arm Education

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2023/2024 – 1st semester

Week08, 06 Nov 2023

Objectives:

-measuring the frequency response of 2nd-order IIR filters running in real-time on the STM32F746G Discovery board:

- **an All-Pole filter**
- **an All-Pass filter**

DSP Education Kit

LAB 7

Measuring the frequency response of an All-Pole and All-Pass filter

Issue 1.0

Contents

1 Introduction 1
1.1 Lab overview 1
2 Requirements 1
3 Preliminary analysis before the current lab 1
4 Measuring the Frequency Response 4
4.1 Setting-up for this lab experiment 6
4.2 Sketching the frequency response of the All-Pole and All-Pass filters 6
4.3 Testing the effect of different group delay responses when the magnitude frequency response is the same (optional) 8
5 Conclusions 10
6 Additional References 10

1 Introduction

1.1 Lab overview

The examples in this lab motivate the analysis and experimentation with second-order all-pole and all-pass filters sharing the same poles. A simple experimental method is used to estimate the frequency response magnitude of both filters, and to compare them to the theoretical one. Finally, a simple procedure is followed to assess, in real-time, the impact of the group delay of the second-order all-pass filter, with that of a trivial zero-order all-pass system.

2 Requirements

To carry out this lab, you will need:

- An STM32F746G Discovery board
- A PC running Keil MDK-Arm
- An oscilloscope
- 3.5 mm audio jack cables + BNC-BNC cables
- An audio frequency signal generator

3 Preliminary analysis before the current lab

The two 2nd-order IIR filters (or discrete-time systems) that we analyze in this lab experiment have the same transfer function denominator polynomials (which means that they have the same poles). These polynomials consist of a particular case given that they just depend on a parameter r, with $|r| < 1$. Thus, the two 2nd-order IIR filters have particular (in the sense of simplified) forms that are illustrated in Figure 1. In one of these cases, the location of the zeros depends on the location of the poles, and in the other case that does not happen.

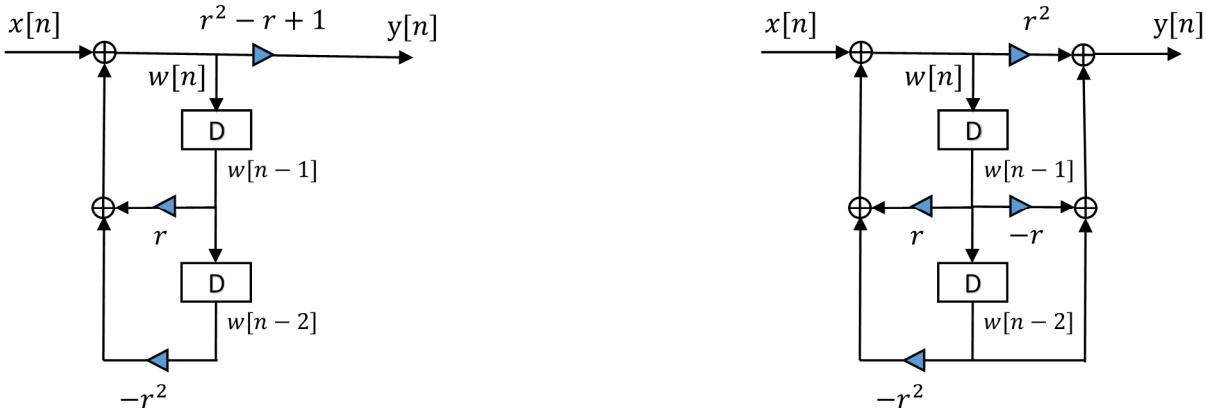


Figure 1: Particular 2nd-order All-Pole filter structure (on the left), and particular 2nd-order All-Pass filter structure (on the right).

Question 1 [2 pt / 10]: Using the Z-Transform, show that the transfer function of the All-Pole filter structure in Figure 1 is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{r^2 - r + 1}{1 - rz^{-1} + r^2 z^{-2}}, \quad |z| > r, \tag{1}$$

and that the transfer function of the All-Pass filter structure in Figure 1 is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{r^2 - rz^{-1} + z^{-2}}{1 - rz^{-1} + r^2 z^{-2}}, \quad |z| > r. \tag{2}$$

Question 2 [2 pt / 10]: Show that the magnitude frequency response of the All-Pole filter structure in Figure 1 is

$$\left|H\left(e^{j\omega}\right)\right| = \frac{\left|r^2 - r + 1\right|}{\sqrt{1 - 2r\cos(\omega) + r^2(1 + 2\cos(2\omega)) - 2r^3\cos(\omega) + r^4}}, \tag{3}$$

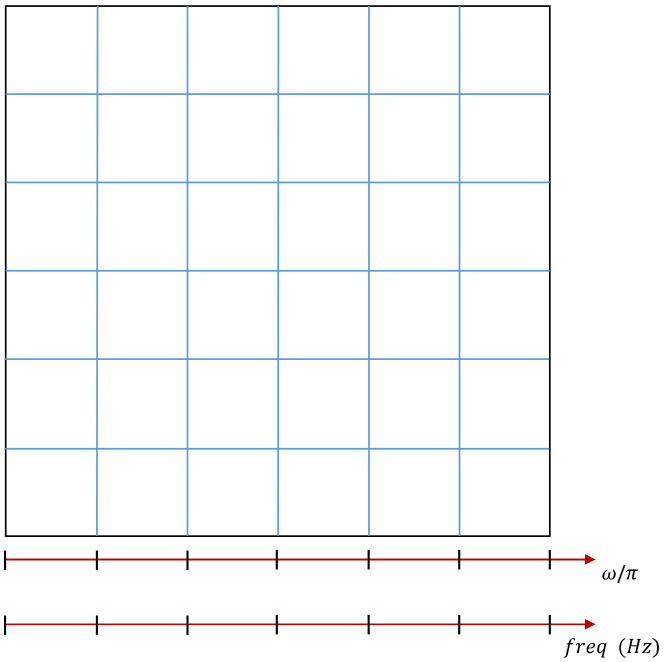
and that the magnitude frequency response of the All-Pass filter structure in Figure 1 is

$$\left|H\left(e^{j\omega}\right)\right| = 1. \tag{4}$$

Question 3 [1 pt / 10]: Validate Equation (3) in Matlab using the following code (which you need to complete):

```
r=0.8;
b=[(1-r+r*r)   0    0];
a=[1          -r    r*r];
frange=pi*[-1:1/256:1-1/256];
myH= please complete here ;
[H W]=freqz(b,a,frange);
plot(W/pi, abs(H))
hold on
plot(W/pi, myH,'m')
hold off
pause
```

Question 4 [1 pt / 10 **]**: represent graphically the magnitude frequency response according to Equation (3) using your answer to Question 3. Represent frequency using two different but equivalent frequency axes: ω/π (in the range [0, 1]) and `freq` (in Hz, in the range [0, F_N] where F_N is the Nyquist frequency).



In particular, take note of what the maximum gain is, and for which frequency it occurs. Relate this frequency to the poles that result from Equation (1).

Max gain: **Frequency:** **Frequency in Hz (assuming 8 kHz sampling frequency):**

Also, take note of the gain in Equation (3) when $\omega = 0$ rad, and when $\omega = \pi$ rad. These values represent important references for the experimental part of this lab (Section 4).

Gain for $\omega = 0$:

Gain for $\omega = \pi$:

4 Measuring the Frequency Response

In this lab experiment, we will use the modified `main()` project file that is named **`stm32f7_IIRorder2.c`** and that is available on the Moodle platform. Its C code is listed next.

```
// stm32f7_IIRorder2.c

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

#define SOURCE_FILE_NAME "stm32f7_IIRorder2.c"
#define N 2
#define r 0.8f
#define r2 r*r
#define Gain (r*r - r + 1)

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

float32_t w[N+1] = {0.0, 0.0 ,0.0};

enum filtertype{AllPole, AllPass};

void BSP_AUDIO_SAI_Interrupt_CallBack()
{
// when we arrive at this interrupt service routine (callback)
// the most recent input sample values are (already) in global variables
// rx_sample_L and rx_sample_R
// this routine should write new output sample values in
// global variables tx_sample_L and tx_sample_R
  int16_t i;
  float32_t w0, yn;

  // uncomment just one of the following two lines
  enum filtertype myfilter=AllPole;
  // enum filtertype myfilter=AllPass;

  w0  = (float32_t)(rx_sample_L);

  switch (myfilter)
  {
       case AllPole:
          w0 += ( (float32_t)(r) * w[N-1] - (float32_t)(r2) * w[N] );
          yn = (float32_t)(Gain) * w0;
          break;
       case AllPass:
          w0 += ( (float32_t)(r) * w[N-1] - (float32_t)(r2) * w[N] );
          yn = (float32_t)(r2) * w0 - (float32_t)(r) * w[N-1] +  w[N];
          break;
       default:
          yn = w0;
  }

  tx_sample_L = rx_sample_L;
  tx_sample_R = (int16_t)(yn);
```

```
  w[0] = w0;
  for (i=N ; i>0 ; i--) w[i] = w[i-1];

  BSP_LED_Toggle(LED1);

  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_9DB,
                      SOURCE_FILE_NAME,
                     NOGRAPH);

  while(1){}
}
```

Take a moment to analyze this C code in order to conclude on:

- what the sampling frequency is
- what lines of the code implement Equation (1)
- what lines of the code implement Equation (2)
- the fact that one output channel just uses the unmodified input (Left) channel.

Question 5: what is the sampling frequency that this code specifies ? What is the Nyquist frequency ?

Question 6: What type of filter (All-Pass or All-Pole) is implemented by the above version of the C code ?

After unzipping it, take the `stm32f7_IIRorder2.c` file 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\

Now, proceed as usual to start the Keil MDK-Arm development environment (µVision) and to replace the existing `main()` file in that project by the new `main()` that is `stm32f7_IIRorder2.c` .

Remember that the directory where you can find the 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.

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

4.1 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. Connect the output of a sinusoidal signal generator (i.e. the function generator) 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 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.

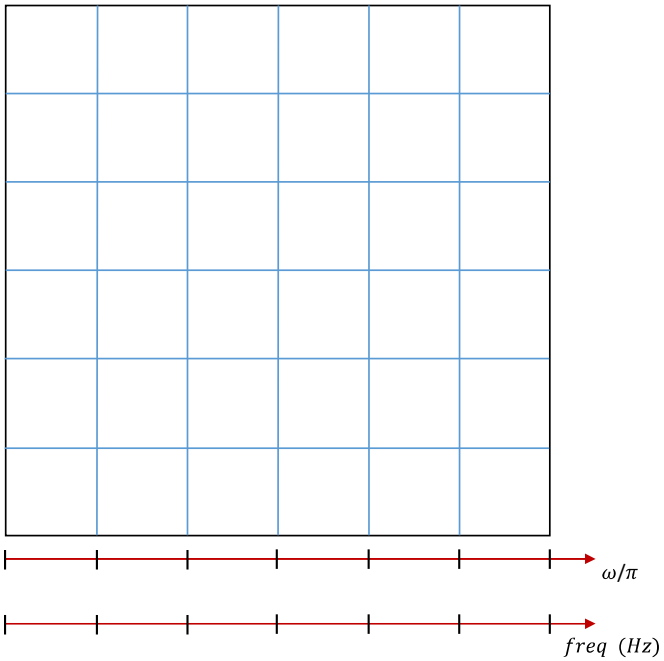
4.2 Sketching the frequency response of the All-Pole and All-Pass filters

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 number of different frequencies.

As the frequency of the inputs signal is varied, the amplitude of the output signal should change.

In this experiment, 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, for three frequencies only: 100 Hz (which we will take as an approximation for 0 Hz), 3800 Hz (which we will take as an approximation for 4 kHz), and the frequency that corresponds to a maximum gain, as anticipated in **Question 4**.

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

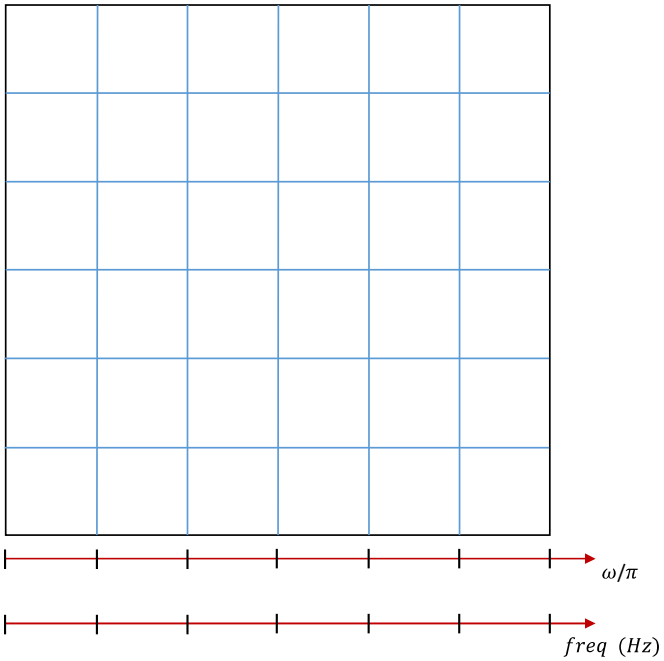


Question 7 [2 pt / 10 **]**: Is this sketch consistent with the expected gain according to your answer to Question 4 ? In particular, are the ratios between the maximum Vpp and minimum Vpp consistent with the ratios between maximum gain and minimum gain in Question 4 ?

Now, modify the `stm32f7_IIRorder2.c` file such that it implements the other type of 2nd-order IIR filter (i.e. if it is configured as All-Pole, change it to All-Pass). Now, compile the modified code, download it to the STM32F746G board and run it.

In this part, you should take the RIGHT channel of the STM32F746G LINE OUT output to the CHAN1 input of the oscilloscope (in *lieu* of the function generator output). Thus, the oscilloscope will be representing both LEFT and RIGHT output channels.

Sketch in the following plot the approximate frequency response of the new filter when you vary the frequency from 100 Hz up to 3.8 kHz.



Question 8 [2 pt / 10 **]**: Is this sketch consistent with the one that results from Equation (4) ?

4.3 Testing the effect of different group delay responses when the magnitude frequency response is the same (optional)

In this experiment, we keep uncommented in the `stm32f7_IIRorder2.c` C code the line `enum filtertype myfilter=AllPass;` which selects the All-Pass filter, as indicated next.

```
// uncomment just one of the following two lines
// enum filtertype myfilter=AllPole;
 enum filtertype myfilter=AllPass;
```

According to this setting, one LINE OUT output channel delivers the output of the above 2nd-order All-Pass filter, while the other channel just copies the input sample `x[n]` to `y[n]`, in other words, it implements a trivial all-pass discrete-time system whose difference equation is `y[n]=x[n]`.

After compiling the code, downloading it to the STM32F746G board and running it, make sure that the function generator generates a 5 Vpp sine wave. Take both LEFT and RIGHT channels of the STM32F746G board LINE OUT to the CHAN1 and CHAN2 inputs of the oscilloscope. Make a quick sweep of the input sine wave frequency between 100 Hz and 3.8 kHz (for example, in steps of 100 Hz) and confirm that the Vpp level of both LEFT and RIGHT channels is essentially constant and equal.

Question 9: When you perform the suggested sine sweep, you notice that the sinusoids in both LEFT and RIGHT channels are in-phase when the frequency of the input sine wave is 100 Hz, and then, as the input frequency increases, the two output sinusoids become increasingly out-of-phase, and then they become again almost in-phase. For what input frequency do you notice that the two output sinusoids are more strongly out-of-phase ? How do you explain that taking into consideration the conclusions of the previous experiments in this lab ?

Now, adjust the function generator to generate a sawtooth wave by setting a triangular wave with a 5% duty cycle, 5 Vpp and 250 Hz frequency. It you take this sawtooth wave to the oscilloscope, it should look like the one that is represented in Figure 2.

Question 10: Now, take both LEFT and RIGHT channels of the STM32F746G board LINE OUT to the CHAN1 and CHAN2 inputs of the oscilloscope. They should look like the waveforms that are represented in Figure 2. Recalling Lab #3 and the direct STM32F746G output to an input sawtooth wave, which oscilloscope channel reflects the All-Pass output to a sawtooth wave ? Why is that the two waveforms that are represented in the oscilloscope look different ?

NOTE: you should keep in mind that the frequency response magnitude in both channels is the same.

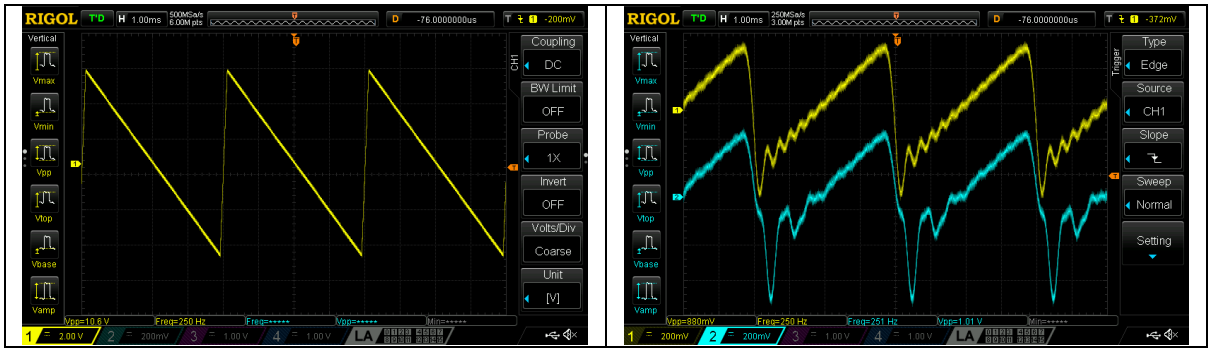


Figure 2: Left: input sawtooth wave (5 Vpp, 250 Hz, 5% Duty cycle). Right: R LINE OUT signal when y[n]=x[n] (yellow color), and L LINE OUT signal and when the 2nd-order All-Pass system is used (with r=0.8).

If you change the `r` parameter to 0.6 in a separate test, or to 0.85 in another test (remember that in each case you have to modify the C code, compile it, and download it to the STM32F7 board), you should obtain graphical representations in the oscilloscope that are similar to the waveforms in Figure 3. Which case corresponds to `r=0.6`, and which case corresponds to `r=0.85` ? Why is that in one of these cases the All-Pass output looks closer to the output of the trivial all-pass system (i.e. the one corresponding to the difference equation `y[n]=x[n]`) ?

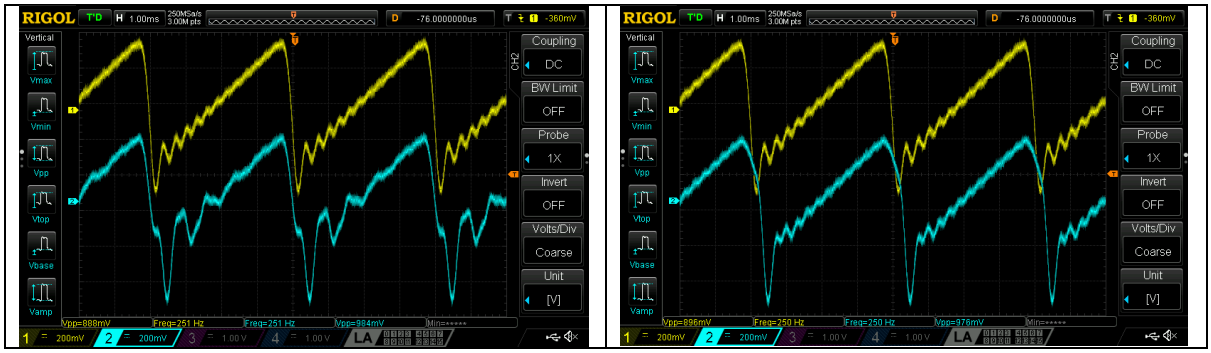


Figure 3: A case of R LINE OUT signal when y[n]=x[n] (yellow color), and L LINE OUT signal when the 2nd-order All-Pass system is used with r=0.6. And a case of R LINE OUT signal when y[n]=x[n] (yellow color), and L LINE OUT signal and when the 2nd-order All-Pass system is used with r=0.85.

5 Conclusions

This laboratory exercise motivated the implementation and real-time test of two 2nd-order IIR filters, one of them corresponding to an All-Pole system, and another one corresponding to an All-Pass system. The impact of the group delay of the All-Pass system has also been observed when the input waveform is a sine wave, or a sawtooth wave.

6 Additional References

FPS slides on the frequency domain characterization of discrete-time LTI systems

https://moodle.up.pt/mod/resource/view.php?id=24685