arm Education

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2023/2024 – 1st semester

Week11, 27 Nov 2023

Objectives:

-revisiting theoretical and practical aspects of special FIR filters:

- **linear-phase, minimum-phase and other phase responses**
- **multiple Z-domain zero configurations having the same frequency response magnitude**
- **design and implementation of a discrete-time differentiator and a Hilbert Transformer**

DSP Education Kit

LAB 9

Revisiting FIR Filters and Lab experiments involving an FIR differentiator and Hilbert Transformer

Issue 1.0

Contents

1 Introduction

1.1 Virtual Lab overview

The examples in this Lab revisit the theory of FIR filters and the underlying concepts of distribution of zeros on the Z-plane, and different alternatives for the phase response. This Lab also motivates the real-time implementation on the STM32F7 Discovery board of a discrete-time differentiator, and a Hilbert transformer, and discusses practical experimental results and implementation aspects.

2 Revisiting LAB 3 and FIR filter design (review)

We recall, here, Section 4 of LAB 3, in which the provided Matlab code (`FPS_lab03.c`) allowed the transformation of the impulse response of an FIR filter (`hfilter`) into the impulse response of another FIR filter (`hfilter2`) sharing the exact same frequency response magnitude. The specific part of the Matlab code that implemented that transformation is:

```
hroots=roots(hfilter); gain=sum(hfilter);
for k=1:length(hroots)
    mult=abs(hroots(k));
  if( mult > 1 )
        hroots(k) = 1/conj(hroots(k));
  end

end
hfilter2=poly(hroots);
hfilter2 = gain*hfilter2/sum(hfilter2);
```

Explain what the Matlab code is doing, and explain what is peculiar in the FIR filter having impulse response `hfilter2` with respect to the FIR filter having impulse response `hfilter`.

3 Exercise involving a 4th-order FIR (review)

Consider a causal 4th-order FIR discrete-time system whose zeros are located at $\alpha = re^{j\theta}$, α^*, $1/\alpha$, and $1/\alpha^*$. Admit that $r = 0.8$, and that $\theta = \pi/3$.

Program a Matlab M-file (i.e., a Matlab script with extension `.m`) that performs the following:

i)	uses the `poly()` Matlab command that takes as input the zeros of the system and finds the transfer function numerator polynomial "`b`",
ii)	uses the `poly()` Matlab command that takes as input the poles of the system and finds the transfer function denominator polynomial "`a`" (note: this may be skipped as the denominator polynomial is trivial),
iii)	uses the `impz()` Matlab command to find the impulse response of the system (use also the `stem()` Matlab command to represent the real part of the impulse response in blue, and to represent in red the imaginary part of the impulse response; overlap the latter to the former using the `hold on – hold off` commands),
iv)	uses the `zplane` Matlab command to represent the zero-pole diagram of the system,
v)	uses the `freqz()` Matlab command to find the frequency response of the system (use also the `plot()` Matlab command to represent magnitude of the frequency response),
vi)	uses the `grpdelay()` Matlab command to find the group delay response of the system (use also the `plot()` Matlab command to represent the group delay response).

You should use the `subplot()` Matlab command to split a figure into an array of 1×4 subfigures where the first represents the system impulse response (real and imaginary parts), the second represents the zero-pole diagram, the third represents the frequency response magnitude, and the fourth represent the system group delay. Your figure should look like the one represented next.
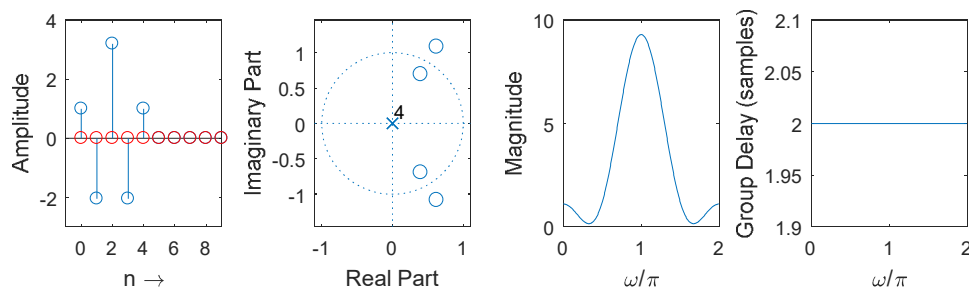
a) Based on the above information and admitting that the impulse response may be real-valued, or complex-valued, explain how many different 4th-order FIR discrete-time systems exist that have the exact same frequency response magnitude as that of the above system,

b) Of the number of possibilities indicated in **a)**, indicate how many correspond to linear-phase systems,

c) Of the number of possibilities indicated in **a)**, indicate how many correspond to minimum-phase systems,

d) Of the number of possibilities indicated in **a)**, indicate how many have a real-valued impulse response,

e) Demonstrate in Matlab how you can obtain (and illustrate…) at least two 4th-order FIR discrete-time systems that are different from the above system but that exhibit the exact same frequency response magnitude as that of the above system.

4 Requirements for the lab part

To carry out this lab, the following material is required:

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

# 5 The designed and tested FIR filters

The two FIR filters that we test in this lab correspond to the linear-phase discrete-time differentiator and Hilbert Transformer (FIR) filters, each of order 80, that are the object of the P2P Exercises 1 and 2 of the previous week. The impulse responses of those two designs are stored in two header files (`FPS_diff.h` and `FPS_HT.h`) that are included in the C code baseline to be used in this lab: `stm32f7_fir_intr_FPS_diff_HT.c` (and which you will need to complete). All three files are included in a ZIP that is available on Moodle.

Question 1 [1 pt / 10]: Assuming that the sampling frequency is 8 kHz, represent graphically the frequency response magnitude that you would expect to observe in the lab with respect to the discrete-time differentiator. 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).
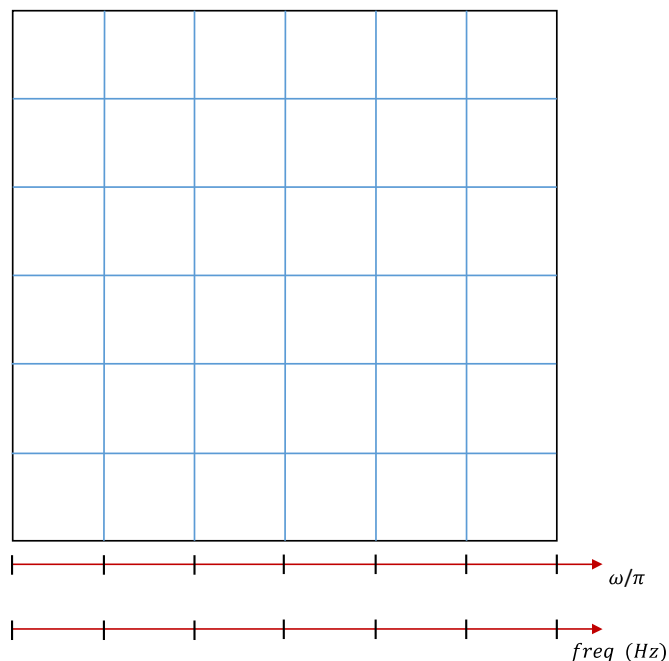
Question 2 [1 pt / 10]: Similarly to Question 1, assume that the sampling frequency is 8 kHz, and represent graphically the frequency response magnitude that you would expect to observe in the lab with respect to the Hilbert Transformer. Represent frequency using two different but equivalent

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).
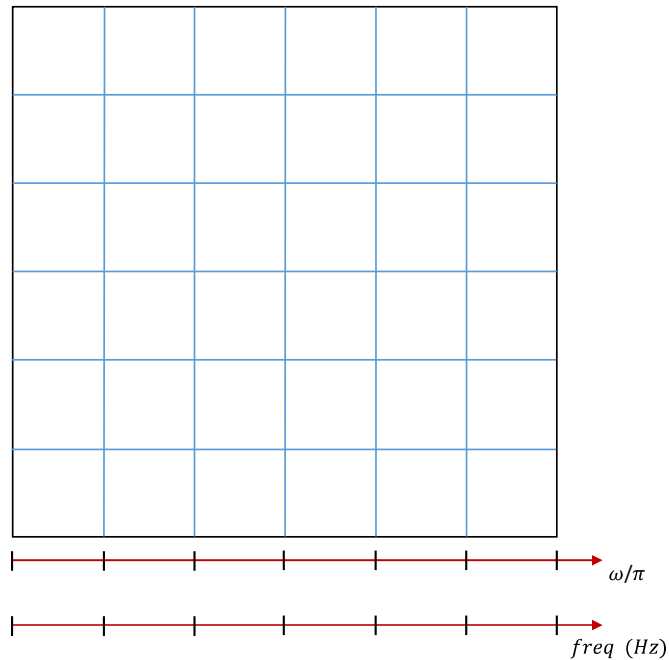
6 Completing the C code

In this lab experiment, we use the `main()` project file that is named **`stm32f7_fir_intr_FPS_diff_HT.c`** and that is available on the Moodle platform. Its C code, which is not complete, is listed next.

```
// stm32f7_fir_intr_diff_HT.c

#include "stm32f7_wm8994_init.h"
#include "FPS_diff.h"
#include "FPS_HT.h"
#include "stm32f7_display.h"
#define SOURCE_FILE_NAME "stm32f7_fir_intr_FPS_diff_HT.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;

#define DELAY (N-1)/2

enum filtertype{diff, HT};

float32_t x[N];
```

```
void BSP_AUDIO_SAI_Interrupt_CallBack()
{
  int16_t i,j;
  float32_t *hptr, xref, yn = 0.0;


  // uncomment just one of the following two lines
     enum filtertype myfilter=diff;
  // enum filtertype myfilter=HT;


  x[0]  = (float32_t)(rx_sample_L);
  BSP_LED_On(LED1);

  switch (myfilter)
  {
      case diff:
          hptr=hdiff;
          break;
      case HT:
          hptr=hHT;
          break;
      default:
          hptr=hHT;
  }

      for (i=0 ; i<N ; i++) yn += *(hptr++) * x[i];

      xref = please complete here ;

  for (j=(N-1) ; j>0 ; j--) x[j] = x[j-1];

  BSP_LED_Off(LED1);
  tx_sample_R = (int16_t)(yn);   // will appear in OUT LEFT channel
  tx_sample_L = (int16_t)(xref); // 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_9DB,
                      SOURCE_FILE_NAME,
                      NOGRAPH);

  while(1){}
}
```

Before compiling and running this code, you need to complete it (i.e., before the lab class!).

Question 3 [2 pt / 10]: in the C code, please locate the C code line corresponding to "`xref = please complete here ;`". This C code line selects the appropriate sample of the input signal $x[n]$ that will be used to compare $x[n]$ with the output $y[n]$ of either the FIR differentiator filter, or the FIR Hilbert Transformer.

Replace "`please complete here`" by one of the following five alternatives (only one is correct):

Alternative A:

```
        xref = x[0];
```

Alternative B:

```
        xref = x[DELAY-1];
```

Alternative C:

```
        xref = x[DELAY];
```

Alternative D:

```
        xref = x[DELAY+1];
```

Alternative E:

```
        xref = x[N-1];
```

Indicate what alternative is correct and explain why.

Question 4 [1 pt / 10]: The above implementation of the Hilbert Transformer is highly inefficient because it does not explore important opportunities to save arithmetic operations involving multiplications. What is the minimum number of multiplication that is strictly required to implement the Hilbert Transformer as specified in the `FPS_HT.h` header file (80, 40, or 20) ? Why ?

7 Testing the operation of the FIR differentiator and Hilbert Transformer

After completing the C code as recommended in the previous section, we are ready to compile the code, upload it to the STM32F7 board, and run it.

After unzipping them, take the `stm32f7_fir_intr_FPS_diff_HT.c` (which should be completed according to the suggestions in Section 6) and the `FPS_diff.h` and `FPS_HT.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 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 icon], or just [DSP_Education_Kit icon]. Double-click on this file/icon. This starts the Keil MDK-Arm development environment (µVision). Replace the existing `main()` file in that project by the new `main()` that is `stm32f7_fir_intr_FPS_diff_HT.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.

7.1 Operation of the FIR differentiator

We set the function generator to generate a square wave in one test (5 Vpp), or a triangular wave (50% duty cycle) in another test (5 Vpp). In both cases, the experimental setup is as illustrated in Figure 1. We connect the output of a function generator to the left channel of the LINE IN socket on the STM32F746G 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, as it is illustrated in Figure 1, using two BNC-BNC cables, we take the LEFT and RIGHT channels of the STM32F746G LINE OUT output to the two input channels of the oscilloscope.
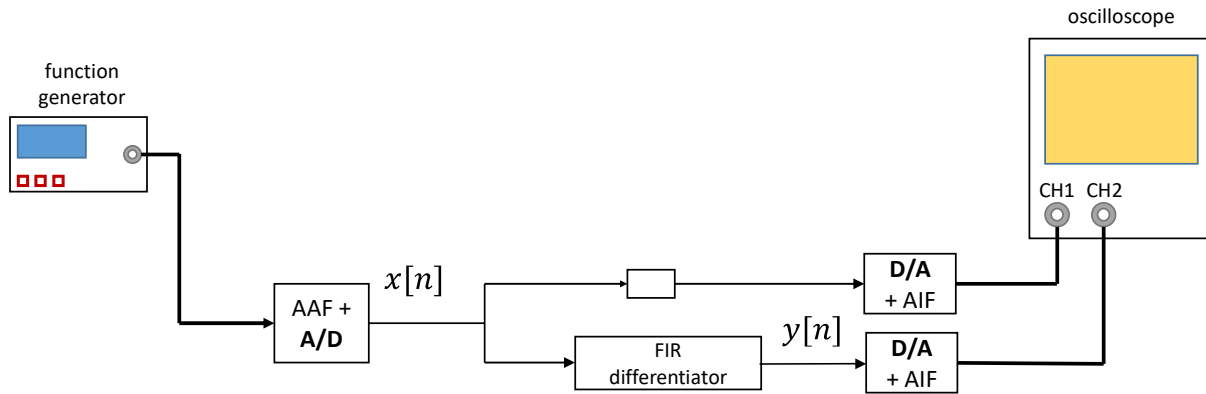
Figure 1: Experimental setup testing the discrete-time differentiator.

Proceed to obtain in the oscilloscope wave plots similar to those that are depicted in Figure 2, one case corresponding to the square wave input, and the other case corresponding to triangular wave input.
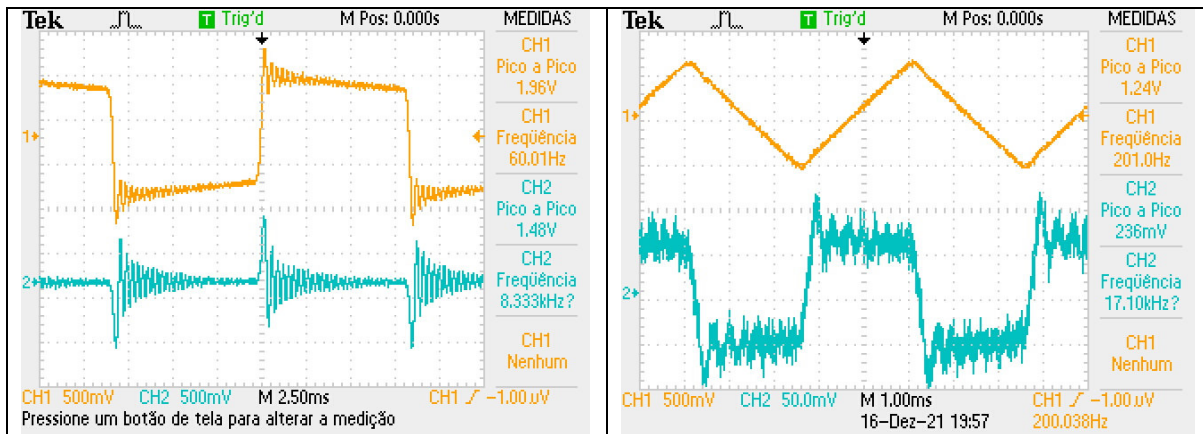

Figure 2: Oscilloscope screenshots for two different input signals: square wave and triangular wave.

Question 5 [2 pt / 10]: what color (yellow, or green) in Figure 2 reflects the input signal $x[n]$ (as in Figure 1), and what color corresponds to the output $y[n]$ of the FIR differentiator ? Why ? Is the output $y[n]$ consistent with the expected operation of a differentiator ?

7.2 Operation of the Hilbert Transformer

In many signal processing applications, it is necessary to generate the In-phase (I) and Quadrature (Q) components of a sinusoidal wave, as it is illustrated in Figure 3.
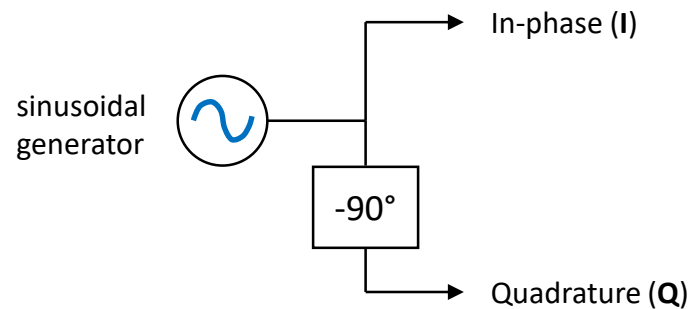

Figure 3: Schematic representation of the In-phase and Quadrature components of a sinusoidal wave.

The Quadrature component is obtained by shifting the phase of the In-phase sinusoidal component by -90° (or −π/2). This signal transformation can be performed by a Hilbert Transformer, also known as (quadrature) phase shifter.

Figure 4 illustrates the experimental setup that we use in order to test the operation of the Hilbert Transformer that was designed in the context of P2P Problem 2, as discussed above in Sections 5 and 6.
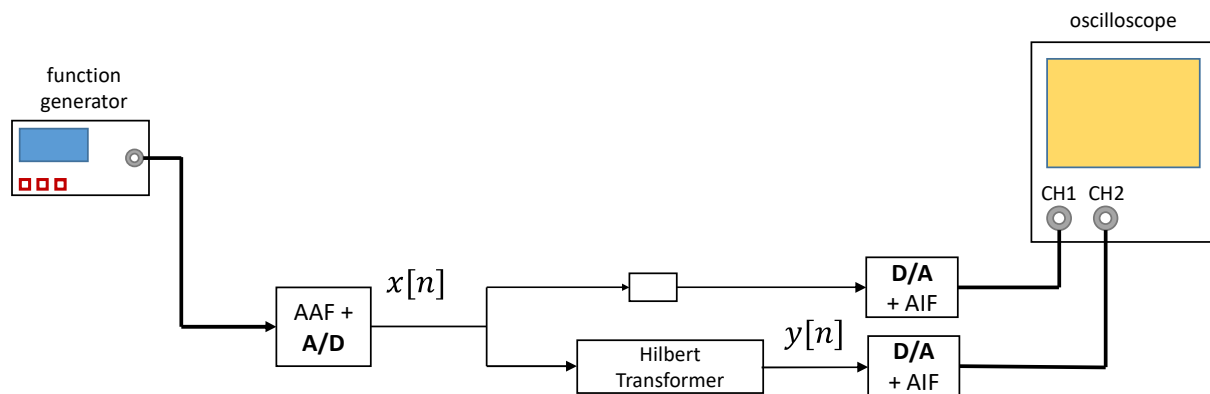

Figure 4: Experimental setup testing the Hilbert transformer.

We set the function generator to generate a sinusoidal wave (5 Vpp), and we vary its frequency from 60 Hz till about 3.8 kHz. Figure 5 illustrates the cases when the input frequency is 200 Hz, and 2500 Hz.
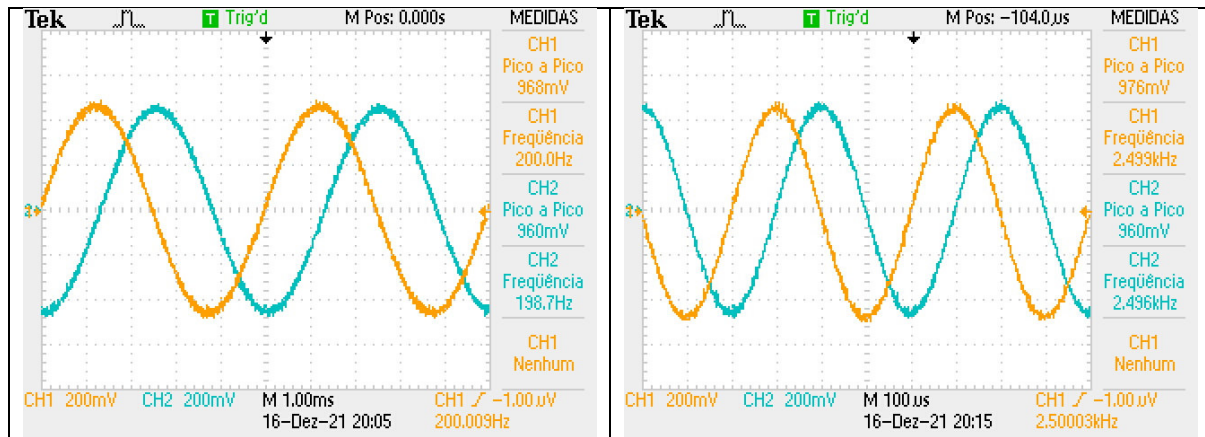
Figure 5: Oscilloscope screenshots for two sinusoid wave frequencies: 200 Hz and 2500 Hz.

Question 6 [2 pt / 10]: what color (yellow, or green) in Figure 5 reflects the input signal $x[n]$ (as in Figure 4), and what color corresponds to the output $y[n]$ of the Hilbert transformer ? Why ? Is the output $y[n]$ consistent with the expected operation of a Hilbert Transformer ?

Question 7 [1 pt / 10]: The constant phase relationship that is observed in Figure 5 and that does not depend on the frequency of the sinusoid is strictly maintained only if the empty building block in Figure 4 is correctly configured. What should this building block correspond to, precisely ? Why ?

Note: This question is obviously related to Question 3.

Now, we set the function generator in Figure 4 to generate a sawtooth wave (0% duty cycle, 5 Vpp) whose fundamental frequency is 200 Hz. As you know, given that this periodic wave is not a pure sinusoid, its spectral structure is harmonic. This means that when the sawtooth wave is subject to the filtering of the Hilbert Transformer, all of its harmonic sinusoids will be affected by a by -90° (or $-\pi/2$) phase shift. Figure 6 illustrates both input and filtered waves.
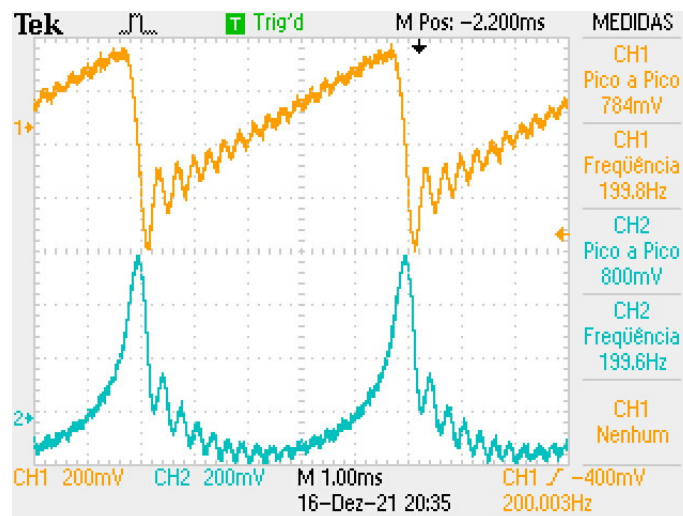

Figure 6: Oscilloscope screenshot representing the input and output of the Hilbert Transformer when the input signal is a 200 Hz sawtooth wave.

In order to assess if the output wave that is represented in Figure 6 is consistent with the expected output wave, we simulate the effect of the Hilbert Transformer by just converting the harmonic sinusoids of a theoretical sawtooth wave, to their quadrature versions. With that goal in mind, take as a baseline the following Matlab code that generates a theoretical sawtooth wave using 15 harmonics (in this simulation, for illustration purposes, we set the sawtooth wave fundamental frequency to 50 Hz).

```
% generate a 50Hz sawtooth wave
L=15;
FS=8000; T=1/FS;
t=[0:T:0.07-T];
freq=50;
% In-phase
x=zeros(1,length(t));
for k=1:L
    x = x - sin(2*pi*k*freq*t)/k;
end
subplot(1,2,1)
plot(t, x)
xlabel('Time (s)')
ylabel('Amplitude')
axis([0 0.07 -2 2])
```

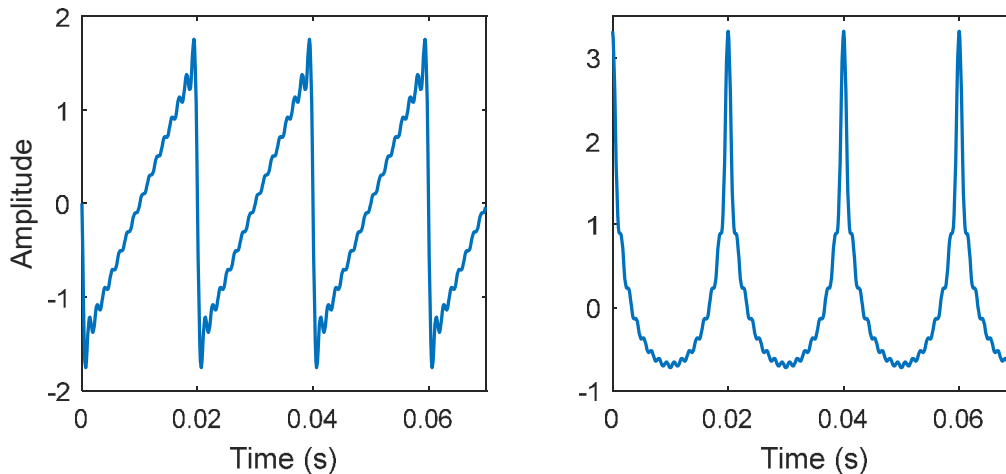The plot that this Matlab code produces is depicted on the left hand-side of Figure 7.

Figure 7: Synthesis of a 50 Hz sawtooth wave when all the harmonics are in-phase (figure on the left) and when all the harmonics are in quadrature (figure on the right).

Question 8: Adjust (yes, it is a simple adjustment) the above Matlab code such that it generates a new plot where all the harmonic sinusoids are in quadrature. You should obtain a plot similar to the one that is represented on the right hand-side of Figure 7. Is this theoretical result, which is obtained by simulation, consistent with the experimental result that is represented in Figure 6 ?

Question 9: Why is that the ripples in Figure 7 are distributed evenly while those in Figure 6 are not ?

Hint: Recall LAB 3 and, in particular, its Figure 2.

8 Conclusions

This laboratory has revisited the theory of FIR filters, the underlying concepts of Z-plane zero distribution, linear-phase, minimum-phase and other alternative phase responses, and has proposed a lab implementation of a discrete-time differentiator, and Hilbert transformer, which motivated a discussion on the practical relevance and interpretation of the corresponding results.

9 Additional References

L.EEC025 Fundamental of Signal Processing course materials (lectures slides, videos, and notes):

https://moodle.up.pt/course/view.php?id=1494