

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2021/2022 – 1st semester

Week11, 10 Jan 2022

Objectives:

-design, analysis, implementation and real-time operation of a 6th-order band-stop filter:

- design and decomposition of the 6th-order band-stop filter into second-order sections
- implementation of 6th-order band-stop filter for real-time operation on the STM32F7 Discovery kit
- conversion of the 6th-order band-stop filter into a 6th-order all-pass filter by just acting on the C code implementing the cascade of second-order sections

DSP Education Kit

LAB 10

Infinite Impulse Response (IIR) Filter

Issue 1.0

Contents

1	Introduction.....	1
1.1	Lab overview	1
2	Requirements	1
3	IIR band-stop filter design and characteristics.....	1
4	Preparing the IIR band-stop filter for real-time operation	4
5	Measuring the IIR band-stop filter Frequency Response	7
5.1	Setting-up for this lab experiment.....	7
5.2	Sketching the frequency response magnitude of the IIR band-stop filter.....	7
6	Converting the IIR band-stop filter into an all-pass filter	8
7	Conclusions.....	11
8	Additional References.....	11

1 Introduction

1.1 Lab overview

This Lab motivates the design and experimental test of a 6th-order IIR filter that is implemented as a cascade of three second-order sections (or biquads). The filter consists of a band-stop filter whose structure of poles and zeros is analyzed. The frequency response of the filter is measured when it is operating in real-time and a simple filter modification is proposed converting the 6th-order band-stop filter into a 6th-order all-pass filter preserving the poles of the band-stop filter.

2 Requirements

To carry out this lab, you will need:

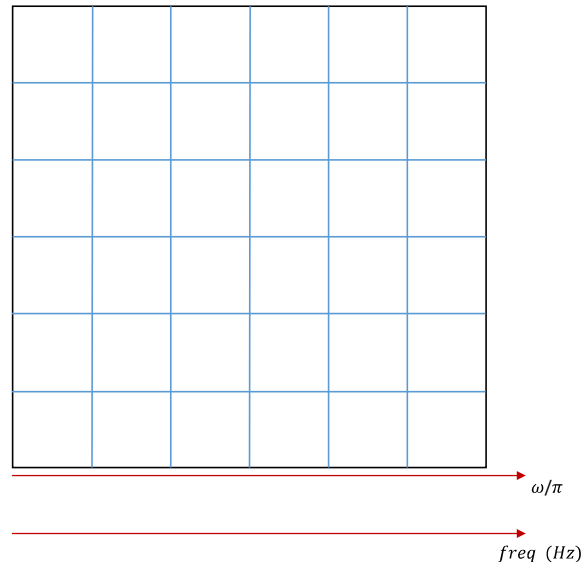
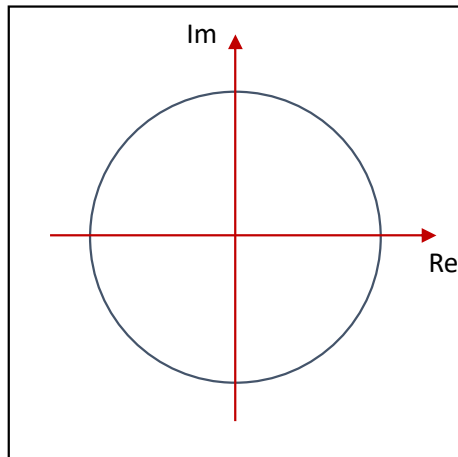
- An STM32F746G Discovery board
- A PC running Keil MDK-Arm
- MATLAB
- An oscilloscope
- Suitable connecting cables
- An audio frequency signal generator

3 IIR band-stop filter design and characteristics

Our IIR band-stop filter correspond to the 6th-order Butterworth filter that was discussed in the context of the week09 TP class, Problem 2. That filter passes the frequency bands between 0 and $2\pi/5$, and $3\pi/5$ and π , and rejects frequencies in between. It is designed using the following Matlab commands:

```
format long
wc=[2/5 3/5];
[b, a]=butter(3,wc,'stop');
[H W]=freqz(b,a,512);
figure(1)
plot(W/pi,abs(H))
```

Question 1: Assuming that the sampling frequency is 8 kHz, represent graphically the zero-pole diagram as well as the frequency response magnitude of the 6th-order band-stop IIR filter. Represent frequency using two different but equivalent frequency axes: ω/π (in the range $[0, 1]$) and f_{req} (in Hz, in the range $[0, F_N]$ where F_N is the Nyquist frequency). In particular, taking into consideration the definition of -3 dB cutoff frequency, specify what the gain (linear, not dB) is for $\omega=2\pi/5$, and $\omega=3\pi/5$, relative to the case where $\omega=0$ rad.



$\omega = 2\pi/5 \rightarrow f =$ (Hz) \rightarrow GAIN =

$\omega = 3\pi/5 \rightarrow f =$ (Hz) \rightarrow GAIN =

Now, we decompose the transfer function of the 6th-order Butterworth filter into a cascade of three second-order sections by executing the following Matlab commands:

```
[SOS] = tf2sos(b,a);
figure(2)
zplane(SOS(1,1:3), SOS(1,4:6))
```

As explained in the preliminary considerations of the week09 TP class Problem 2 (also by typing `help tf2sos` on the Matlab command window), in this case, `SOS` is a 3 by 6 matrix with the following structure:

```
SOS = [ b00 b01 b02 1 a01 a02
        b10 b11 b12 1 a11 a12
        b20 b21 b22 1 a21 a22 ]
```

where each row specifies the coefficients of the transfer function of each second-order section, as illustrated in Figure 1.

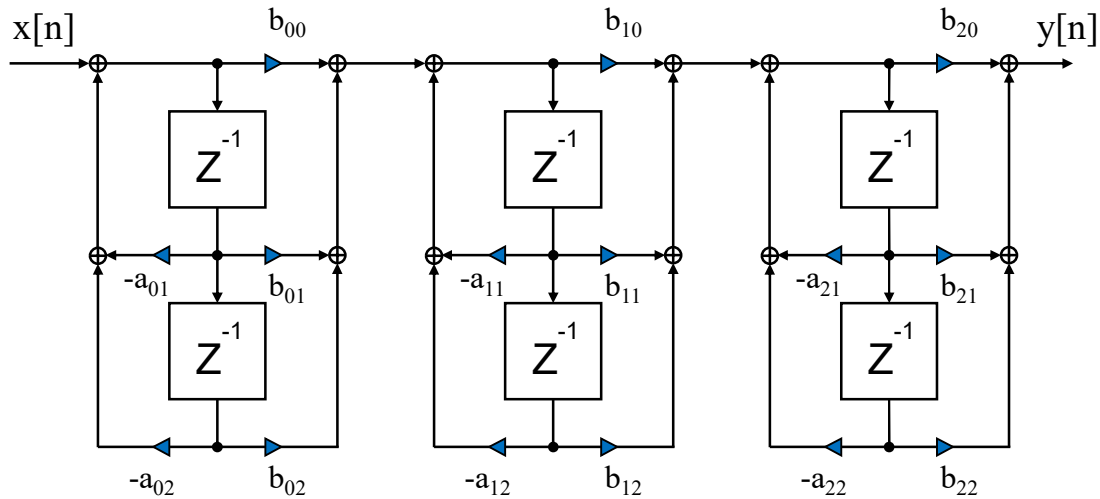
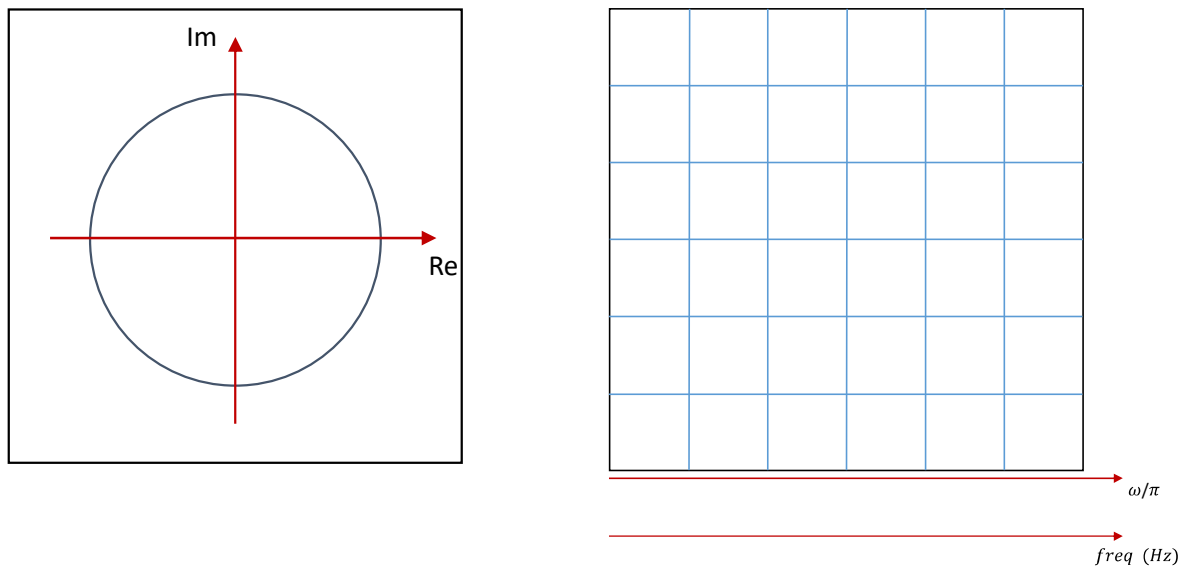


Figure 1: Cascade of three second-order sections (Direct form II implementations) of an IIR filter

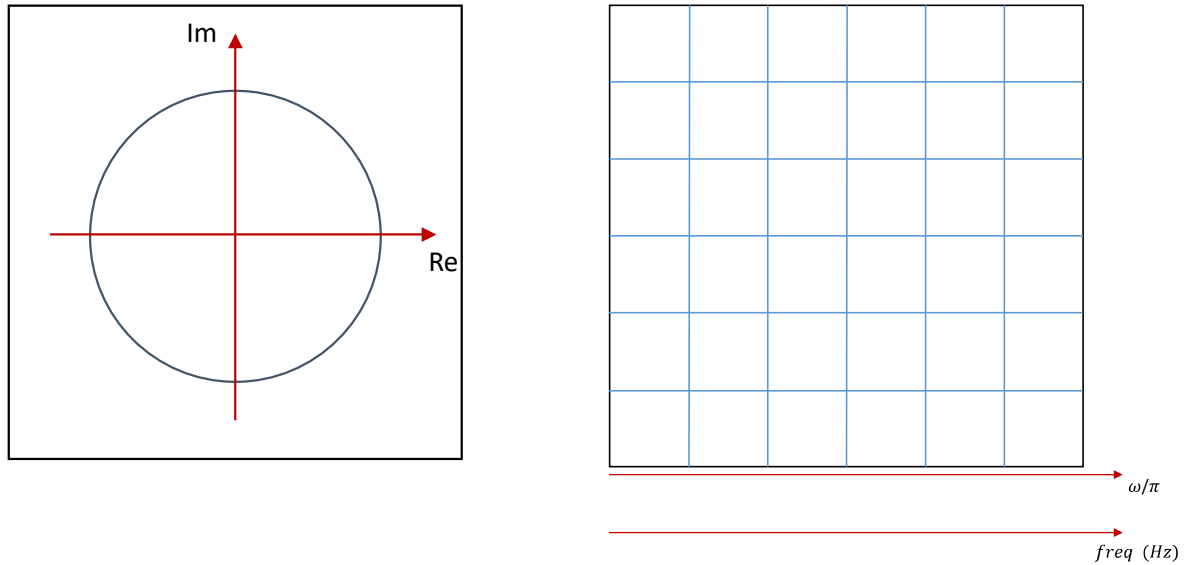
The above Matlab command also plots the zero-pole diagram characterizing the first second-order section.

Question 2: Represent next the zero-pole diagram as well as the frequency response magnitude of each individual second-order section (or biquad), and explain why zeros and poles occur in complex-conjugate pairs. Are the individual plots consistent with the plots characterizing the 6th-order band-stop IIR filter as represented in your answer to Question 1 ? In particular, what specific aspect of the 6th-order band-stop filter is that the poles off of the imaginary axis help to specify ?

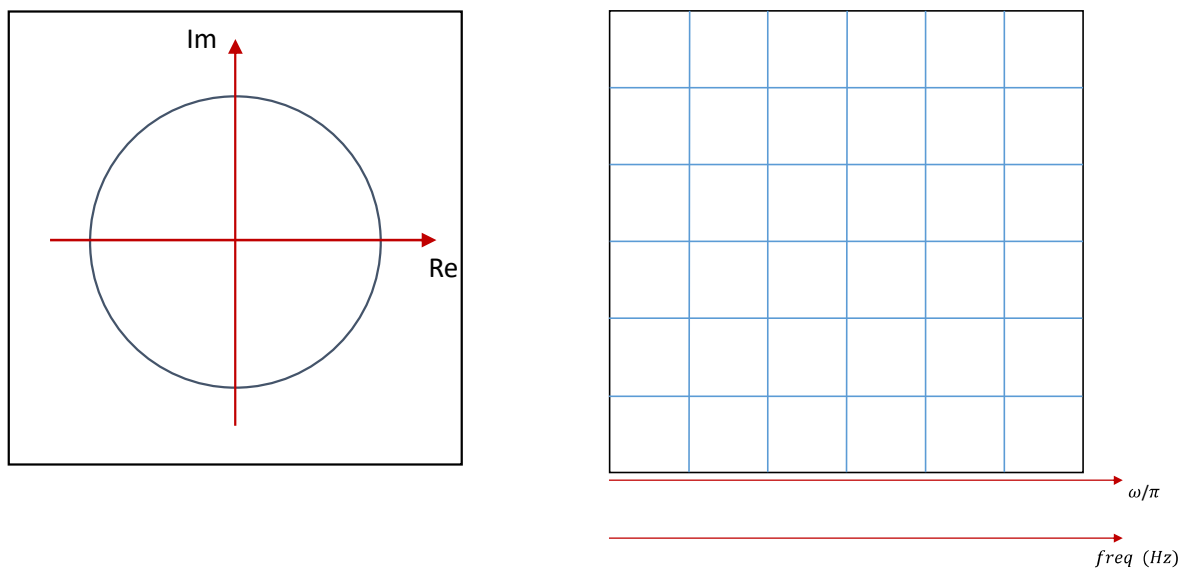
Zero-pole diagram and frequency response magnitude of the first biquad:



Zero-pole diagram and frequency response magnitude of the second biquad:



Zero-pole diagram and frequency response magnitude of the third biquad:



4 Preparing the IIR band-stop filter for real-time operation

The SOS matrix specifying the coefficients of the transfer functions of all second-order sections will be used in order to create a header file, named `bandstopIIR.h`, that will be included in the main file of the STM32F7 project. In that sense, we use the Matlab function named

`iirsos_coeffs_mod()` that is available on the Moodle platform. At the MATLAB terminal, type `iirsos_coeffs_mod(SOS)` and enter the filename, `bandstopIIR.h`. Check this file in order to understand how the coefficients are arranged in vectors `b[] []`, and `a[] []`.

Question 3: Given that all second-order sections will be implemented using Direct form Type 2 realization structures, i.e., using the corresponding difference equations, what is common and what is different between the coefficients in vectors `b[] []` and `a[] []` and the coefficients used by the realization structures?

In this Lab, we use the `main()` project file that is named `stm32f7_iirsos_intr_FPS.c` and that is available on the Moodle platform. Its C code is listed next.

```
// stm32f7_iirsos_intr_FPS.c

#include "stm32f7_wm8994_init.h"
#include "stm32f7_display.h"
#include "bandstopIIR.h"

#define SOURCE_FILE_NAME "stm32f7_iirsos_intr_FPS.c"

extern int16_t rx_sample_L;
extern int16_t rx_sample_R;
extern int16_t tx_sample_L;
extern int16_t tx_sample_R;

float w[NUM_SECTIONS][2] = {0};

void BSP_AUDIO_SAI_Interrupt_CallBack()
{
    int16_t section;    // second order section number
    float32_t input;    // input to each section
    float32_t wn, yn;   // intermediate and output values

    input =(float32_t)(rx_sample_L);
    for (section=0 ; section < NUM_SECTIONS ; section++)
    {
        wn = input - a[section][1]*w[section][0]
              - a[section][2]*w[section][1];
        yn = b[section][0]*wn + b[section][1]*w[section][0]
              + b[section][2]*w[section][1];
        w[section][1] = w[section][0];
        w[section][0] = wn;
        input = yn;
    }
    tx_sample_R = (int16_t)(yn); // will appear in OUT LEFT channel
    tx_sample_L = rx_sample_L;   // will appear in OUT RIGHT channel

    return;
}

int main(void)
{
    stm32f7_wm8994_init(AUDIO_FREQUENCY_8K,
```

```

        IO_METHOD_INTR,
        INPUT_DEVICE_INPUT_LINE_1,
        OUTPUT_DEVICE_HEADPHONE,
        WM8994_HP_OUT_ANALOG_GAIN_0DB,
        WM8994_LINE_IN_GAIN_0DB,
        WM8994_DMIC_GAIN_0DB,
        SOURCE_FILE_NAME,
        NOGRAPH);
while(1){}
}

```

Take a moment to analyze this code, to understand how each second-order section is implemented, how the cascade is organized and implemented, and to check the consistency between this code and your answer to Question 3.

Now we proceed, as indicated next, to compile the code, upload it to the STM32F7 board, and to run it.


After unzipping them, take the `stm32f7_iirsos_intr_FPS.c` and `bandstopIIR.h` files to the “src” directory that is located under the folder:

C:\uvision\Keil\STM32F7xx_DFP\2.9.0\Projects\STM32746G-Discovery\Examples\DSP Education Kit\

Remember that the directory where you can find the **DSP_Education_Kit.uvprojx** project file is:

C:\uvision\Keil\STM32F7xx_DFP\2.9.0\Projects\STM32746G-Discovery\Examples\DSP Education Kit\MDK-ARM

You can copy-paste this link directly to File Explorer in Windows for a quick and easy access. For your convenience, this link is also available on a TXT file on Moodle.

As in previous labs, we use the **DSP_Education_Kit.uvprojx** project file as our baseline project. This project file is represented by the icon  **DSP_Education_Kit.uvprojx**, or just  **DSP_Education_Kit**. Double-click on this file/icon to start the Keil MDK-Arm development environment (µVision). Replace the existing `main()` file in that project by the new `main()` that is `stm32f7_iirsos_intr_FPS.c`.

Now, proceed as usual to compile the code, downloading it to the STM32F746G board (by starting the debugger), and then to run the code.

5 Measuring the IIR band-stop filter Frequency Response

5.1 Setting-up for this lab experiment

Set the function generator to generate a sine wave having 5 Vpp and 100 Hz. Using a “T” and a BNC-BNC cable, take the output of the function generator to CHAN1 of the oscilloscope, and also to the (left channel of the) LINE IN socket on the Discovery board (**Remember: make sure that you use the adapter with the blue mini-jack whose interface board has a resistor divider. It is meant to protect the analog input of the *kit* against excessive voltage levels**).

Then, using another BNC-BNC cable, take the LEFT channel (yes, LEFT channel) of the STM32F746G LINE OUT output to the CHAN2 input of the oscilloscope.

Using the oscilloscope SETTINGS button and menu, make sure that the Vpp and frequency of both input and output signals are being measured in real-time.

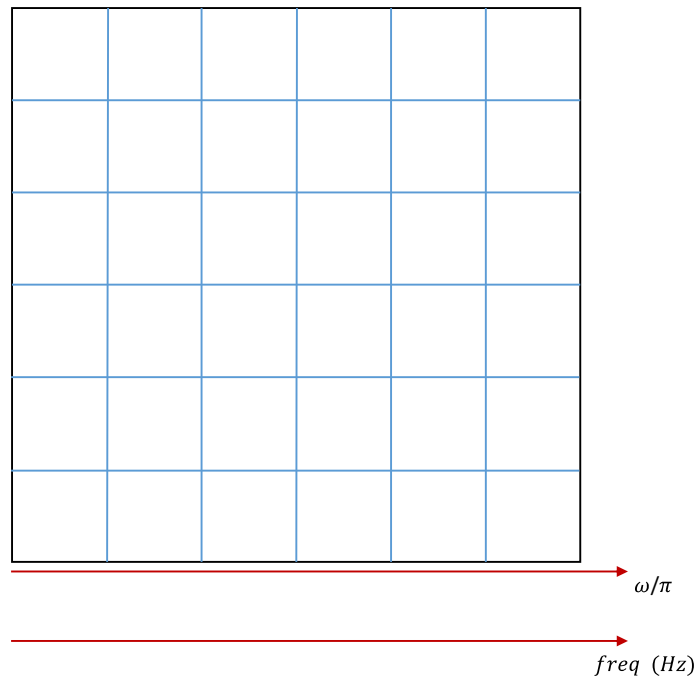
5.2 Sketching the frequency response magnitude of the IIR band-stop filter

The frequency response of a filter reflects its gain at different frequencies. As seen in previous lab experiments, one way of assessing the frequency response of the filter is simply to measure its gain using a sinusoidal input signal at a few frequencies of interest.

Here, we measure the magnitude frequency response of the IIR band-stop filter whose theoretical frequency response is your answer to Question 1.

You will vary the frequency from 100 Hz, up to 3.8 kHz, and take note of the Vpp and frequency of the output wave represented on the oscilloscope. Only a few frequencies should suffice: 100 Hz (which we will take as an approximation for 0 Hz), 3800 Hz (which we will take as an approximation for 4 kHz), the two cut-off frequencies that are anticipated in Question 1, and just a few more frequencies to characterize the filter response in the stop-band.

Based on these measurements, sketch in the following plot the approximate frequency response magnitude of the IIR band-stop filter that is implemented by the above C code when you vary the frequency from 100 Hz up to 3.8 kHz.



Question 4: Is this sketch consistent with the expected frequency response magnitude that is specified in your answer to Question 1 ? Are the measured cut-off frequencies as expected in your answer to Question 1 ?

6 Converting the IIR band-stop filter into an all-pass filter

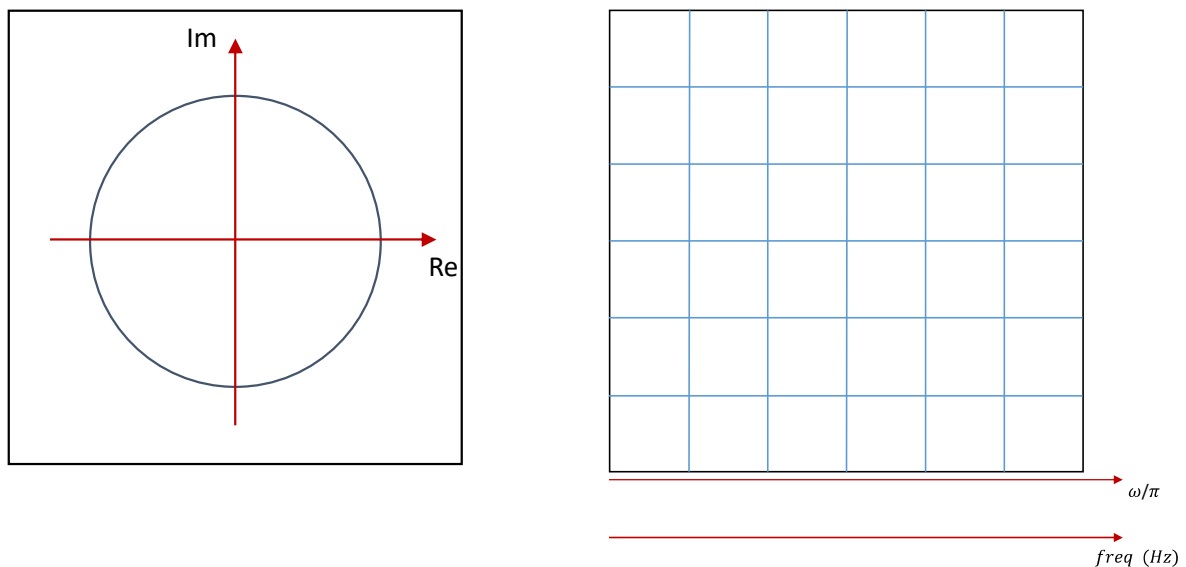
Now, we convert the above IIR band-stop filter into a 6th-order all-pass filter that preserves the same poles of the IIR band-stop filter. This should be achieved by just changing the `Interrupt_CallBack()` function in a simple and appropriate way:

```
void BSP_AUDIO_SAI_Interrupt_CallBack()
{
    int16_t section;    // second order section number
    float32_t input;    // input to each section
    float32_t wn, yn;   // intermediate and output values

    input =(float32_t)(rx_sample_L);
    for (section=0 ; section < NUM_SECTIONS ; section++)
    {
        ...
    }
    tx_sample_R = (int16_t)(yn); // will appear in OUT LEFT channel
    tx_sample_L = rx_sample_L;   // will appear in OUT RIGHT channel
}
```

```
return;
}
```

Question 5: Represent next the zero-pole diagram of the all-pass filter, as well as its expected frequency response.



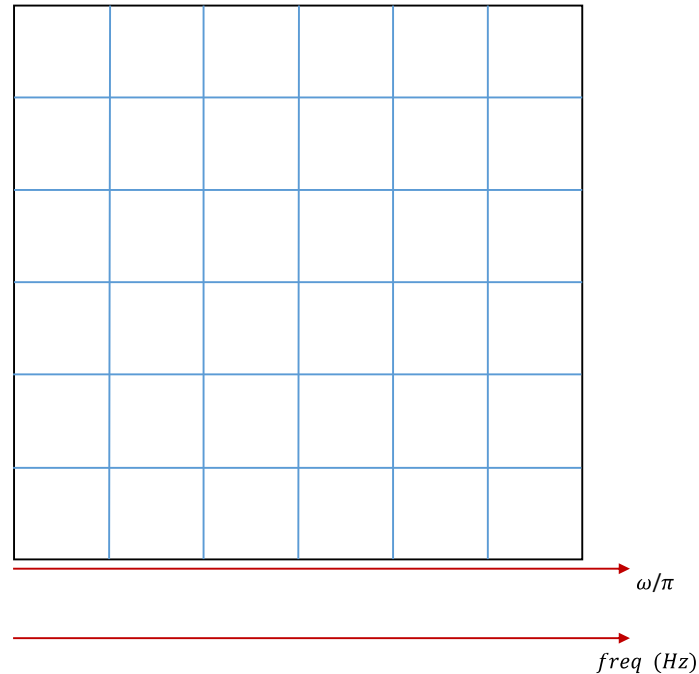
Create a copy of the above `stm32f7_iirsos_intr_FPS.c` code and name it `stm32f7_iirsos_intr_FPS_AP.c`. Perform the appropriate modifications in this new C code in order to program a 6th-order all-pass filter that preserves the same poles of the above IIR band-stop filter.

Now, proceed as usual to set this code as the main project file, to compile the code, to download it to the STM32F746G board (by starting the debugger), and then to run the code.

Take the two STM32F746G LINE OUT channels to the CHAN1 and CHAN2 inputs of the oscilloscope. When you vary the frequency of an input sinusoid from 100 Hz, up to 3.8 kHz, the Vpp amplitudes in the two channels should be comparable (given that both channels are affected by all-pass filters having the same frequency response magnitude). If the Vpp amplitudes in the two channels are not comparable for all frequencies in the range 100 Hz - 3.8 kHz then, most likely, your `stm32f7_iirsos_intr_FPS_AP.c` code contains errors, fix them before proceeding.

Note: refrain from using trial-and-error: think first of what could be wrong before you try to change/fix the C code.

By observing the real-time operation of the all-pass filter, sketch in the following plot its approximate frequency response magnitude when you vary the frequency from 100 Hz up to 3.8 kHz.



Question 6: Is this sketch consistent with the expected frequency response magnitude that is specified in your answer to Question 5 ? If not, what is different and why ?

Now, use as input a sawtooth wave (i.e., a triangular wave with a 5% duty cycle) having a fundamental frequency of 200 Hz and keep the two STM32F746G LINE OUT channels connected to the CHAN1 and CHAN2 inputs of the oscilloscope. You should obtain a graphical representation on the oscilloscope that is similar to the screenshots that are represented in Figure 2, where one channel represents the input to the all-pass filter, and the other represents the all-pass filter output.

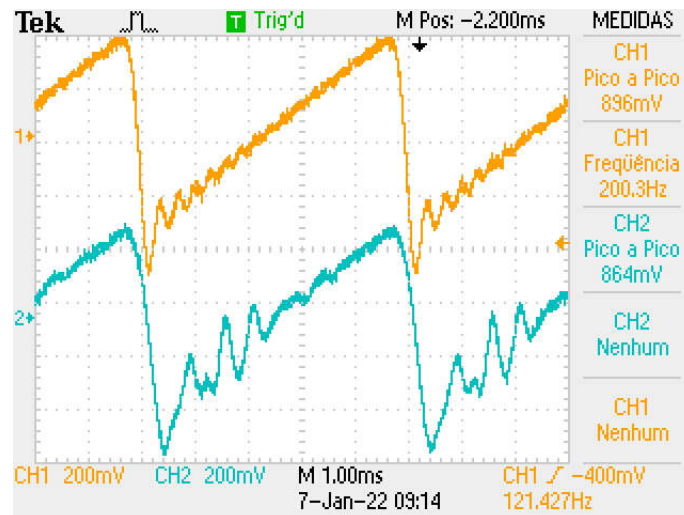


Figure 2: Oscilloscope screenshot representing the input and output of the 6th-order all-pass filter when the input signal is a 200 Hz sawtooth wave

Question 7: Are the graphical representations in the screen of your oscilloscope consistent with those in Figure 2 ? Explain why the represented LEFT and RIGHT signals are not the same given that the filters affecting both channels are all-pass filters sharing the same frequency response magnitude .

7 Conclusions

This Lab motivated the design, implementation and experimental test of a 6th-order IIR band-stop filter. The advantages of decomposing the filter into second-order section have been implicitly highlighted. Finally, a simple conversation of the 6th-order IIR band-stop filter into a 6th-order IIR all-pass filter has been motivated as well as the experimental verification of its impact on structured waves such as the sawtooth wave.

8 Additional References

L.EEC025 Fundamental of Signal Processing course materials (lectures slides, videos, and notes), especially the lecture slides on “The frequency domain characterization of discrete-time LTI systems”:

<https://moodle.up.pt/course/view.php?id=3853>