Model-Based Software Design

Laboratory 3 Report

Components of the working group (max 2 people)

* Maria Di Gregorio 302407
* Elenora Tomassetti 305208

The aim of this laboratory is to design a quarter-car physical model, a skyhook control system and error checker, which gives an error each time the input signals are stuck to any value for more than 20 ms and it is able to enter a safe state when it is triggered.

# Plant model

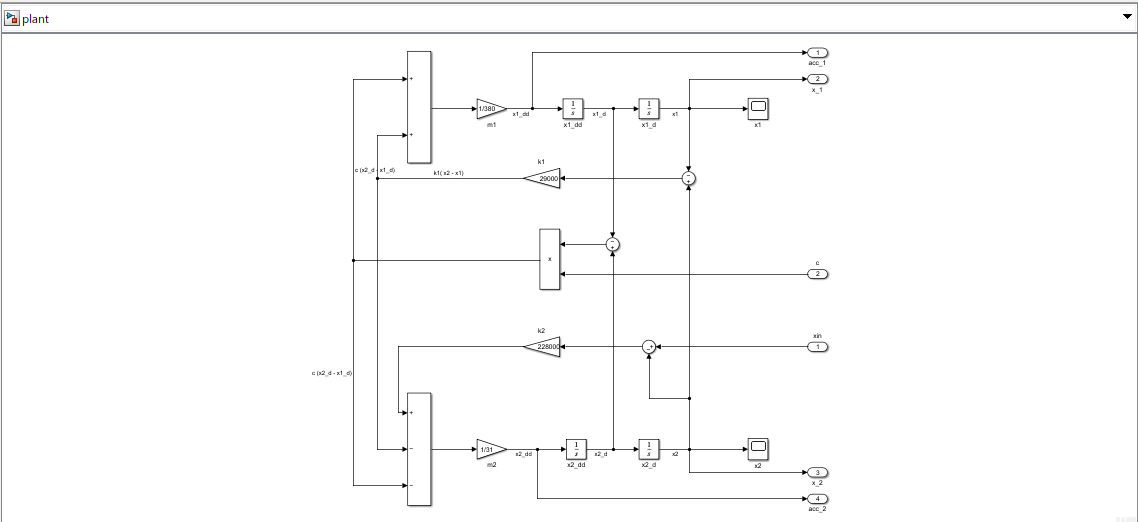


Figure 1: Plant

In order to create the plant, we implemented the following equations:

Therefore, we picked the following values to give to the parameters:

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Car body mass () | 380 Kg |
| Wheel body mass () | 31 Kg |
| Suspension stiffness () | 29.000 N/m |
| Wheel stiffness () | 228.000 N/m |

One of the most important interfaces of the plant is , which represents the road displacement. In order to simulate the road profile, we build as a sinusoidal signal, between -0.2 and 0.2, summed to a white guassian noise.

## Interfaces

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Unit\*** | **Type[[1]](#footnote-1)[[2]](#footnote-2)** | **Data Type[[3]](#footnote-3)** | **Dimension** | **Min** | **Max** |
| Damping control | N.A. | Input | Boolean | 1x1 | N.A. | N.A. |
| Road displacement | M | Input | Double | 1x1 | -0.2 | 0.2 |
| Vehicle acceleration | m/s2 | Output | Double | 1x1 |  |  |
| Wheel acceleration | m/s2 | Output | Double | 1x1 |  |  |
| Vehicle displacement | m | Output | Double | 1x1 |  |  |
| Wheel displacement | m | Output | Double | 1x1 |  |  |

# Controller implementation

Immagine che contiene piazza

Descrizione generata automaticamente

Figure 2:Controller's overview

In order to build the controller, we create two states in parallel:

* One containing the skyhook control system.
* Another one containing the error checker.

Skyhook control system

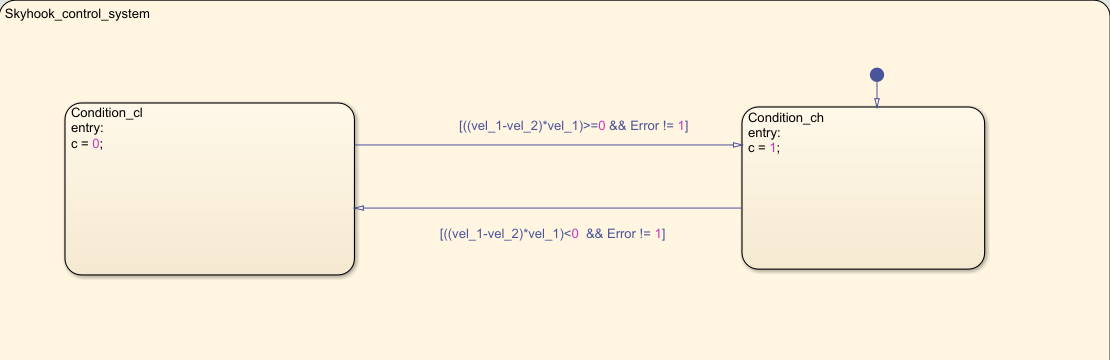


Figure 3: Skyhook control system

In the first state, we implemented the following inequalities:

As it is clear, in order to make the skyhook controller work properly, we need two important inputs which are the two velocities, .

The velocities are obtained by integrating the two accelerations taken by the outputs of the plant.

Since the damping control is a Boolean function, it can only assume two values: 0 and 1.

Thus, we assign to the value 0 and to the value 1.

Therefore, we created two states:

* The first one containing c set to 0, which represents c = 1500
* The second one containing c set to 1, which represents c = 6000 .

We can move form one states to the other through the transition condition, in particular :

* The first state is entered when:
* The second state is entered when:

In both case we need the *Error* set to 0, because otherwise there is a malfunctioning of the sensors and we need to enter the safe state so we need to maintain c set to 6000.

Finally, in order to actually express c equal to 1500 or 6000 , we putted a switch connected to a display on the controller output, using 0.5 as a threshold.

Error checker

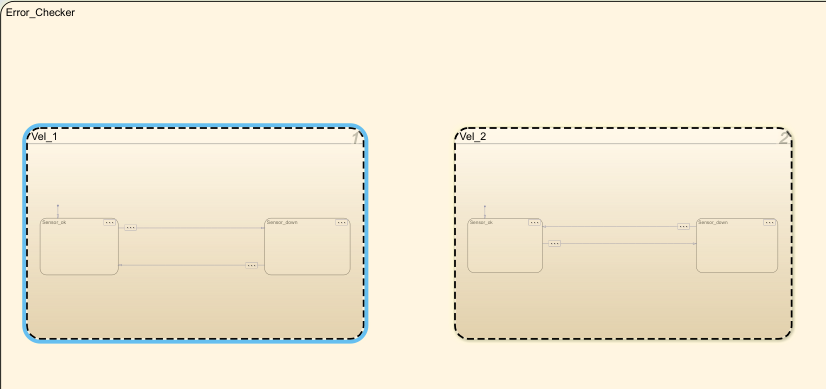


Figure 4:Error checker

In this state, we need to verify if the controller inputs, in particular remain stuck to any value for more than 20ms and if so, the controller shall enter the safe state (c = 6000). Therefore, we build two parallel states:

* The first one that checks
* The second one that checks .

The two states work exactly in the same way:

Immagine che contiene testo

Descrizione generata automaticamente

Figure 5: check of vel\_1

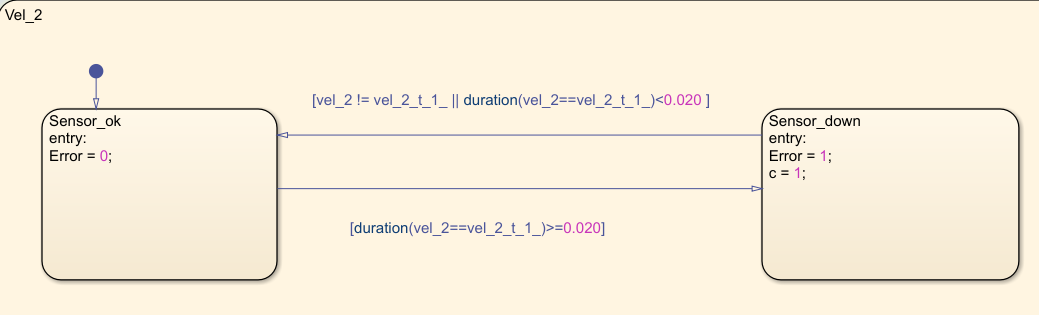


Figure 6: check of vel\_2

In order to see if a variable is equal to the same value for a certain amount of time, we compere each input with its previous value, using an *unit delay block*, then we used the temporal logic operator “*duration*”, to express the amount of time in which the two data were the same.

Thus, we build two states:

* In the first one, everything is working properly, so we set *Error* = 0.
* In the second one, we have a malfunction of the sensors, so we set *Error* = 1 and we also need to enter the safe state, so we set c = 1.

In order to enter the first state we need to satisfy the following conditions:

As anticipated before, we enter the first state if the two values of the velocities are different or if they are equal for an amount of time less than 20ms.

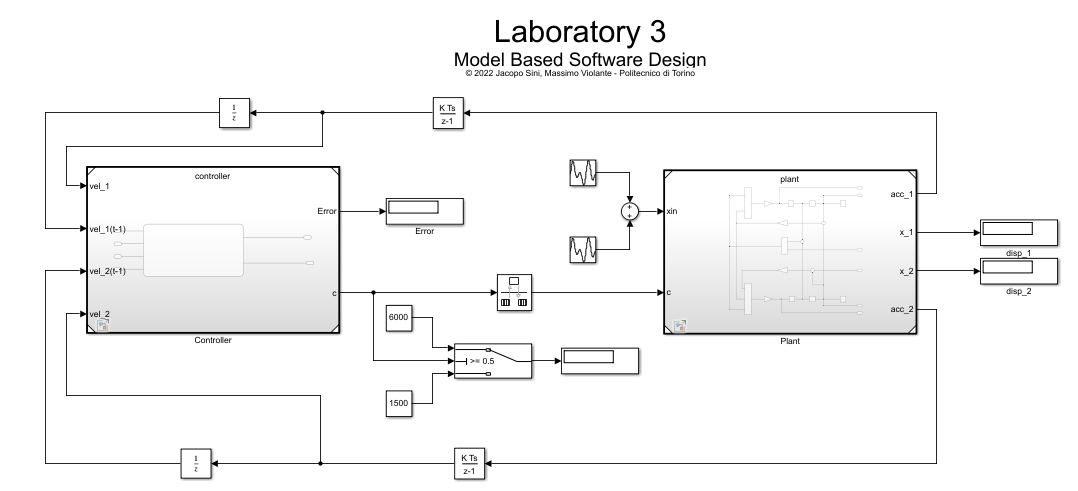
In order to enter the second state we need to satisfy the following conditions:

As anticipated before, we enter the second state if the two values of the velocities are equal for an amount of time more than 20ms.

## Interfaces

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Unit\*** | **Type[[4]](#footnote-4)** | **Data Type[[5]](#footnote-5)** | **Dimension** | **Min** | **Max** |
| Vel\_1 | m/s | Input | Double | 1x1 |  |  |
| Vel\_2 | m/s | Input | Double | 1x1 |  |  |
| Vel\_1(t-1) | m/s | Input | Double | 1x1 |  |  |
| Vel\_2(t-1) | m/s | Input | Double | 1x1 |  |  |
| Error | N.A. | Output | Boolean | 1x1 |  |  |
| c | N.A. | Output | Boolean | 1x1 |  |  |

# Harness

**

## Test stimuli

In order to check the controller functionality, we created two simulations:

* One was aimed to check if the skyhook controller system was working properly.
* The second one was aimed to check if the error checker was working properly.

## Error checker test

Since the error checker needs to verify if the controller inputs are stuck to any value for more than 20ms, we build three simulations:

* In the first one both *vel\_1* and *vel\_2* were set to constant values. As a result, we obtained what we were expecting, so *Error* = 1 and c = 6000.
* In the second one just *vel\_1* was set to a constant value. Also in this case, we obtained what we were expecting, so *Error* = 1 and c = 6000.
* In the last one just *vel\_2* was set to a constant value. As a result, we obtained what we were expecting, so *Error* = 1 and c = 6000.

## Skyhook test

In order to test the skyhook control system, we build two simulations, in both of them vel\_1 and vel\_2 were created as two square waves synchronized:

* In the first one, we wanted to obtain c = c\_l. Thus we gave to vel\_1 an amplitude of 2 and to vel\_2 an amplitude of 3, since they had the same dutycycle and the same frequency (f = 100Hz, so that it wouldn’t trigger the error and the safe state), we obtained this two pair of values:
* Vel\_1 = 2 and vel\_2 = 3 -> c = c\_l.
* Vel\_1 = -2 and vel\_2 = -3 -> c = c\_l.
* In the second one, we wanted to obtain c = c\_h. Thus we gave to vel\_1 an amplitude of 3 and to vel\_2 an amplitude of 2, since they had the same dutycycle and the same frequency (f = 100Hz, so that it wouldn’t trigger the error and the safe state), we obtained this two pair of values:
* = 3 and vel\_2 = 2 c .
* = -3 and vel\_2 = -2 -> c = c\_h.

The results that we obtain met our expectation.

## Performances results

Once we made sure that everything was working properly, we selected the following stimuli set and we comparedthe performances in terms of vertical accelerations with the controller enabled or disabled:

* Sinusoidal signal (Amplitude = 0.2, frequency = 1 rad/s, phase = 0)

When the controller is enabled, we obtained the following result for acc\_1 and acc\_2:

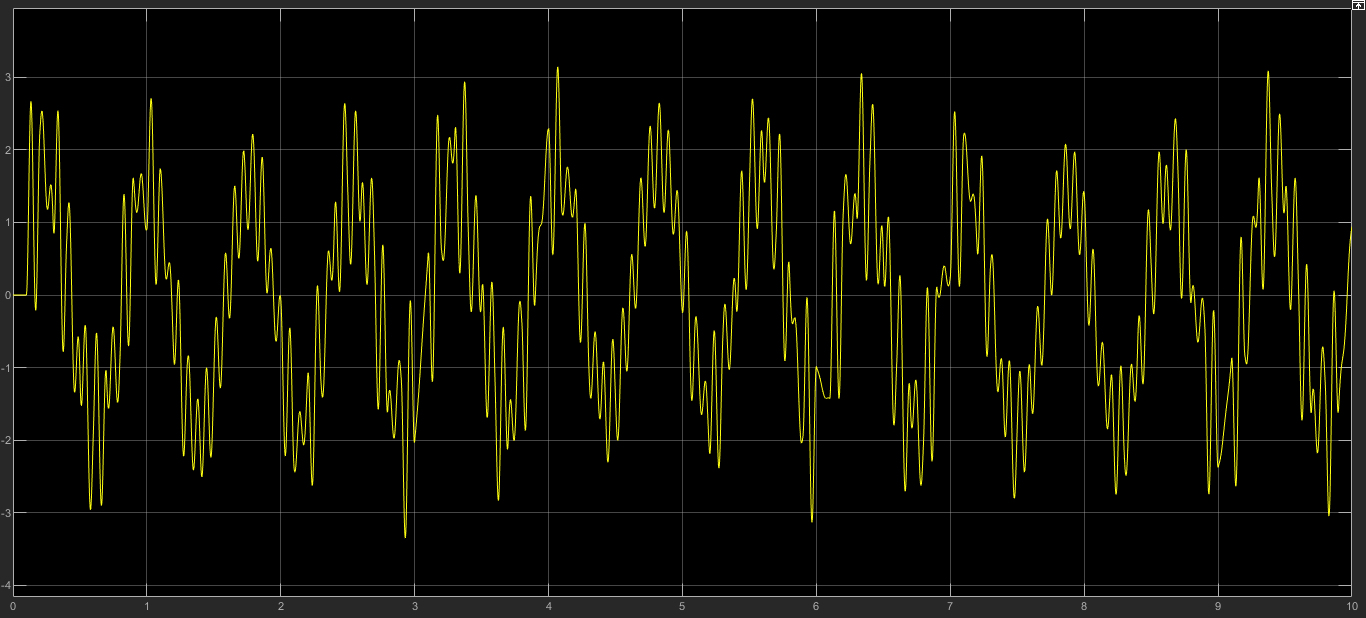


Figure 7: acc\_1

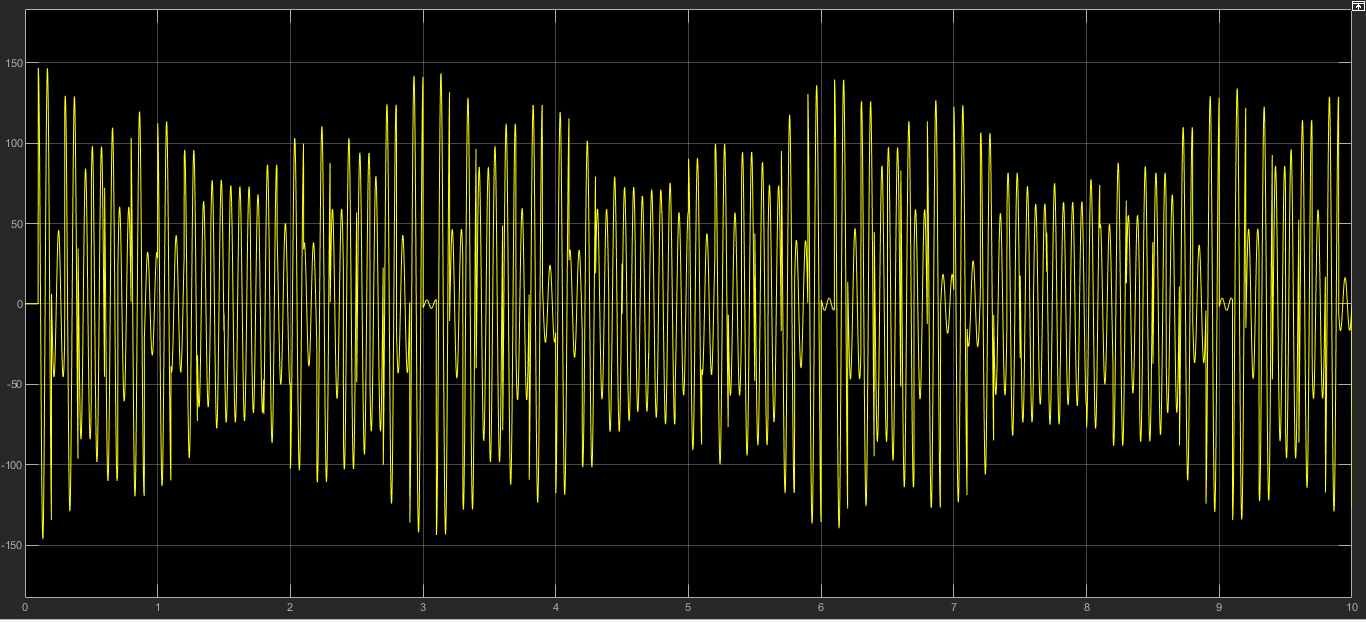


Figure 8: acc\_2

When the controller is disabled and c fixed to 6000 , we obtained the following result for acc\_1 and acc\_2:

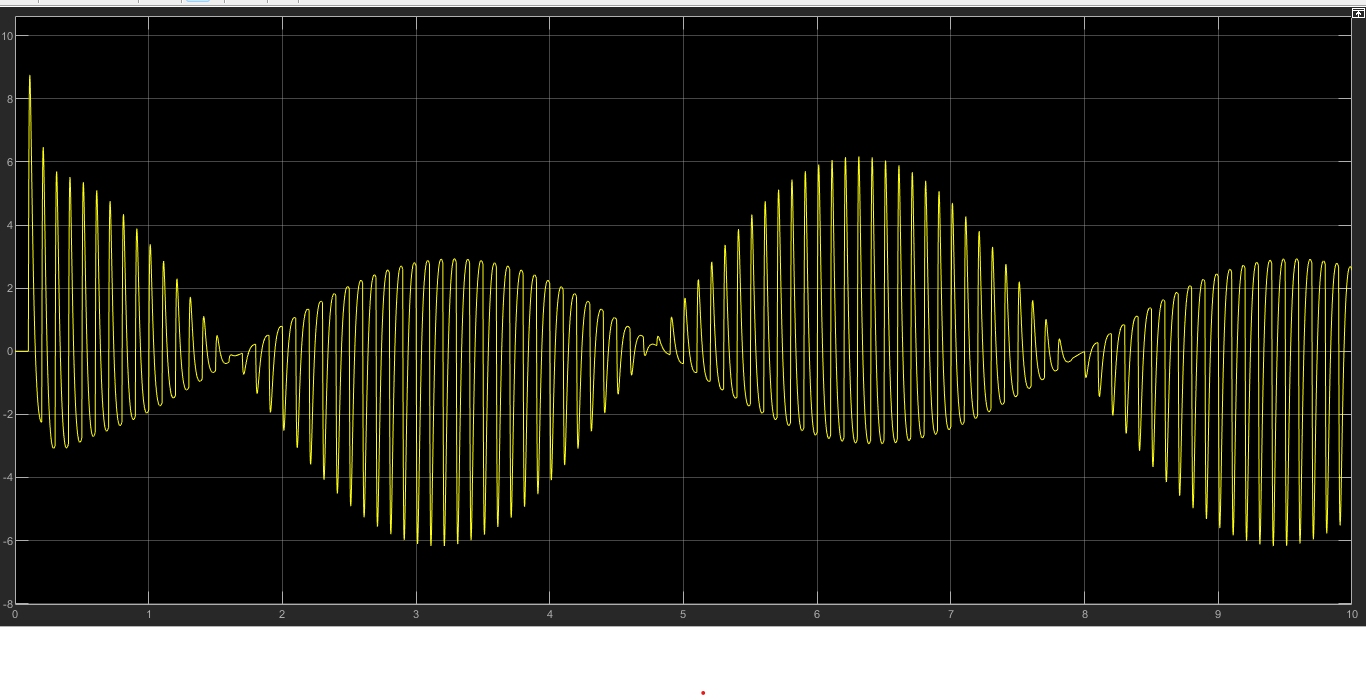


Figure 9:acc\_1

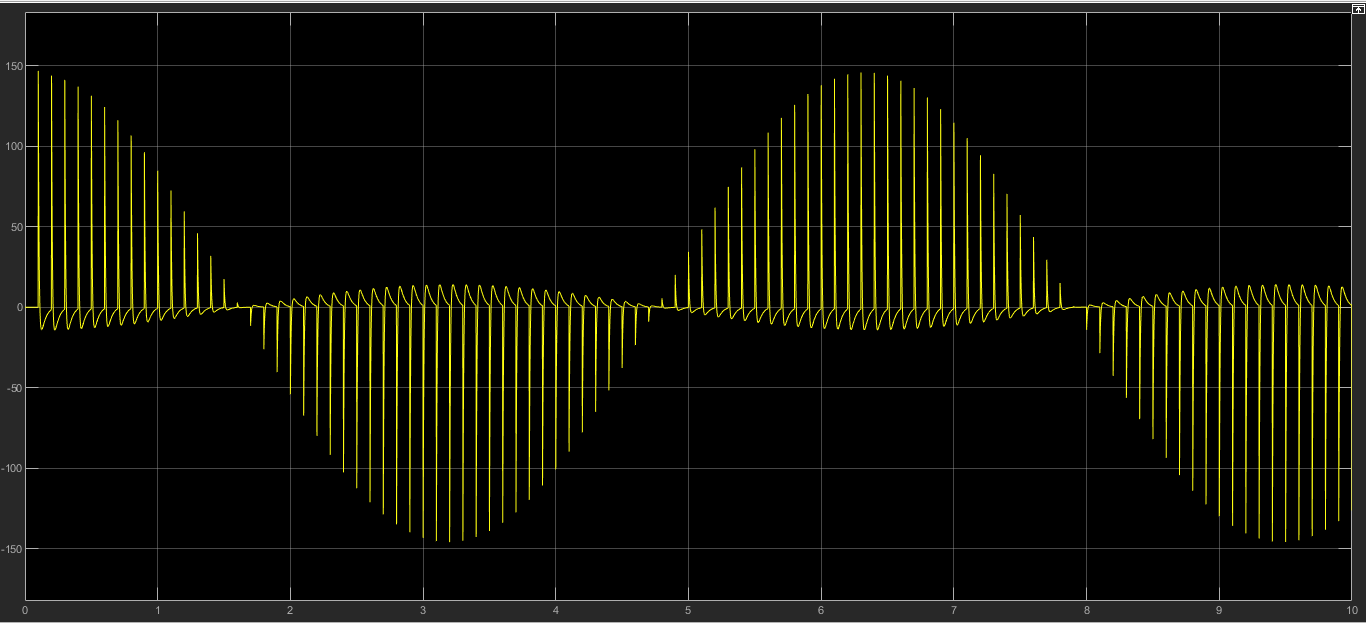


Figure 10: acc\_2

* Sinusoidal signal + white gaussian noise. (mean = 0, variance =0.001, seed = 42)

When the controller is enabled, we obtained the following result for acc\_1 and acc\_2:

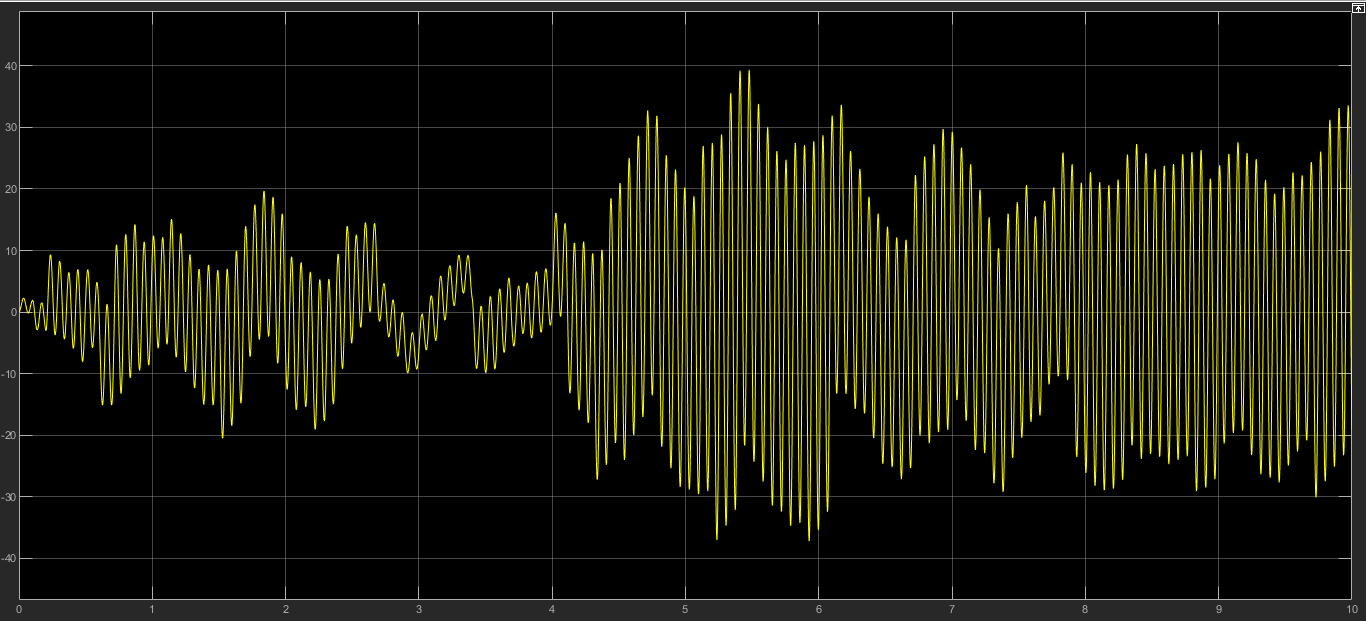


Figure 11: acc\_1

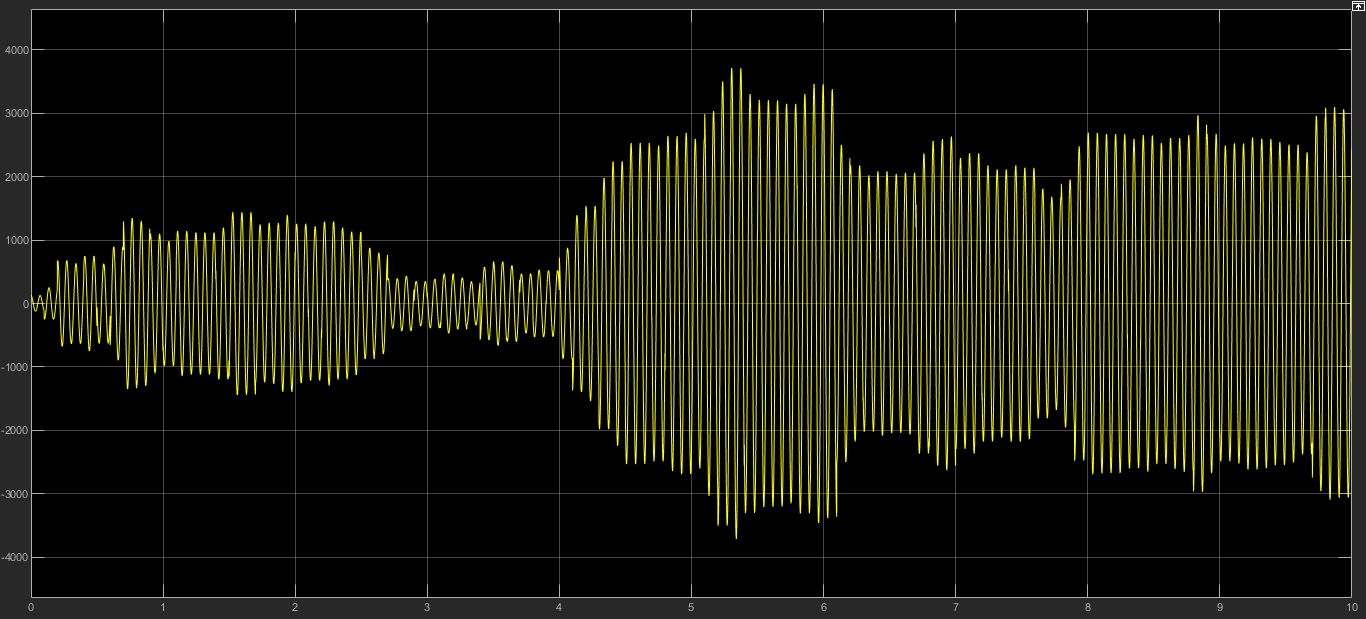


Figure 12: acc\_2

When the controller is disabled and c fixed to 6000 , we obtained the following result for acc\_1 and acc\_2:

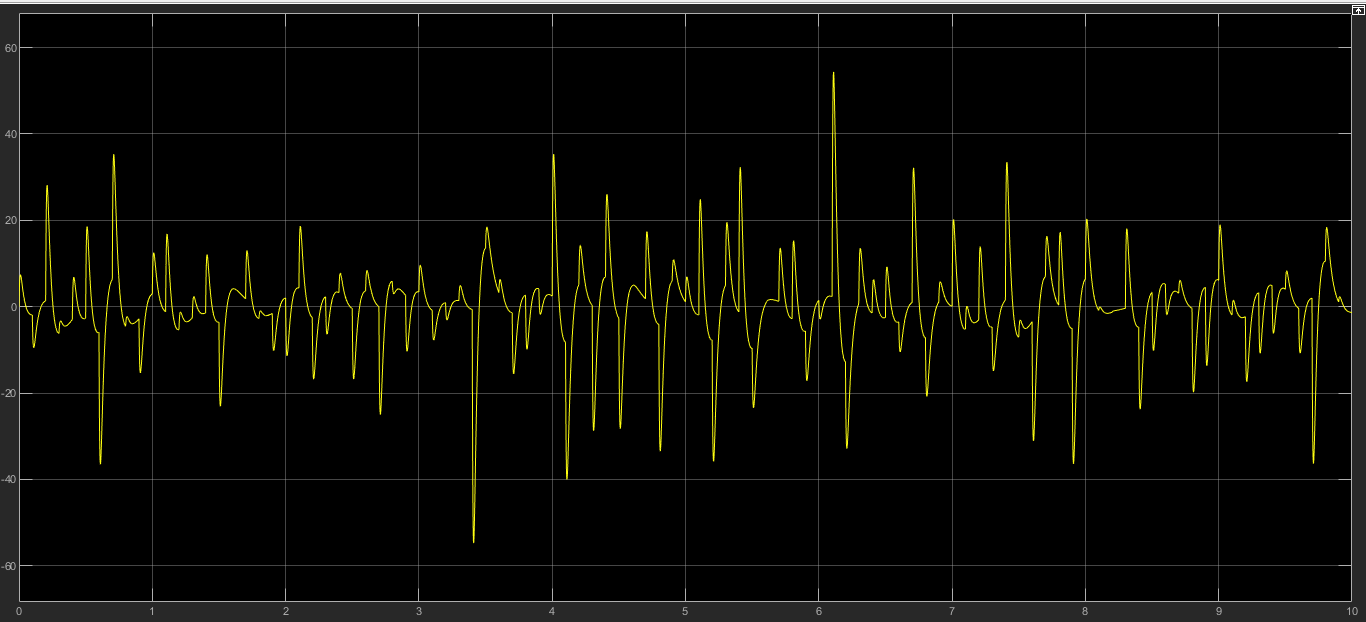


Figure 13: acc\_1

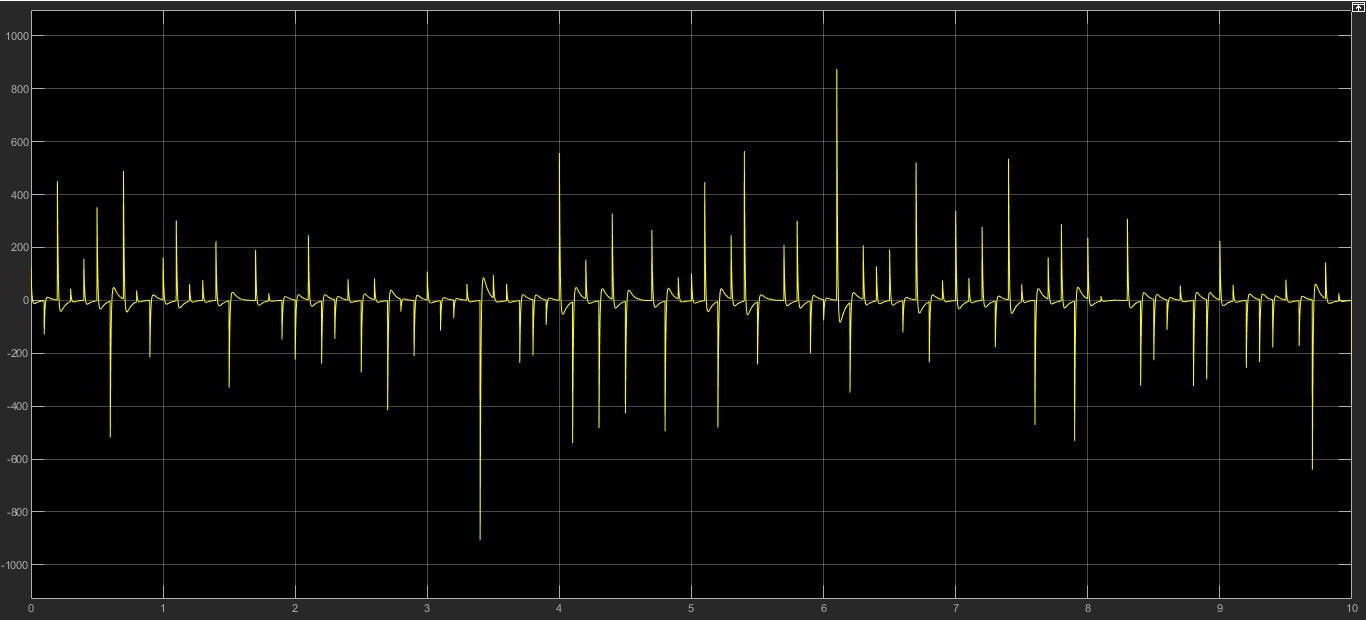


Figure 14: acc\_2

* Trapezoidal signal (Amplitude = 02, frequency = 0.628 rad /s)

When the controller is enabled, we obtained the following result for acc\_1 and acc\_2:

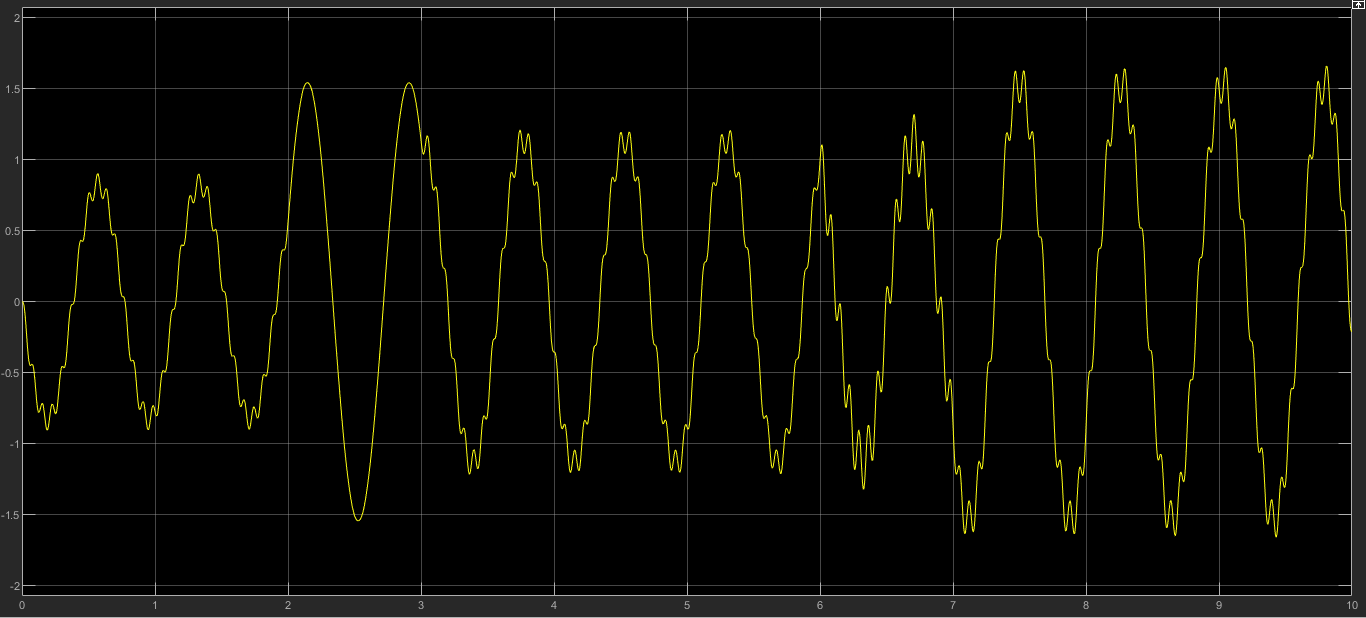


Figure 15: acc\_1

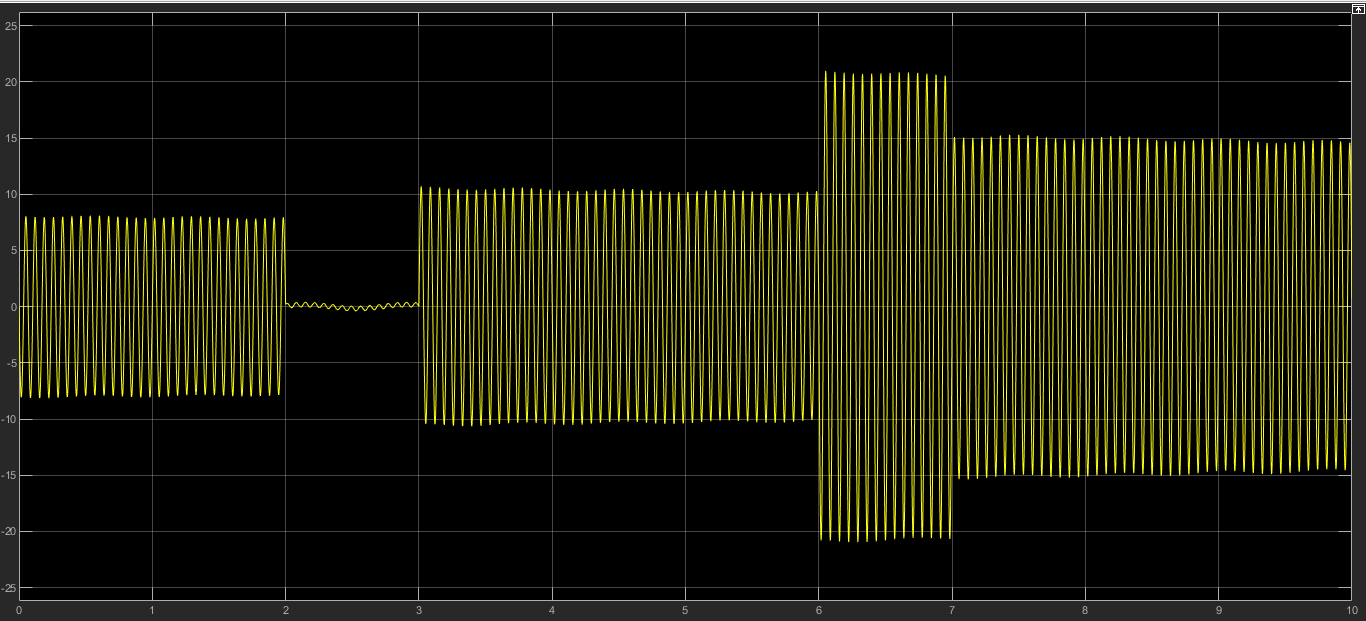


Figure 16:acc\_2

When the controller is disabled and c fixed to 6000 , we obtained the following result for acc\_1 and acc\_2:

Immagine che contiene testo, interni, piastrellato

Descrizione generata automaticamente

Figure 17: acc\_1

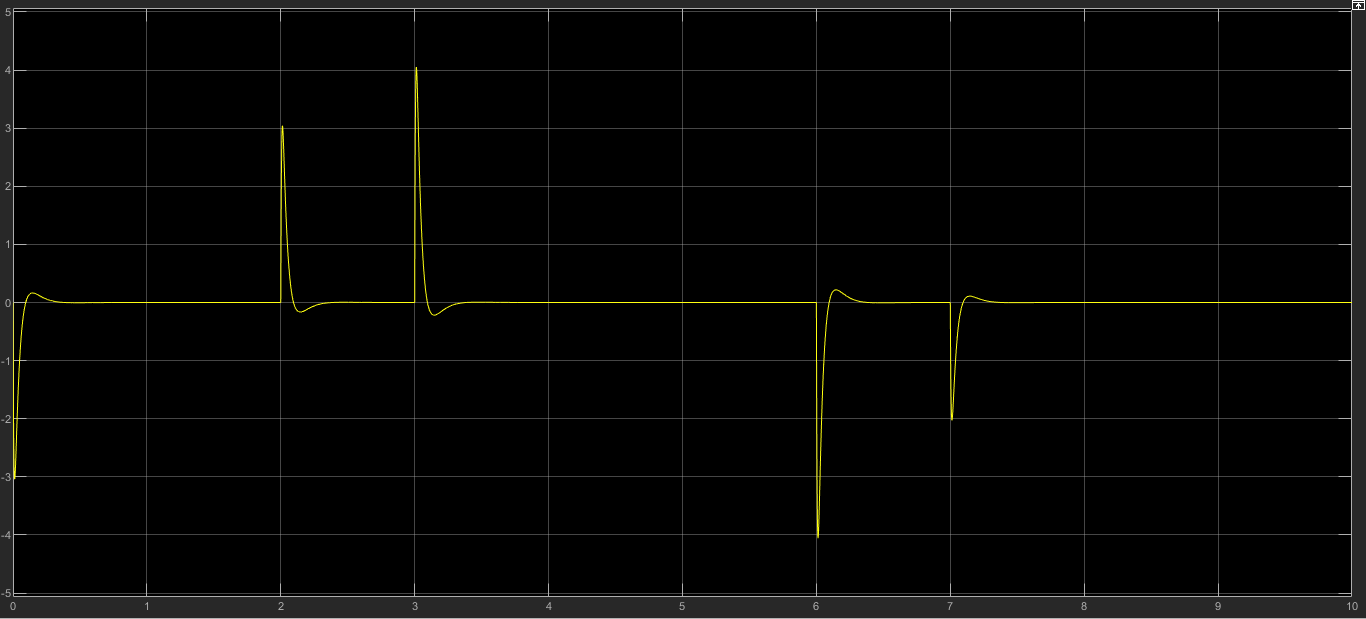


Figure 18: acc\_2

* Trapezoidal signal + white gaussian noise. (mean = 0, variance =0.001, seed = 42)

When the controller is enabled, we obtained the following result for acc\_1 and acc\_2:

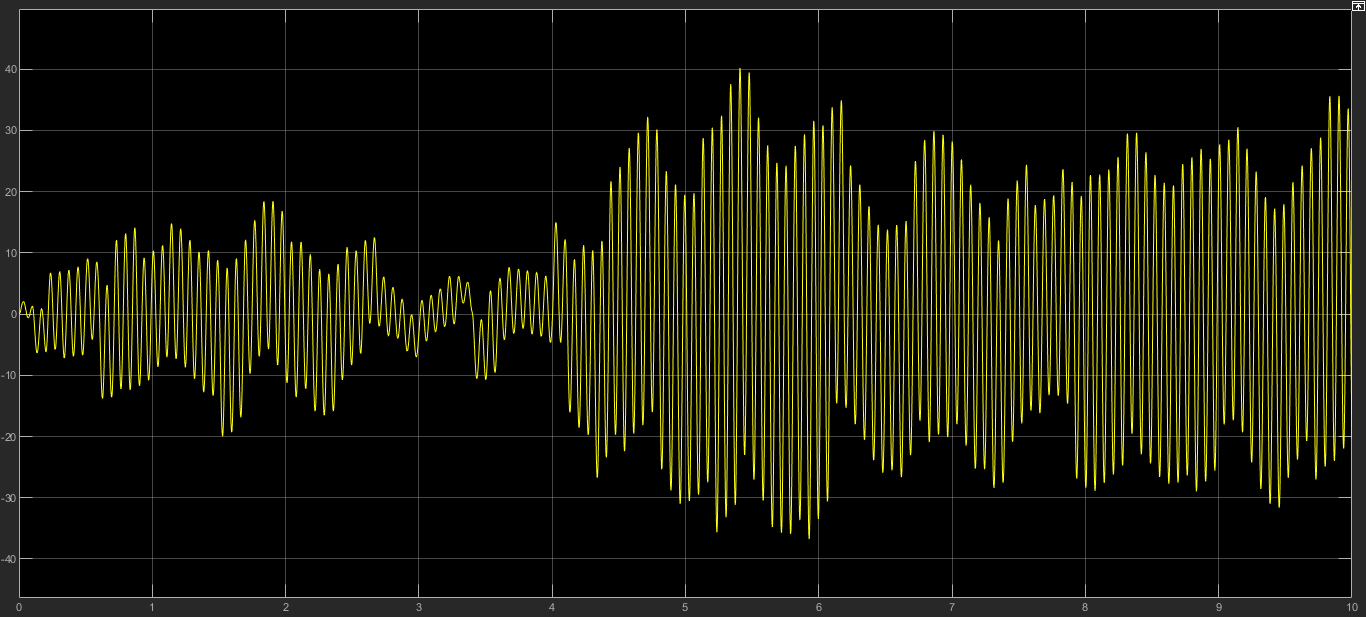


Figure 19: acc\_1

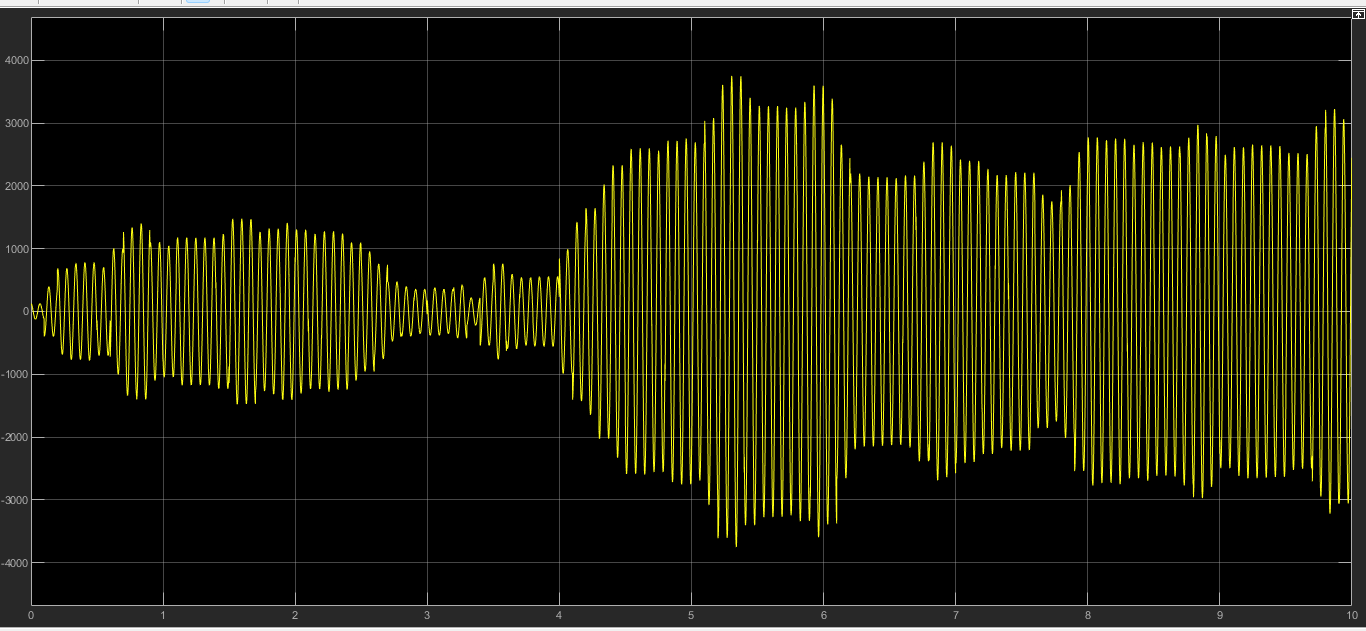


Figure 20: acc\_2

When the controller is disabled and c fixed to 6000 , we obtained the following result for acc\_1 and acc\_2:

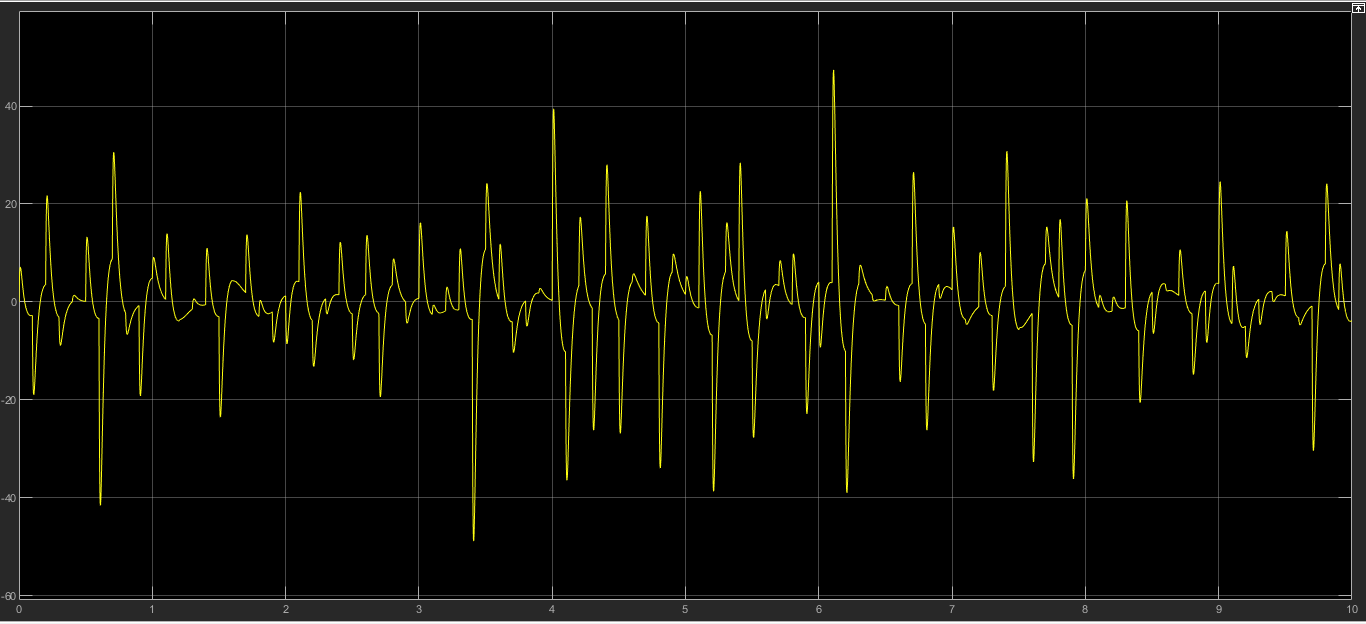


Figure 21: acc\_1

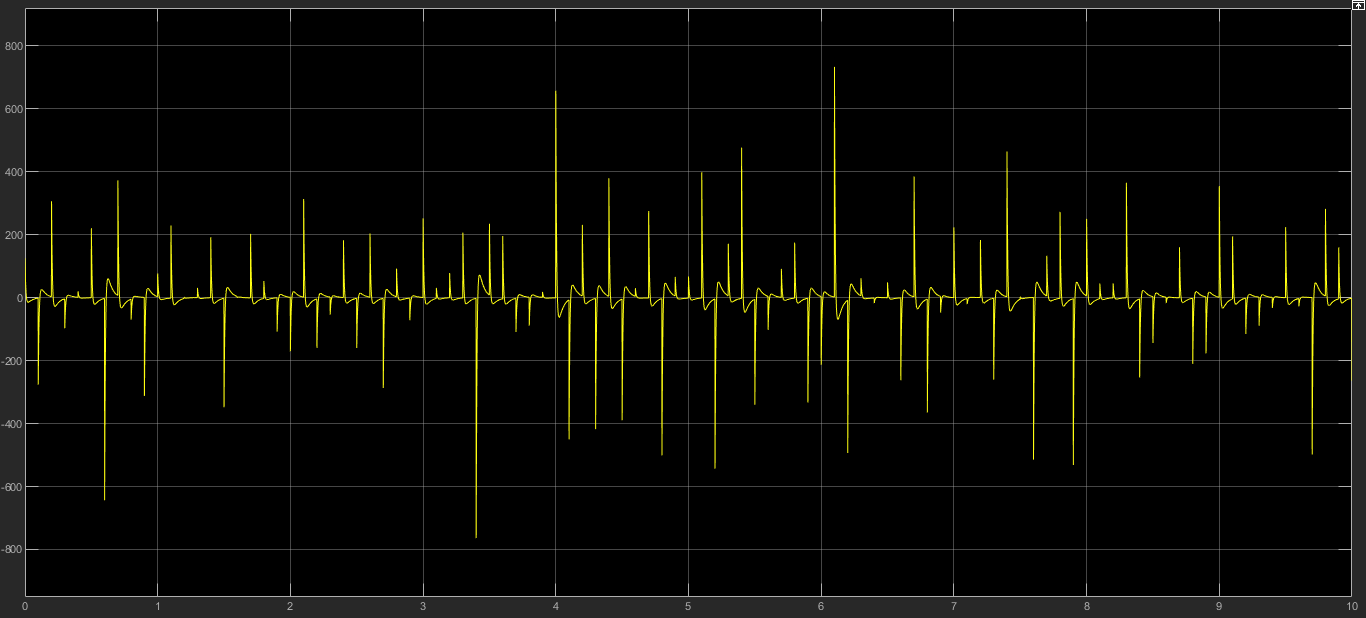


Figure 22: acc\_2

We used these two kinds of signals because they were the two closest to the reality of the road. Of course since the sinusoidal case is the worst case, we obtained high values of acceleration.

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)