POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master Degree in Computer Engineering 2019/2020

Model-Based Software Design

Prof: Violante Massimo

Student:

COSIMO MANISI – 265598

1. LAB 1

Purpose

Design a digital IIR filter whit MathWorks Simulink and perform a Model-In-the-Loop test.

A digital filter is a system that performs mathematical operations on a sampled, discrete-time signal to reduce or enhance certain aspects of that signal

Software

Software: Matlab & Simulink 2019

Assignment

- ### Filter design

  The goal is to design a second order filter in discrete-time IIR (infinite impulse response) started from the general of a low pass filter in continuous-time:

$$H(s) = \frac{k_0}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2}$$

  The general equation of IIR filter is:

$$y[n] = \sum_{i=0}^{M} b_i \cdot u[n-i] - \sum_{j=1}^{N} a_i \cdot y[n-j]$$

  For a second order filter M=N=2
  According to the requirements in a continuous-time:

$$\begin{cases} k_0 = (\omega_0)^2 \\ \omega_0 = 100 \cdot \pi \\ Q = 1 \end{cases} \rightarrow H(s) = \frac{10^4 \cdot \pi^2}{s^2 + 100 \cdot \pi \cdot s + 10^4 \cdot \pi^2}$$

  We compute the z-transform to obtain a discrete filter (in a canonical form):

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{\frac{1}{z^2}(0.0442z + 0.0398)}{\frac{1}{z^2}(z^2 - 1.646z + 0.7304)} = \frac{0.0442 z^{-1} + 0.0398\, z^{-2}}{1 - 1.646\, z^{-1} + 0.7304\, z^{-2}}$$

  In order to discretize the filter we used the c2d function.
  Starting form this transfer function i obtain the Simulink model below:

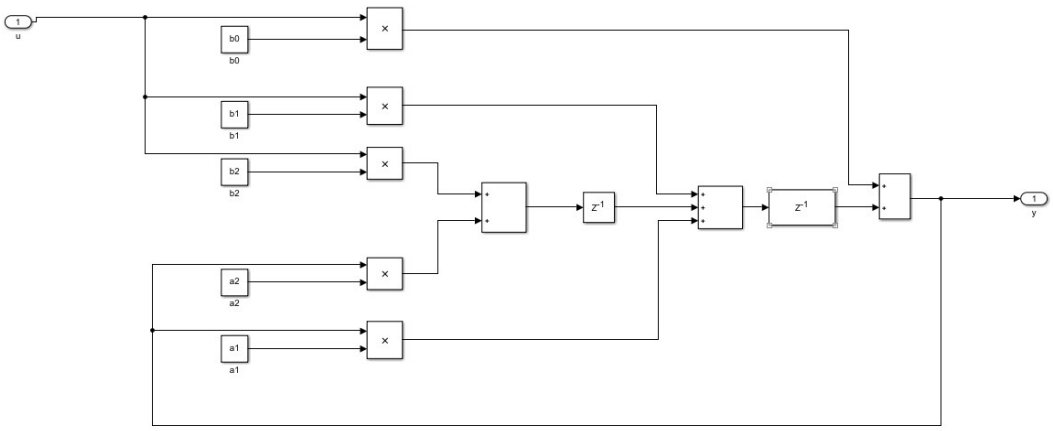


Figure 1 - Model of the filter

In the Figure 1, we have a several component:

- Constant – used to represent the value of b0, b1, b2, a1, a2 of Transfer Function *H(z)*
- Multiplication blocks
- Add/Sum Blocks
- Delay blocks – used to perform operation whit the previous value (z^{-1})

Below the Matlab code that perform the transformation continuous to discrete:

lab1.m

```
clear all
close all
clc

s=tf('s');

ct_filter= (100*pi)^2/(s^2 + (100*pi)*s + (100*pi)^2);

dt_filter = c2d(ct_filter,0.001,'zoh')

b0= 0
b1= 0.0442
b2= 0.0398

a0=1
a1=+1.646
a2=-0.7304
```

Figure 2 MatLab code

PARAMETER	VALUE
a_1	-1.646
a_2	0.7304
b_0	0
b_1	0.0442
b_2	0.0398

- ## Creation of the complete Simulink model

The model of the second order filter described above is build inside in the subsystem called ‘ Filter ‘.

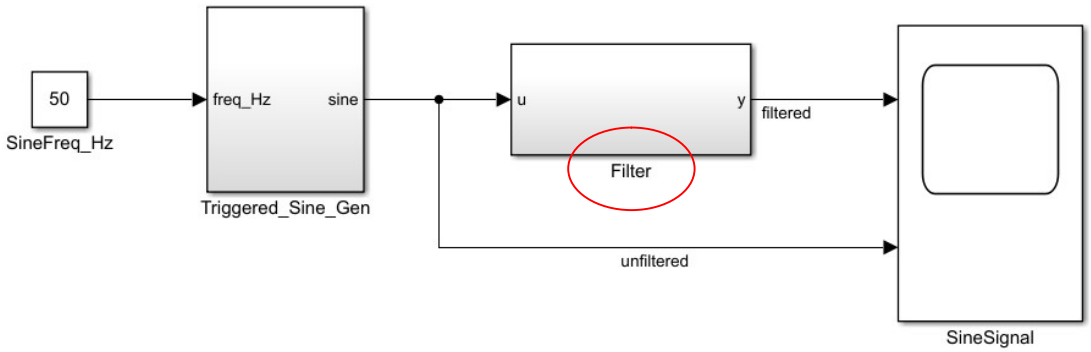


Figure 3 - Subsystem

- ## Simulation

The first check to be done in order to see if the filter work, in order to explore if the component are put together in the correct way , is to put the value b0=1 and the other set to zero. The output must be the same because this first configuration impose that the input is projected to the output.

Parameter	values
a_1	0
a_2	0
b_0	1
b_1	0
b_2	0
Frequency [Hz]	50

Figure 4 – Simulation whit no filter action

A second check that should be done is to set b1=1 and the other parameters set to zero. The output expected is the input signal translated of one step time.

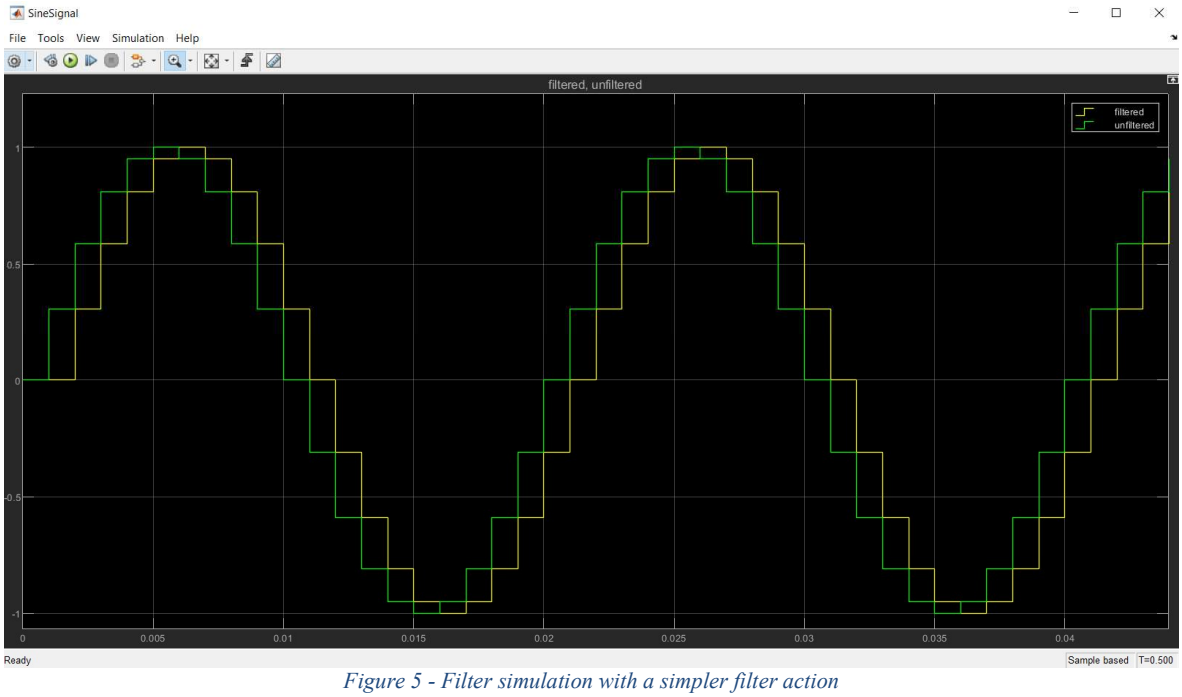


Figure 5 - Filter simulation with a simpler filter action

Now is time to tune the filter, with the coefficients obtained by means of the c2d MATLAB function and verify the output of the input signal after passed through the filter. The Figure 6, show the behavior of the filter that attenuates signals with frequencies higher than the cutoff frequency.

Parameter	values
a_1	-1.646
a_2	0.7304
b_0	0
b_1	0.0442
b_2	0.0398
Frequency [Hz]	150

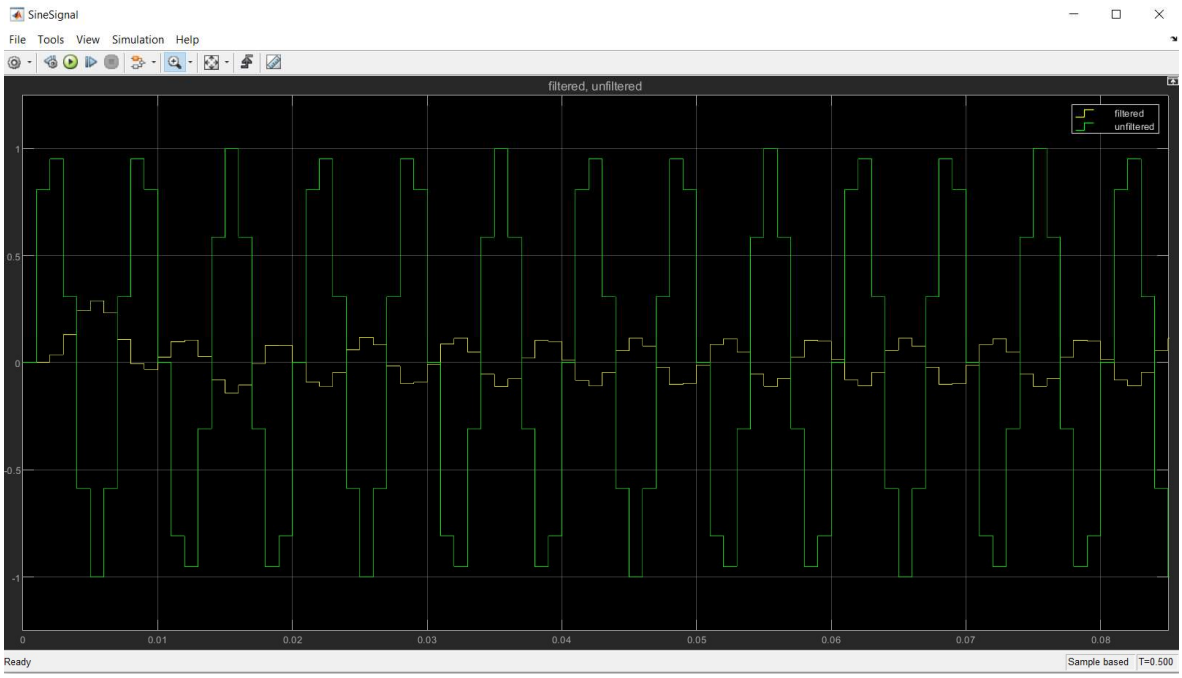


Figure 6 - Filter in action

The finally check in order to be sure that the filter work properly, I have to compute the bode diagram. At the magnitude -3db, i get the frequency and set this value as the frequency of the input signal.
-3db is a point that for definition where the signal have an attenuation of 0.71.

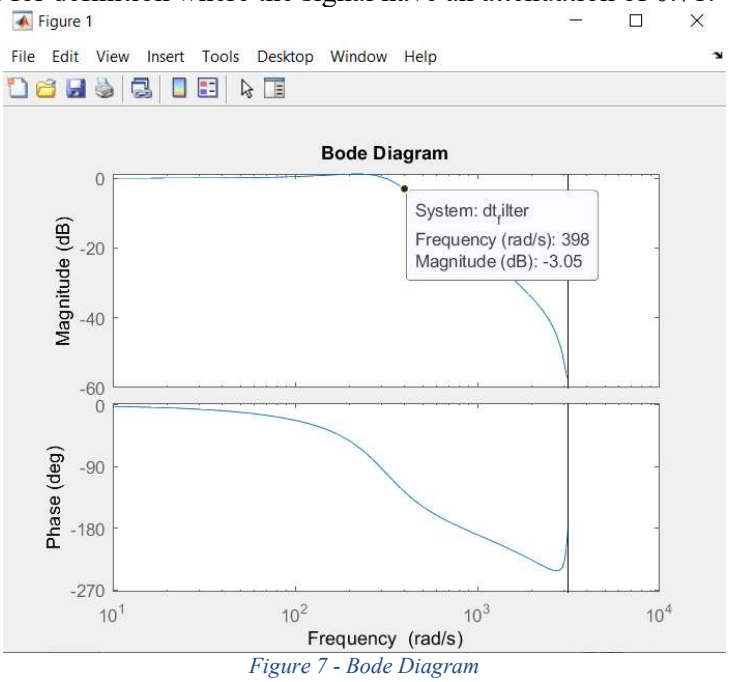


Figure 7 - Bode Diagram

The output is as expected. I can deduct that the filter is working properly.

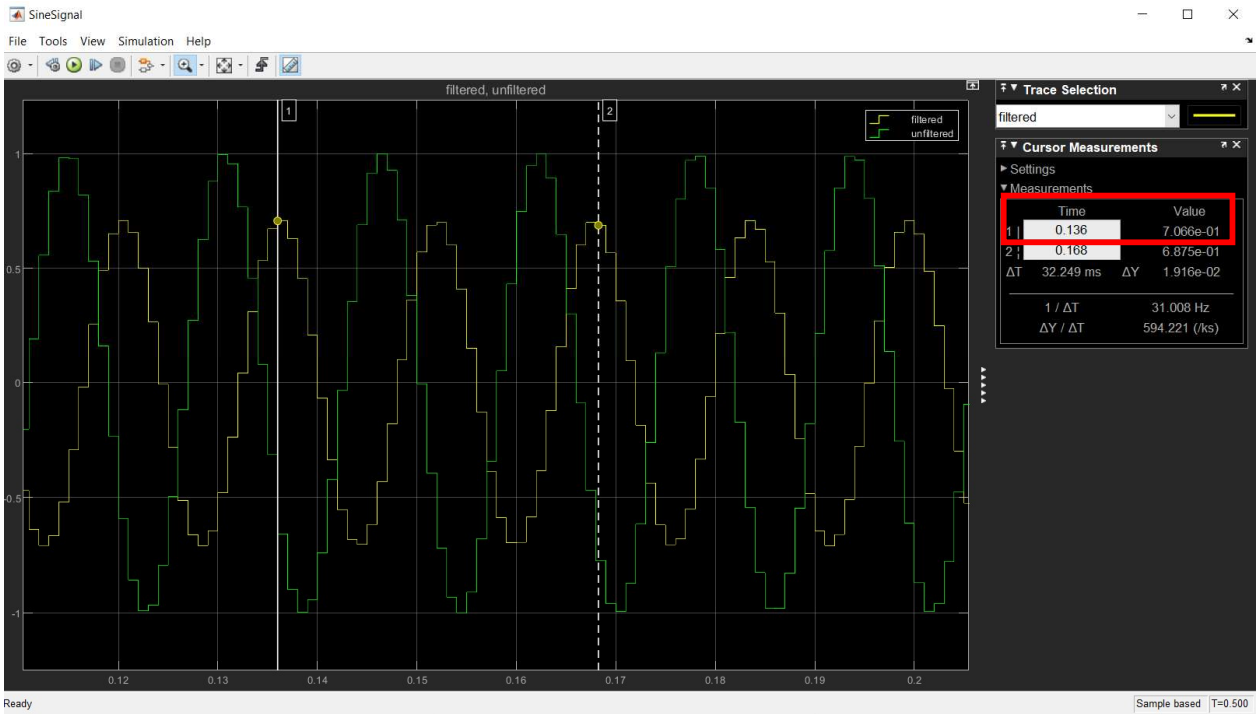


Figure 8 - Filter in action and show the attenuation of 0.71

2. LAB 2

Purpose

The three main point of the second laboratory are:

1. Modify the digital IIR filter of Lab #1 to run it on a target,
2. Generate the code to compile and run it on National Instruments VeriStand.
3. Run the filter to perform a Software-In-the-Loop test.

Software

Software: Matlab & Simulink 2019 , National Instrument Veristand 2019

Step#1

Starting the Wrapper model provided, shown below, where find on the left the parameters set discovered in the lab one, on the right the oscilloscope in order to see the signal during the simulation, and in the middle the block 'MBSD_2020_LAB2_SIL' that rappresent the wrapper to perform software in the loop testing.

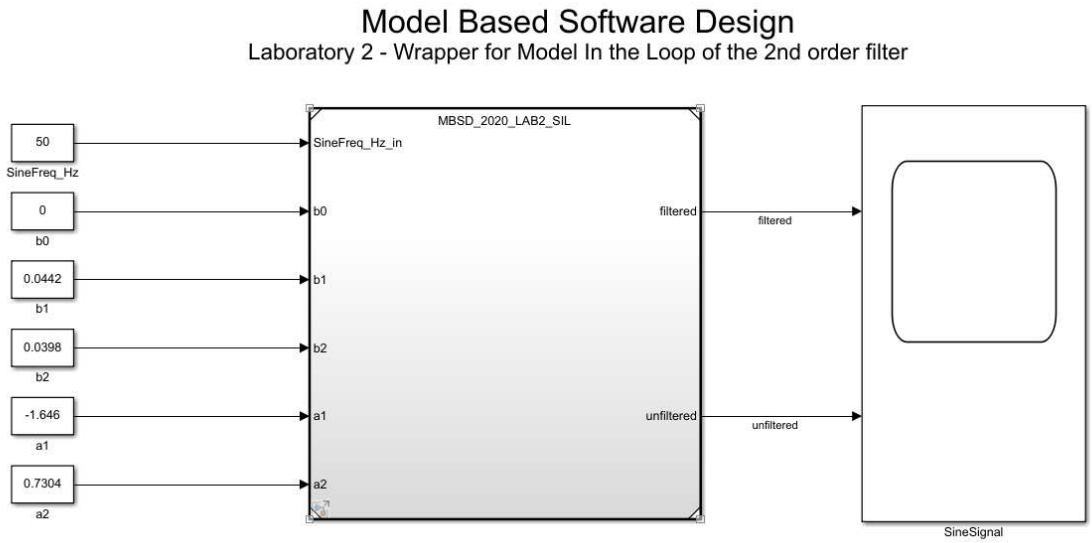


Figure 9 - First Wrapper

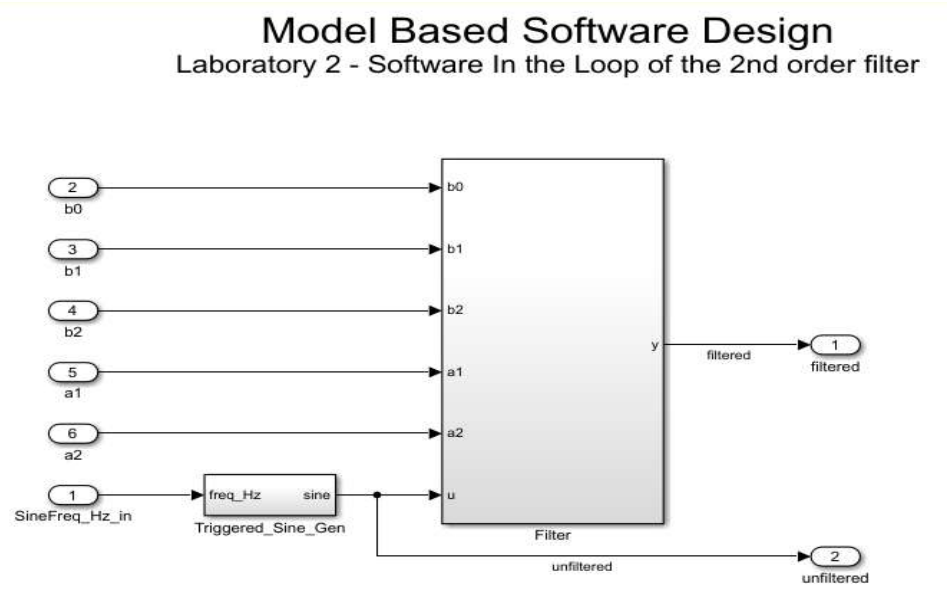


Figure 10 -Second wrapper

In the filter block we find all the block choose to implement the second order filter.

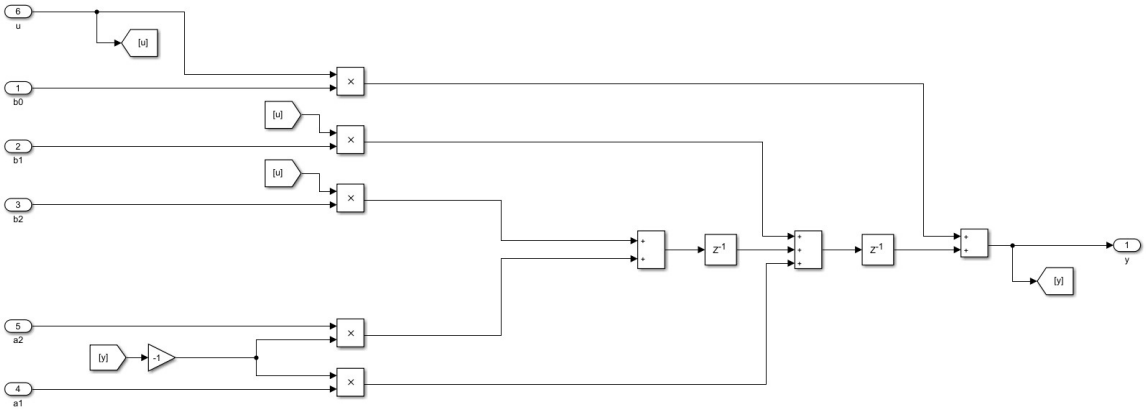


Figure 11 - The model inside the 'Filter' block

Now I'm focus on the second wrapper, Figure 10 , where we need to set a configuration in order to generate code properly.

Step#2

In general, we have to change these parameters:

- `Simulation time ->Start time:0  Stop time: inf;`
- `Solver option -> Type to fixed-step;`
- `Solver option -> Fixed-step size (fundamental sampling time)` to a suitable value depending on simulation (for the filter 1/1000 s).

At this point it's useful select the right target for the cross-compiler. Some pre-built scripts are available for most common targets. Change the following setting.

- `Code generation ->Target selection -> System target -> VeryStand.tlc;`
- `Code generation ->Toolchain -> MinGW64 | gmake (64 bit Windows);`

Now i'm ready to generate code and compile it. The diagnostic viewer show me that all is correct, and no error are present.

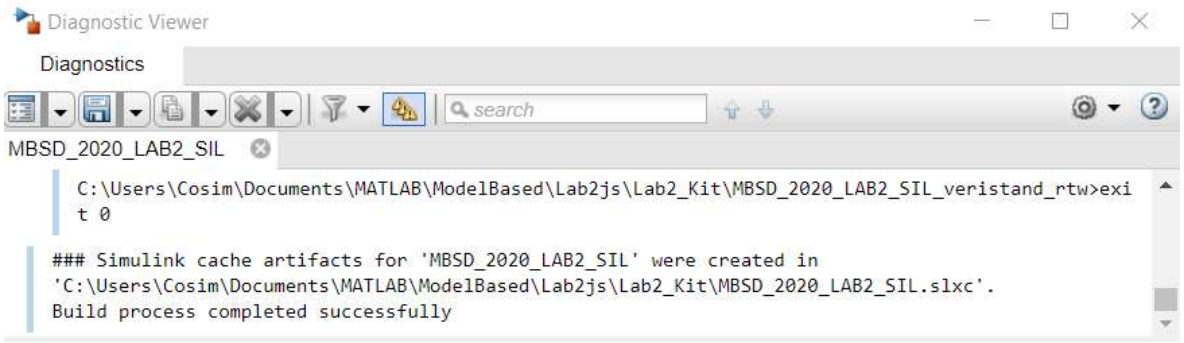


Figure 12 - Diagnostic viewer after compile

Step#3

The first step is import the " .dll " file obtained from the Simulink model to the VeriStand project.
In the graphical user interface of the project i build a model and map the controls to the I/Os of the model :

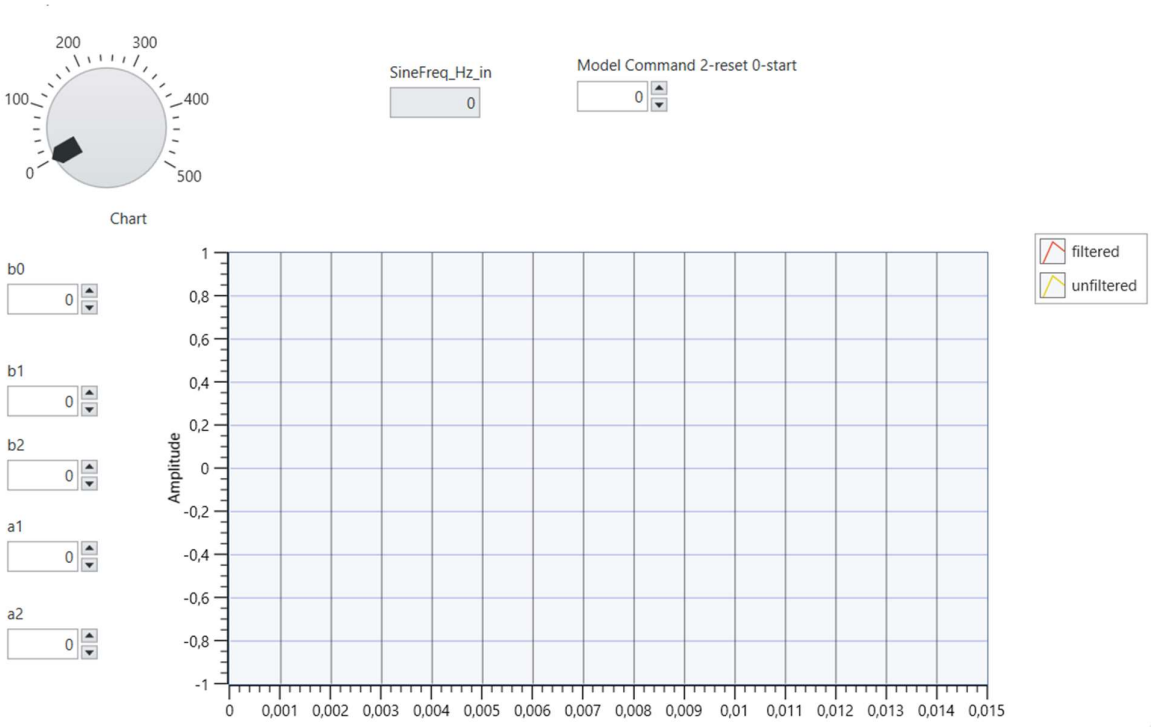


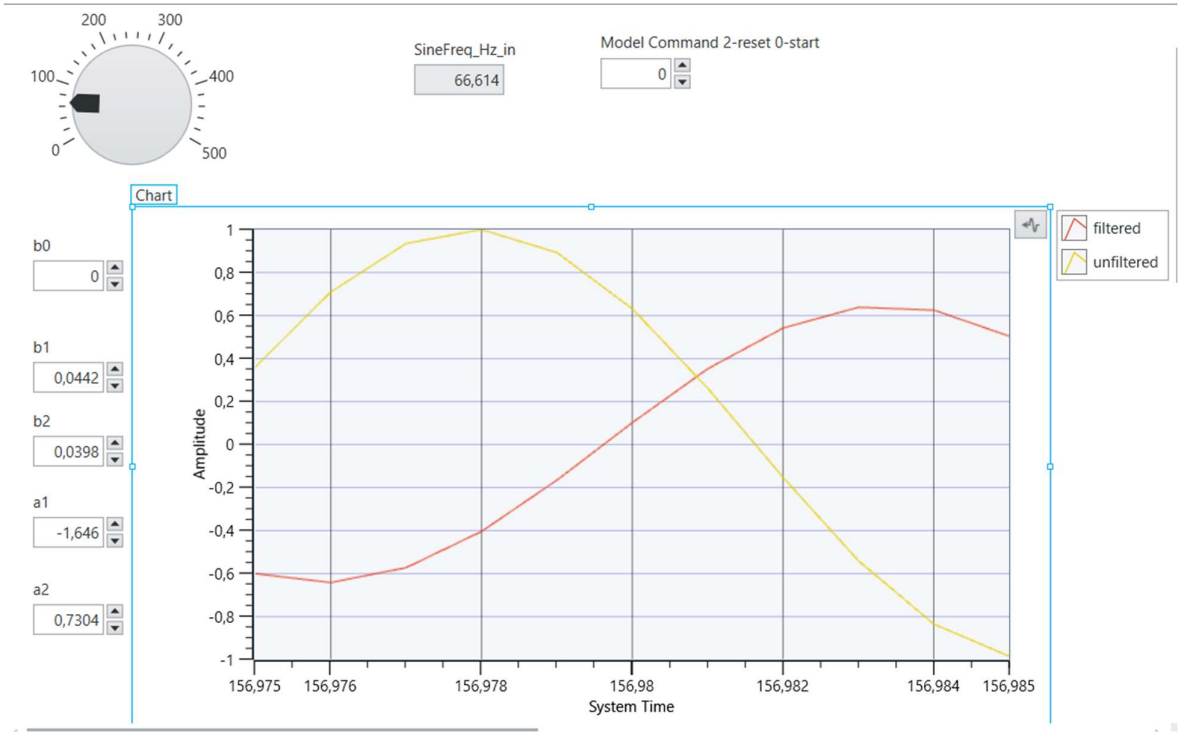
Figura 13 - Graphical User Interface

After all the signal has been properly mapped, i deploy the model whit the following parameters:

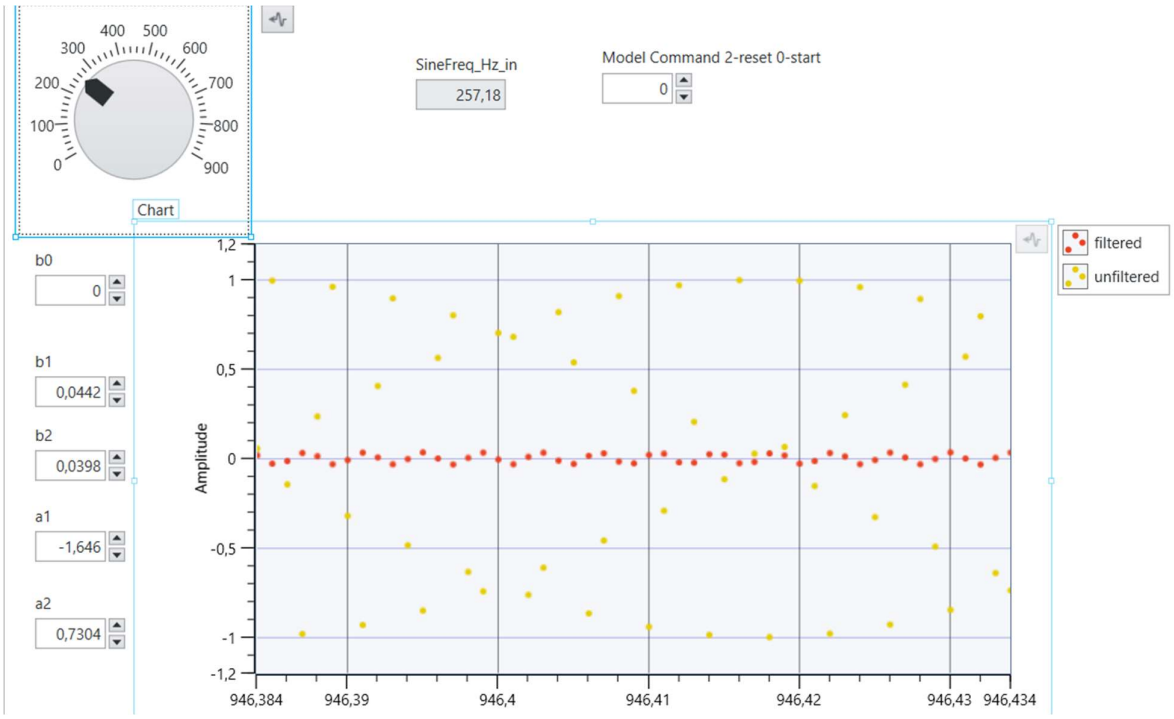
Parameter	values
a_1	-1.646
a_2	0.7304
b_0	0
b_1	0.0442
b_2	0.0398

Simulation

The First simulation is done with frequency about 65 Hz. The filter work as expected.The signal that have a frequency greater than 50 Hz is attenuate. The signal in red is the filtered signal the comes out of the filter.



The second simulation is performed by accelerating the frequency up to 250 Hz. At this frequency I get a signal that apparently has a strange pattern. This phenomenon is called "aliasing of perception". This is because the setting it is not sufficient to satisfy the requirement of Shannon's theorem.



If I keep increasing the frequency, the graph starts to emit a sinusoidal signal. This signal is correct because it is sinusoidal but wrong because the signal is under sampled. This situation in real life is usually canceled with another filter (anti-aliasing filter).