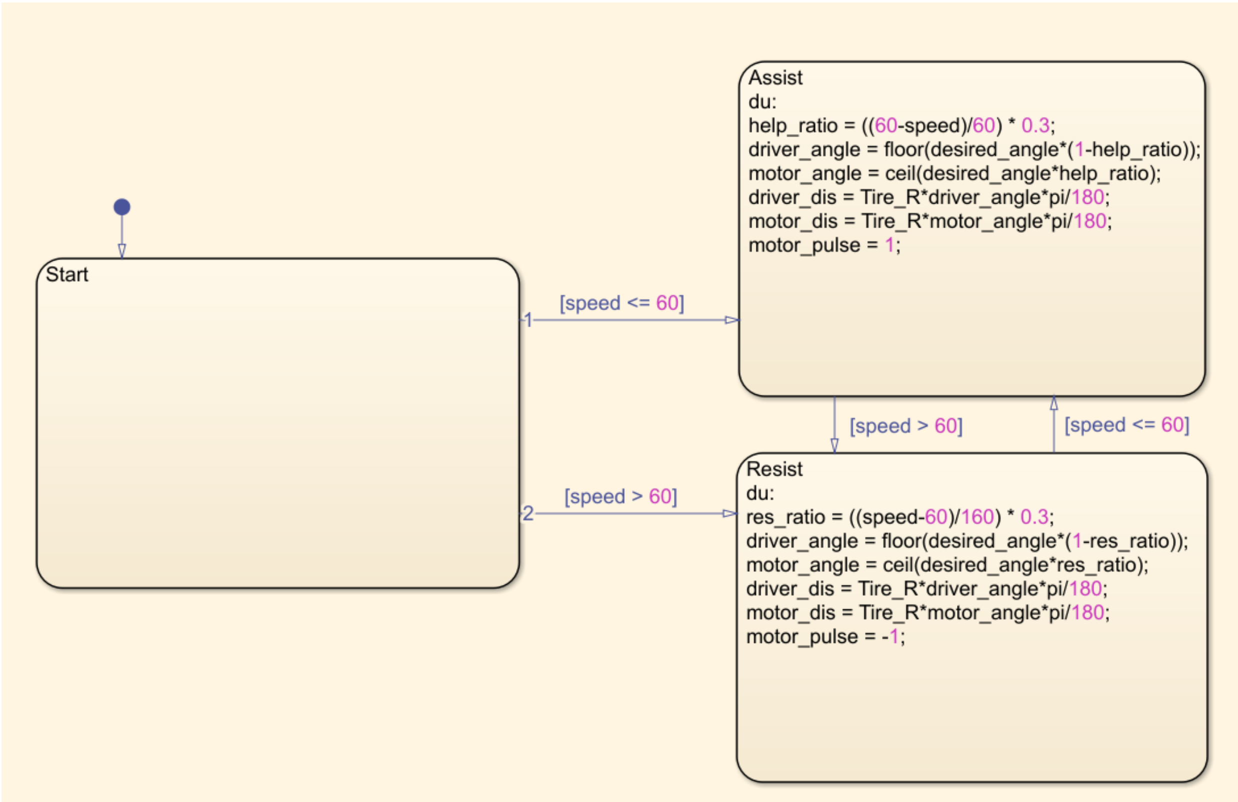


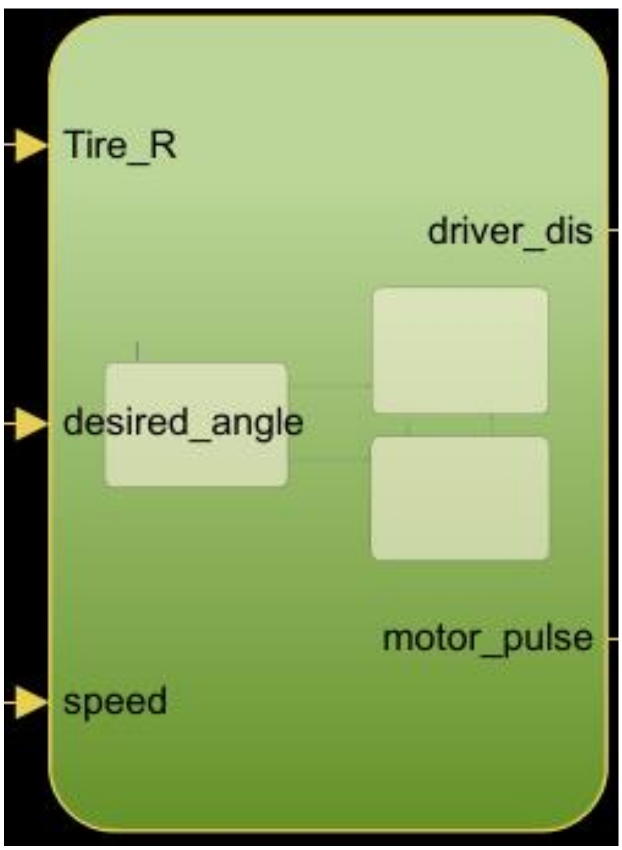
X2
File Tools View Simulation Help
1.2
1
0.8
0.6
0.4
0.2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

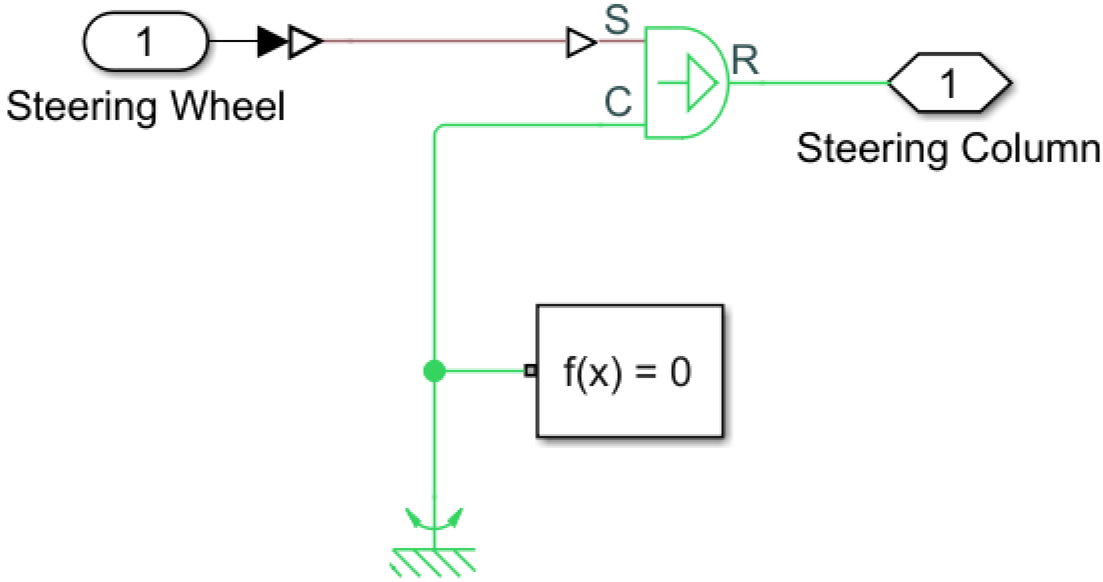

RMS
RMS of Suspention Travel
-K-
Gtravel_sus_factorain
RMS
RMS of Dynamic Tire Deflection
-K-
dynamc_def_factor
Air_Spring_Action
[Vel_ms]
Velocity Sprung Mass
Acc Sprung Mass
RMS
RMS of Sprung Mass Acceleration
-K-
sprung_acc_factor
Air Spring Action

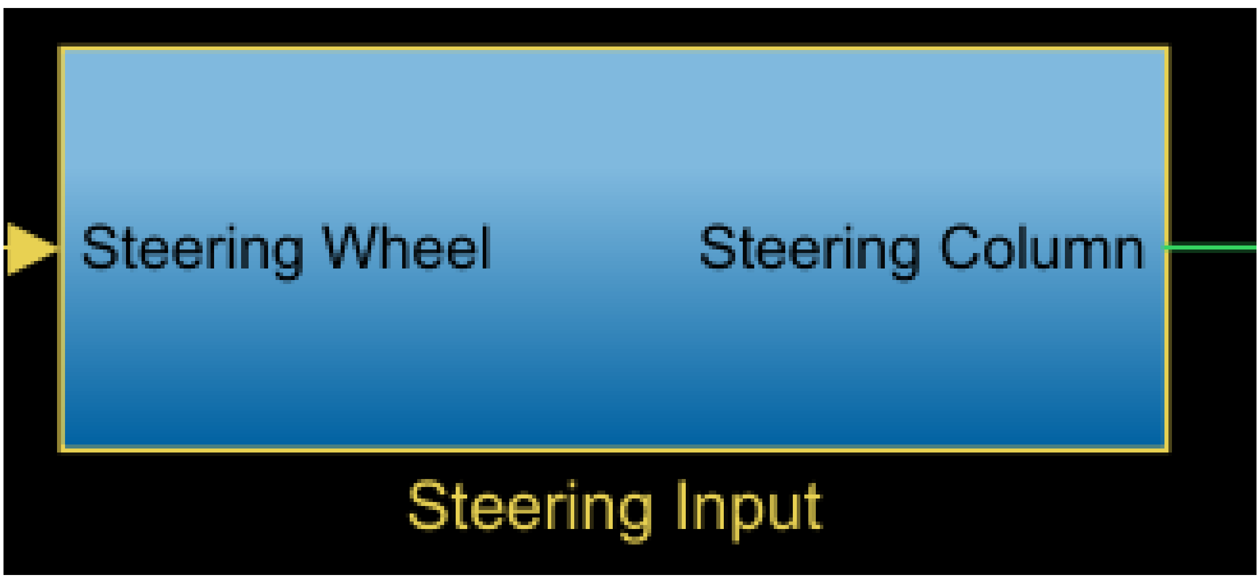

X_road
X_dot_road
Road
X_road
X2_double_dot
X_dot_road
X2
New_X
New_X_dot
X2_dot
Unsprung Mass
X2
New_X
X2_dot
New_X_dot
Sprung Mass
X2_double_dot
X1
-T-
X1_dot
X2
X2_dot
X1_dot
X1
X2
X2_dot
RMS
RMS of Suspention Travel
-K-
Gtravel_sus_factorain
RMS
RMS of Dynamic Tire Deflection
-K-
dynamc_def_factor
Air_Spring_Action
[Vel_ms]
Velocity Sprung Mass
Acc Sprung Mass
RMS
RMS of Sprung Mass Acceleration
-K-
sprung_acc_factor
Air Spring Action

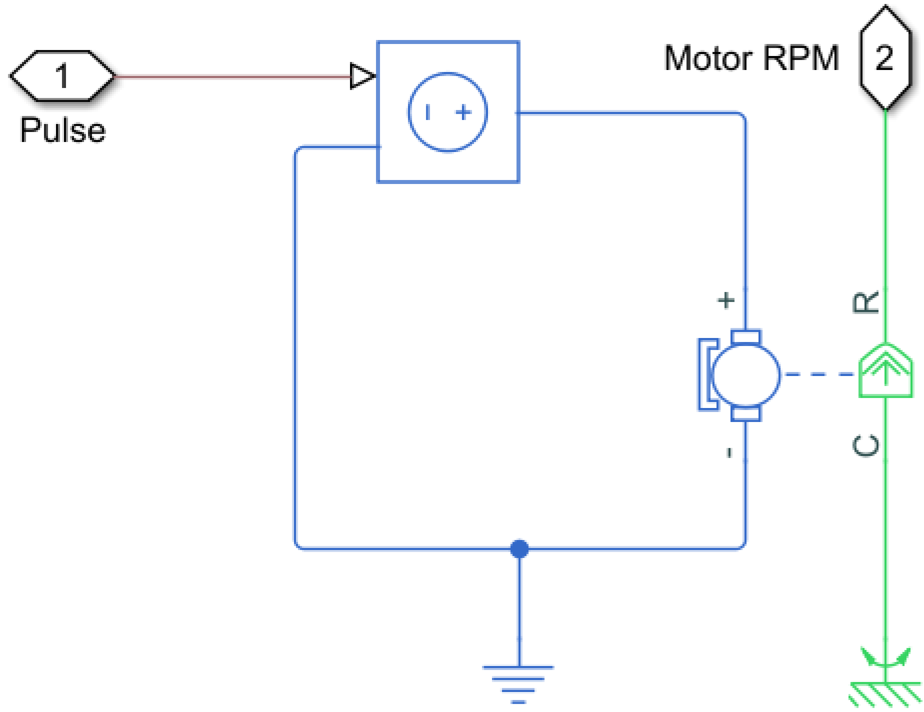
4.0 Active Steering

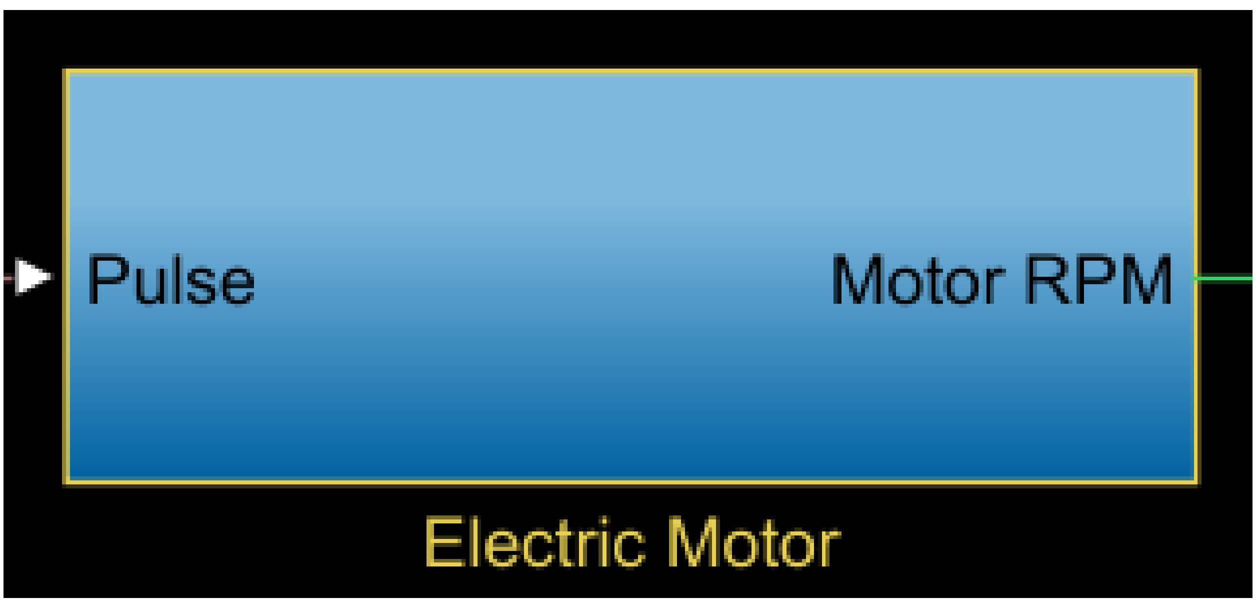


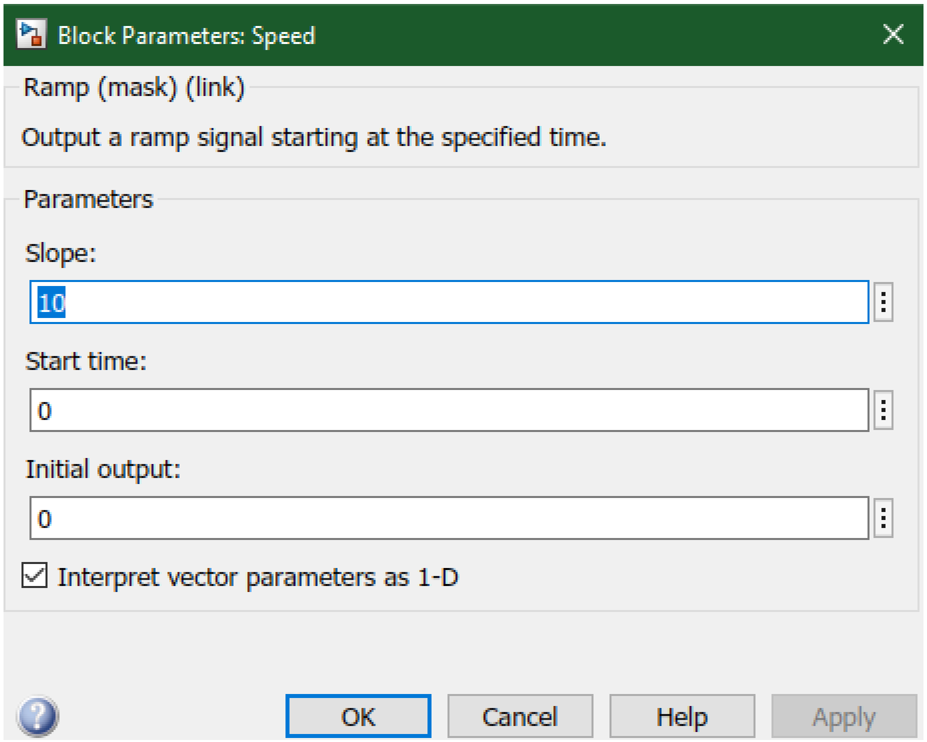


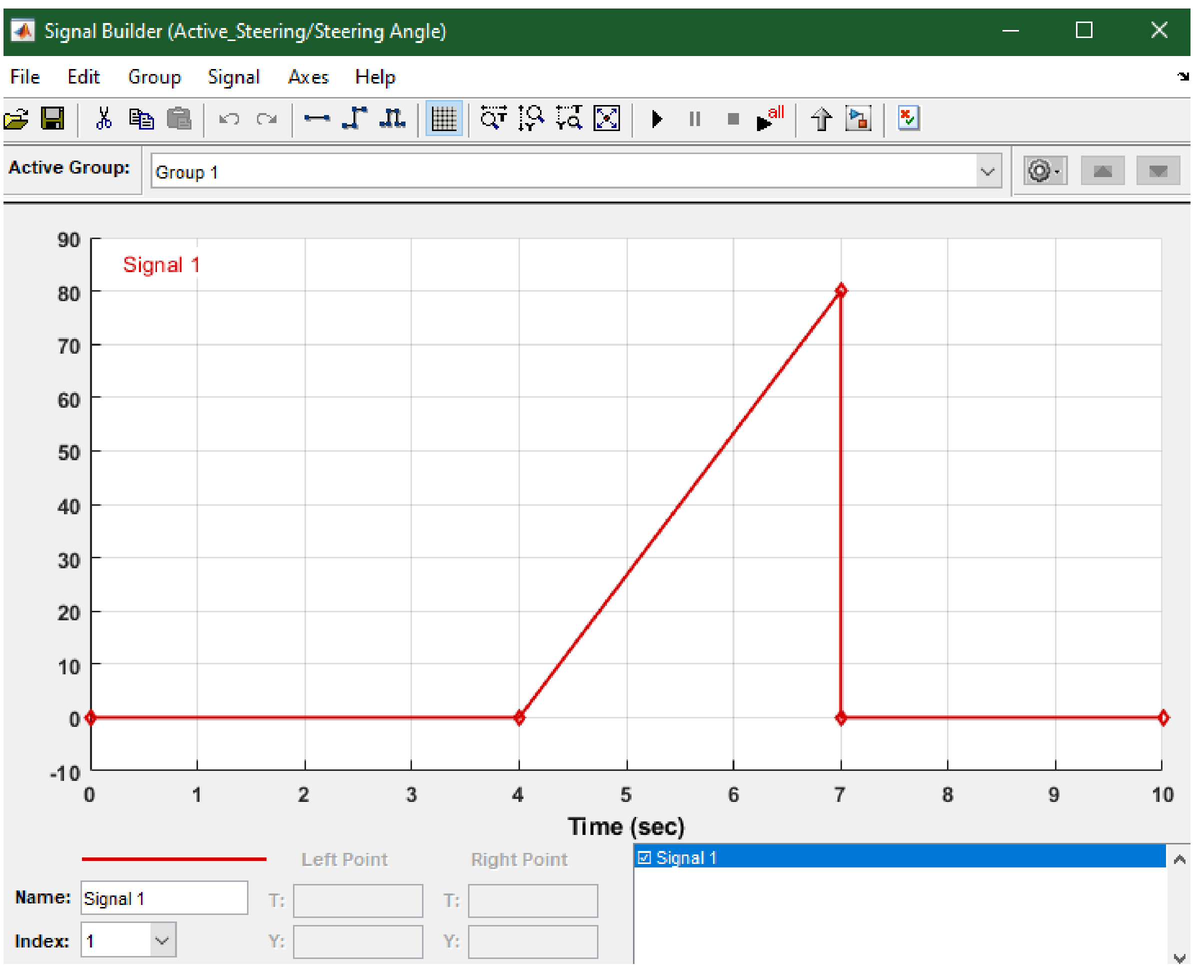

1
Steering Wheel
S
C
R
1
Steering Column
f(x) = 0

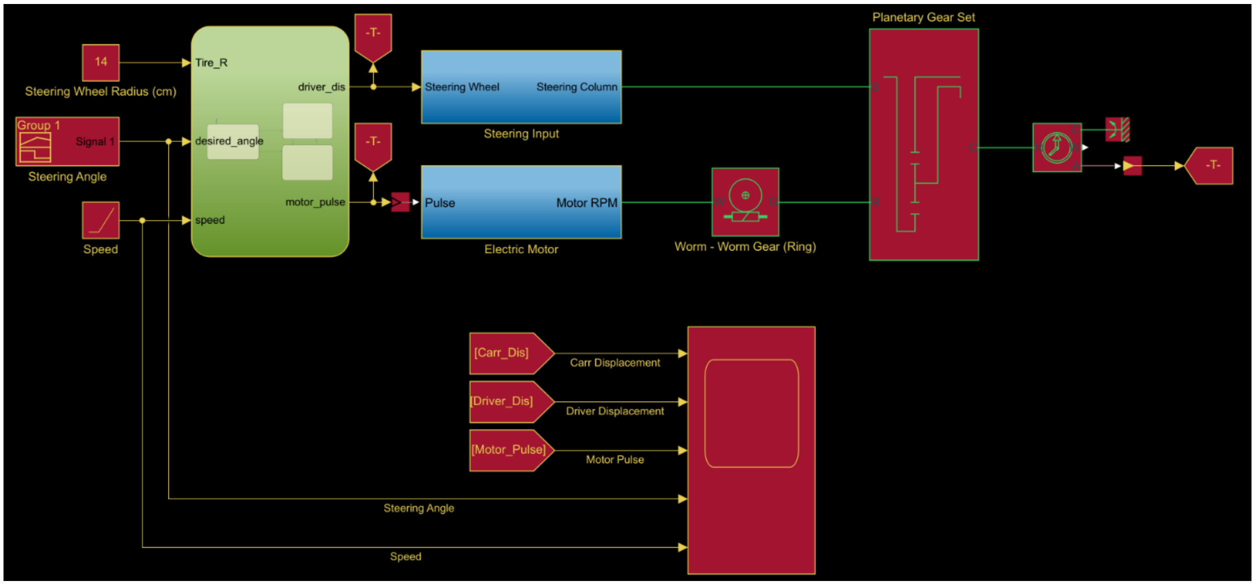

Steering Wheel
Steering Column
Steering Input


1
Pulse
Motor RPM
2
+
-
R
C

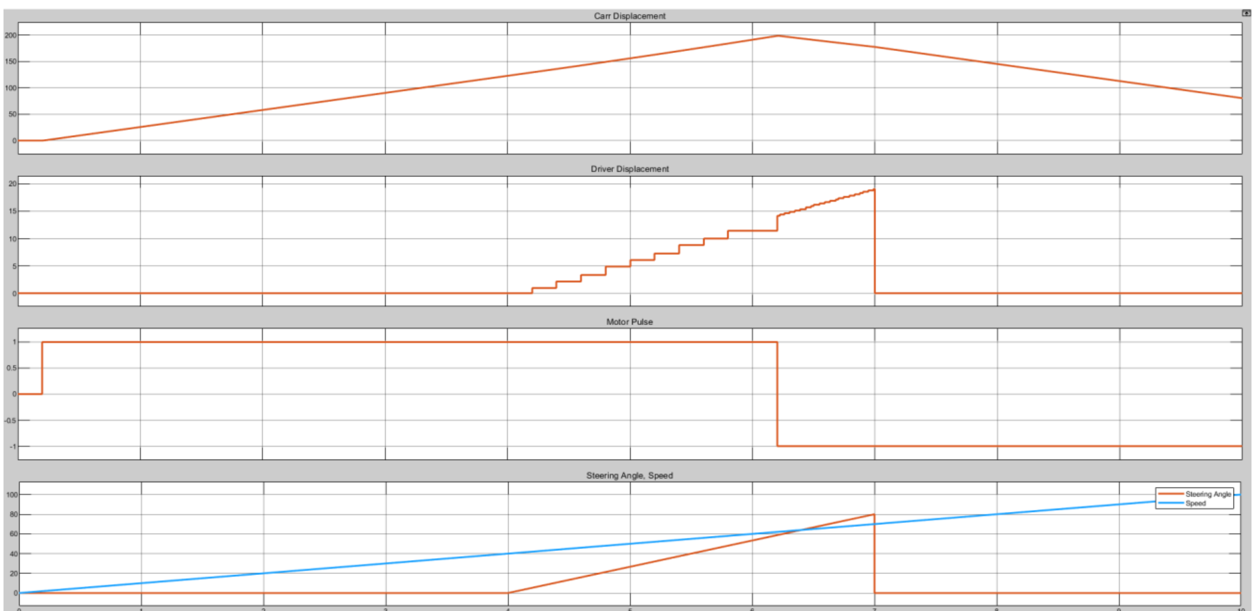

Pulse
Motor RPM
Electric Motor


Block Parameters: Speed
Ramp (mask) (link)
Output a ramp signal starting at the specified time.
Parameters
Slope:
10
Start time:
0
Initial output:
0
Interpret vector parameters as 1-D
OK
Cancel
Help
Apply


Signal Builder (Active_Steering/Steering Angle)
File
Edit
Group
Signal
Axes
Help
Active Group:
Group 1
Signal 1
90
80
70
60
50
40
30
20
10
0
-10
0
1
2
3
4
5
6
7
8
9
10
Time (sec)
Left Point
Right Point
Name:
Signal 1
Index:
1
T:
Y:
T:
Y:
Signal 1



14
Steering Wheel Radius (cm)
Group 1
Signal 1
Steering Angle
Speed
Tire_R
driver_dis
desired_angle
motor_pulse
speed
-T-
-T-
Steering Wheel
Steering Column
Steering Input
Pulse
Motor RPM
Electric Motor
Worm - Worm Gear (Ring)
Planetary Gear Set
-T-
[Carr_Dis]
[Driver_Dis]
[Motor_Pulse]
Carr Displacement
Driver Displacement
Motor Pulse
Steering Angle
Speed



Carr Displacement
Driver Displacement
Motor Pulse
Steering Angle, Speed
Steering Angle
Speed