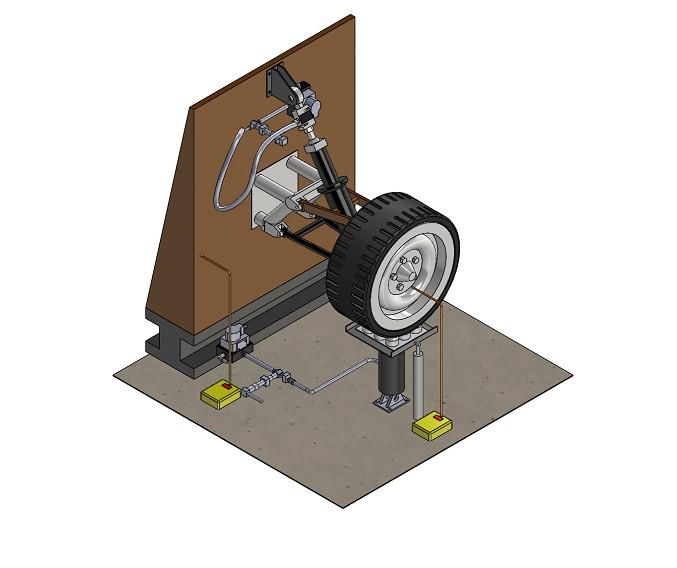
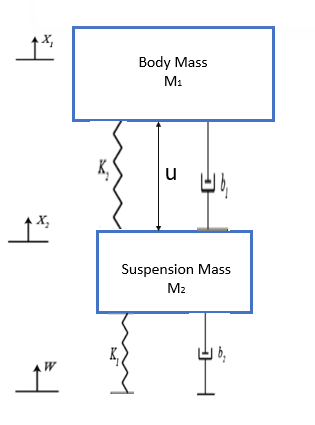
**PID CONTROL SYSTEM FOR MATLAB and SIMULINK**

PHYSICAL SETUP



Designing an automotive suspension system is a fascinating and difficult control challenge. When designing the suspension system, a 1/4 model (one of the four wheels) is employed to reduce the problem to a 1-D multiple spring-damper system. This mechanism is depicted in the diagram below. This model represents an active suspension system with an actuator that can create the control force U to regulate the motion of the automobile body.

MODEL OF THE CAR SUSPENSION SYSTEM



SYSTEM PARAMETERS

(M1) 1/4 car body mass 2500 kg

(M2) suspension mass 320 kg

(K1) spring constant of suspension system 80,000 N/m

(K2) spring constant of wheel and tire 500,000 N/m

(b1) damping constant of suspension system 350 N.s/m

(b2) damping constant of wheel and tire 15,020 N.s/m

(U) control force

EQUATIONS OF MOTION

From the above picture and Newton’s law, we obtain the dynamic equation:

Car Body Bounce:

 (1)

Wheel Bounce:

MX2 = -b1 (X2 – X1) – b2 (W – X2) – K1 (X2 - W) – K2 (X2 – X1) (2)

TRANSFER FUNCTION MODELS

Assume that all of the starting conditions are zero, and that these equations reflect the case in which the vehicle wheel encounters a bump. Using the Laplace Transform, the preceding dynamic equations may be represented as transfer functions. The particular derivation from the aforementioned equations to the transfer functions G1(s) and G2(s) is provided below, with each transfer function having an output of X1-X2 and inputs of U and W.

(M1s2 + b1s + K1) X1(s) – (b1s + K1) X2(s) = U(s) (3)

-(b1s + K1) X1(s) + (M2s2 + (b1 + b2) s + (K1 + K2)) X2(s) = (b2s + K2) W(s) – U(s) (4)

(5)

(6)

(7)

(8)

Finding the inverse of the matrix A and multiplying it with inputs (s) and W(s) on the righthand side:

(9)

(10)

To isolate the control input U(s) only, the W(s) = 0. Then G1(s) becomes:

$$ G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} = \frac{(M_1+M_2) s^2 + b_2 s + K_2}{\Delta} $$ (11)

To isolate the disturbance input W(s) only, the U(s) = 0. Then G2(s) becomes:

$$ G_2(s) = \frac{X_1(s) - X_2(s)}{W(s)} = \frac{-M_1 b_2 s^3 -M_1 K_2 s^2}{\Delta} $$ (12)

GENERATING THE ABOVE FUNCTION IN MATLAB

Enter the following command in the MATLAB command window:

*M1 = 2500;*

*M2 = 320;*

*K1 = 80000;*

*K2 = 500000;*

*b1 = 350;*

*b2 = 15020;*

*s = tf('s');*

*G1 = ((M1+M2)\*s^2+b2\*s+K2)/((M1\*s^2+b1\*s+K1)\*(M2\*s^2+(b1+b2)\*s+(K1+K2))-(b1\*s+K1)\*(b1\*s+K1));*

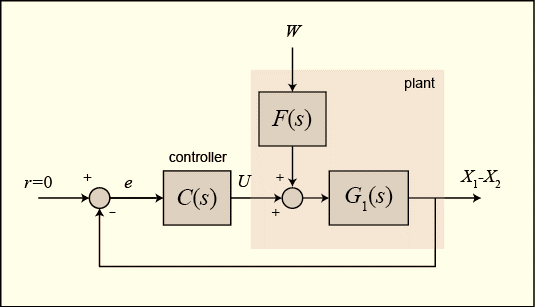
*G2 = (-M1\*b2\*s^3-M1\*K2\*s^2)/((M1\*s^2+b1\*s+K1)\*(M2\*s^2+(b1+b2)\*s+(K1+K2))-(b1\*s+K1)\*(b1\*s+K1));*

**THE PID CONTROLLER**

From the dynamic equations 11 and 12 in transfer function above,

We recall (8),

Where F(s)G1(s) = G2(s), the system schematic becomes:



We intend to create a feedback controller with a settling time of less than 5 seconds and an overshoot of less than 5% when the road disturbance (W) is mimicked by a unit step input. For example, if the automobile is driven onto a 10-cm step, the bus body will oscillate within a 5 mm range and stop oscillating after 5 seconds.

In MATLAB, the system model may be represented by creating a new m-file and inputting the following commands.

*m1 = 2500;*

*m2 = 320;*

*k1 = 80000;*

*k2 = 500000;*

*b1 = 350;*

*b2 = 15020;*

*nump=[(m1+m2) b2 k2];*

*denp=[(m1\*m2) (m1\*(b1+b2))+(m2\*b1) (m1\*(k1+k2))+(m2\*k1)+(b1\*b2) (b1\*k2)+(b2\*k1) k1\*k2];*

*G1=tf(nump,denp);*

*num1=[-(m1\*b2) -(m1\*k2) 0 0];*

*den1=[(m1\*m2) (m1\*(b1+b2))+(m2\*b1) (m1\*(k1+k2))+(m2\*k1)+(b1\*b2) (b1\*k2)+(b2\*k1) k1\*k2];*

*G2=tf(num1,den1);*

*numf=num1;*

*denf=nump;*

*F=tf(numf,denf);*

ADDING THE PID CONTROLLER

Transfer Function formula for a PID Controller is:

(13)

where is the proportional gain, is the integral gain, and is the derivative gain. Assuming the three gains are required for the controller. Assuming values for the gains; K1 = 200300, K2 = 450000, and K3 = 634000. We implement this in MATLAB by adding the code below to the matlab file:

*K1 = 200300;*

*K2 = 450000;*

*K3 = 643000*

*C = pid(K2,K1,K3);*

To model how the system (the distance X1-X2) would react if there were a step disruption on the road. The transfer function from the road disturbance W to the output (X1-X2) may be found in the preceding diagram, and the following can be simulated:

*Sys\_cl = F \* feedback(G1,C);*

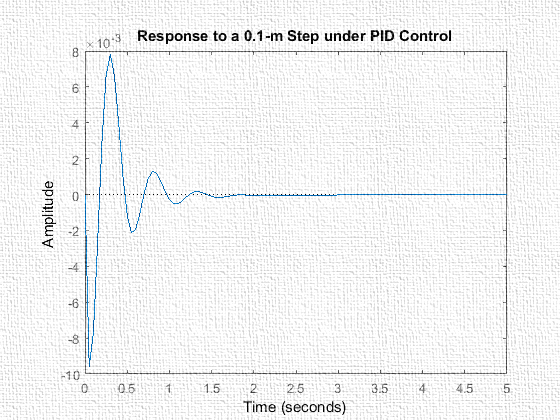
TO PLOT THE CLOSED LOOP RESPONSE

The closed loop transfer function has been created to represent the automobile, disturbance and the controller. The *sys\_cl has to be multiplied 0.1-m in the step block as the disturbance, as this line of code:*

t=0:0.05:5;

step(0.1\*sys\_cl,t)

running the simulation of the scope block gives:



To generate a root locus with the PID controller, we add the following command in the matlab file to understand what gain to use for K1, K2, and K3.

z1=1;

z2=3;

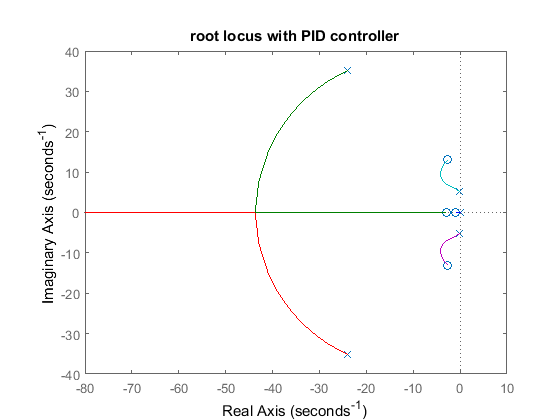
p1=0;

s = tf('s');

C = ((s+z1)\*(s+z2))/(s+p1);

rlocus(C\*G1)

which will yield graph:



CHOOSING GAINS FOR THE PID CONTROLLER

With the closed-loop transfer function in hand, all that's left to do is fine-tune the K1, K2, and K3 gains to manage the system. The system has more damping than is necessarybut the settling period is rather quick. The reaction may be fixed to get a better response by modifying the K1, K2, and K3 by a factor of 2. Rerunning the program gives the following figure.

K3=2\*K3;

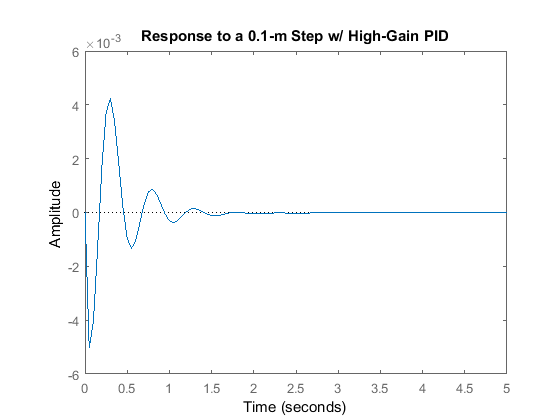
K2=2\*K2;

K1=2\*K1;

C=pid(K2,K1,K3);

sys\_cl=F\*feedback(G1,C);

step(0.1\*sys\_cl,t)



Summarily, the root locus plot evidently proofs that the PID design is able to control the system. Tweaking the three gains will produce varying overshoot and settling time.**SIMULINK**

Design Requirements

A decent automobile suspension system should provide adequate road holding while being comfortable when going over bumps and holes in the road. When the vehicle encounters a road disturbance (such as potholes, fissures, or uneven pavement), the vehicle body should not experience substantial oscillations, and the oscillations should fade soon. Because the distance X1-W is difficult to measure and the deformation of the tire (X2-W) is minimal, we will utilize the distance X1-X2 as the output in our issue rather than X1-W. Remember that this is an estimate.

A step input will be used to mimic the road disturbance (W) in this case. This step might depict a car exiting a pothole.

Building the Model

To get velocities and locations, the forces acting on both masses (body and suspension) will be added and the accelerations of each mass will be integrated twice. Each mass will be subjected to Newton's law. Simulink will be launched and a new model window will be created. First, we shall compute the integrals of the masses' accelerations.

 (1)

 (2)

Processes taken involves:

* Insert an Integrator block (from the Continuous library) and draw lines to and from its input and output terminals.
* Label the input line "Xdot\_dot" (for acceleration) and the output line "Xdot" (for velocity) To add such a label, double click in the empty space just above the line.
* Insert another Integrator block and connect it to the output of the first.
* Draw a line from its output and label it "X" (for position).

We will model Newton’s Law for the masses which will be expressed as:

 (3)

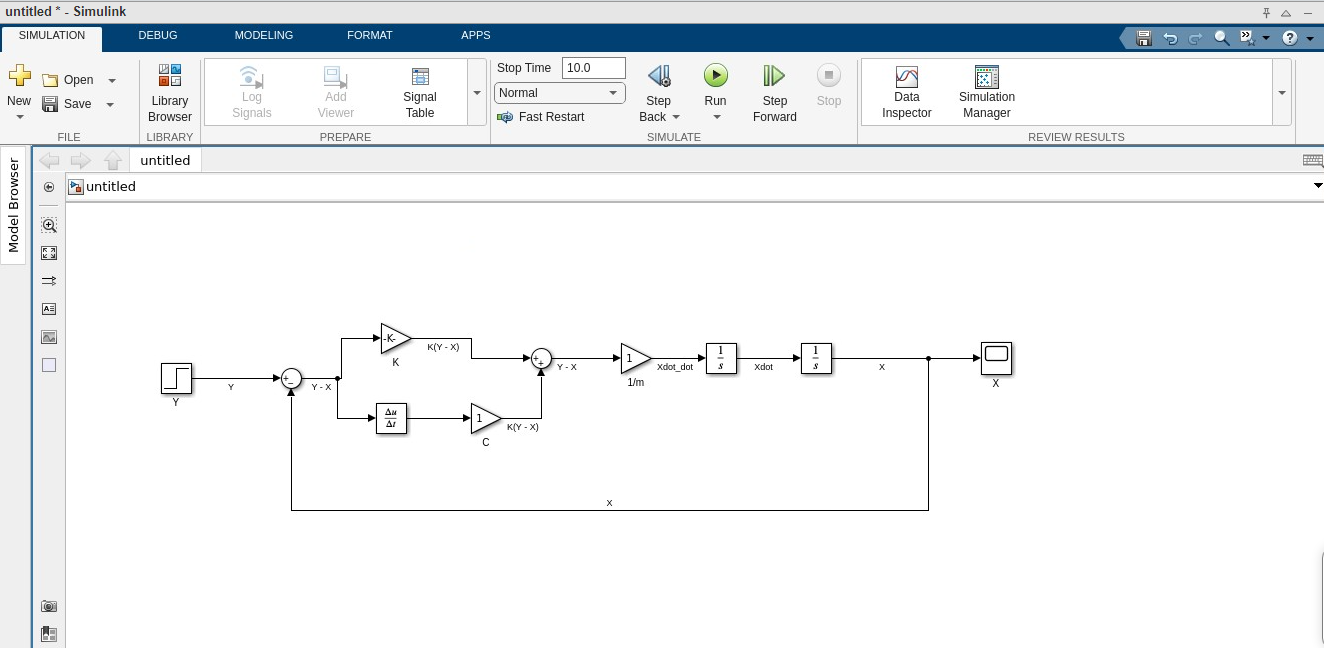
* Insert three Gain blocks, (from the Math Operations library) one attached to the inputs of each of the integrator pairs.
* Edit the third gain block corresponding to M1 by double-clicking it and changing its value to "1/500" and labeled to “1/m”.
* Change the label of first Gain block to "K" by clicking on the word "Gain" underneath the block and the value to “25,000”.
* Similarly, edit the label of the second Gain block to "C" and its value to “1,000”. (You may want to resize the gain blocks to view the contents. To do this, single click on the block to highlight it, and drag one of the corners to the desired size.)
* Add a Step block in the input of the model window, label it “Y”.
* Edit its Step Time to “1” and its Final Value to “1” and its Initial Value to “0” while enabling the zero-crossing detection.
* Insert an Add block to the right of Y Step block and edit its signs to “+-”
* Insert another Add block at the output of the two Gains and edit its signs to “++”

Since there is no existing signal representing the derivative of Y we will need to generate this signal.

* Insert a Derivative block (from the Continuous library) to the right of the Y step block.
* Tap a line of the Step's output and connect it to the input of the Derivative block.
* Connect the output of the derivative block to the Gains labeled C.

To view the output X

* Insert a Scope block (from the Sinks library) and connect it to the output of the rightmost Integrator block.



OPEN-LOOP RESPONSE

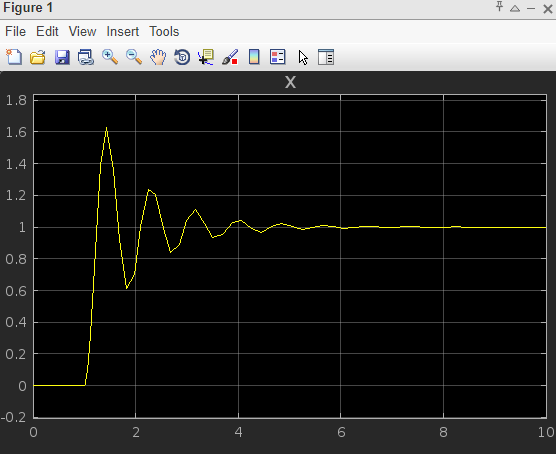
To simulate this system, first determine an acceptable simulation time. Set the time in the Stop Time box by selecting Model Configuration Parameters from the Simulation menu to a Scope Simulation Time of 10.000s. Physical parameters must now be defined. At the MATLAB prompt, enter the following commands:

Gains 1 = 25, 000

Gains 2 = 1, 000

Gains 3 = 1/500

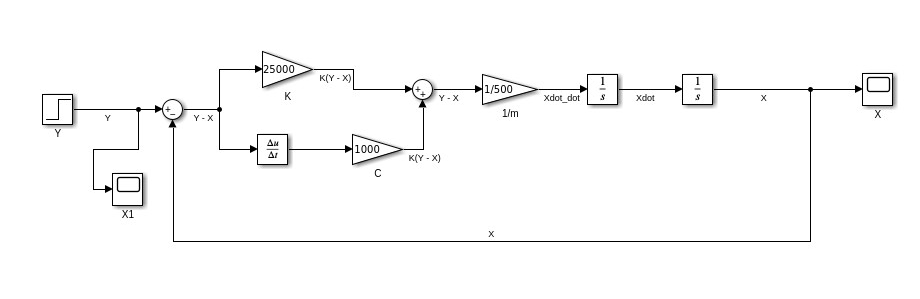
We run the simulation (**Ctrl-T** or **Run** form the Simulation view). Double-click on the Scope block and the following output comes out:



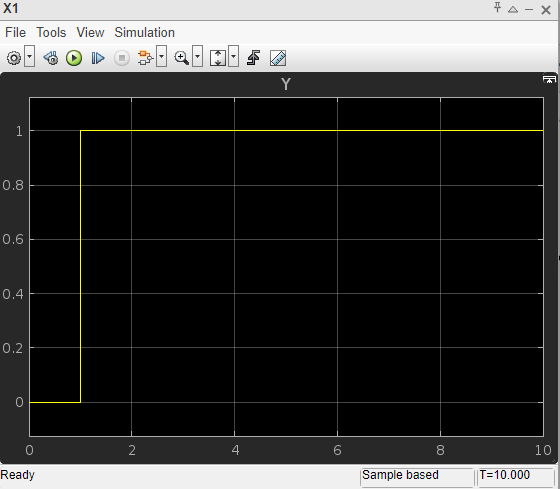
STEP-RESPONSE

To check the simulink simulation of the step response:

* Insert a Scope block from the Library Browser, attach it to the output of Step block Y as seen in the figure below, and label it as “X1”

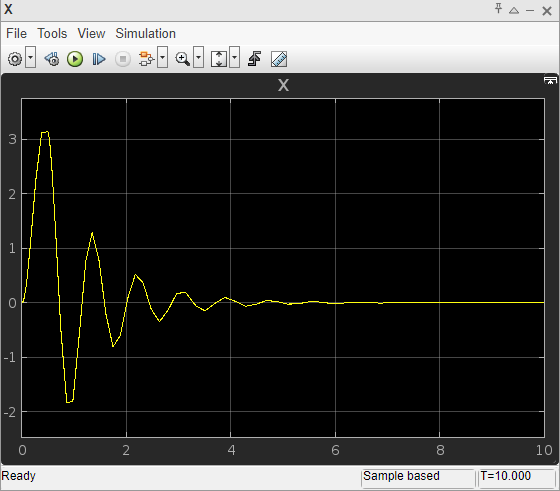


* Rerun the simulation to get the step response figure.



EXPERIMENTING WITH A STEP RESPONSE OF 0.5

* Double click on the Step block, change the Step time to “0.5”, Initial value to “2”
* Rerun the Scope block, we get:



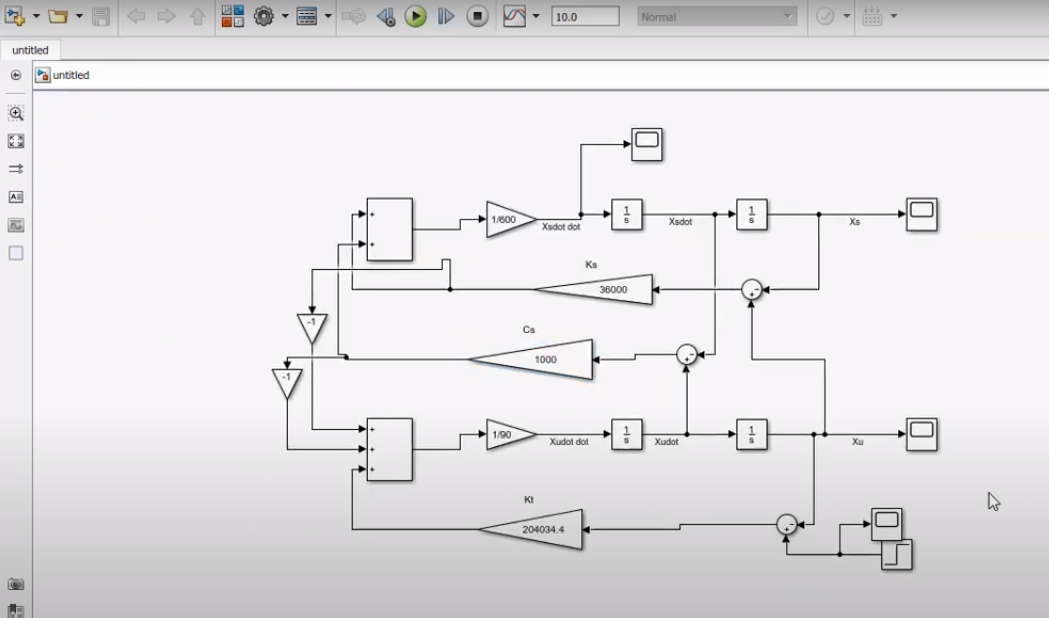
CONTROLLER DESIGN

* Insert a Step block
* Insert a Scope block
* Insert two integrator blocks, the gains should be connected to the first integrator and labeled ‘Xdotdot’
* Insert a Add block from “Math Operations” in the Continous library

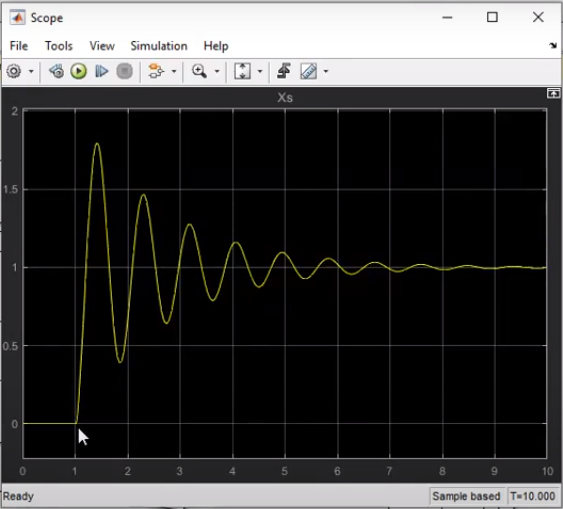
The block order is

Add -> Gain -> Integrator -> Integrator -> Scope

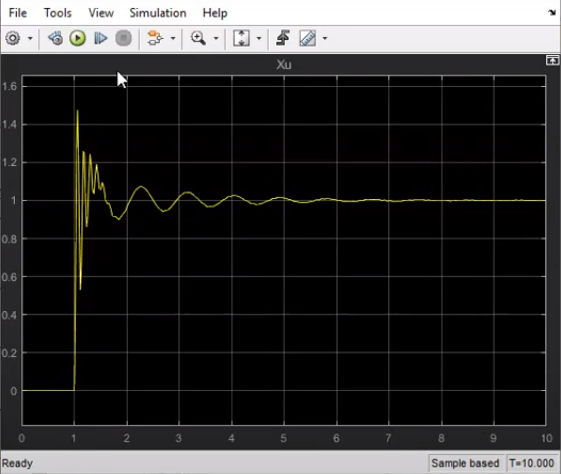
* Create a copy of the block order
* Change the second Add block to ‘+++’
* Insert a Sum block, change its value to ‘-+’
* The second integrator in both orders will be connected to the Scope block and labeled ‘Xa’
* Create two gain blocks with a value of ‘-1’ and connect them to the Add block
* Change the value of first gain to ‘1/600’ and the second to ‘1/90’
* Insert a gain block labeled ‘Kt’ for the tire stiffness of a value of ‘204034.4’
* The damping coefficient gain block is to be labeled as ‘Cs’ with a value of ‘1000’
* The spring stiffness gain block is to be labeled as ‘Ka’ with a value of ‘36000’



We run the simulation (**Ctrl-T** or **Run** form the Simulation view). Double-click on the Scope block and the following output comes out:



Scope block 2 generates:



Scope block 3 generate the acceleration simulation:

